Configuring SSL Termination

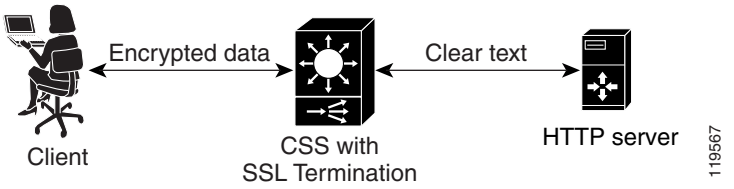
This chapter describes the steps required to configure a CSS as a virtual SSL server for SSL termination. It contains the following major sections:

Overview of SSL Termination

SSL termination in a CSS occurs when an SSL module, acting as a proxy server, terminates an SSL connection from a client, and then establishes a TCP connection to a server. When the module terminates the SSL connection, it decrypts the data and sends the data as clear text to the CSS for a decision on load balancing. The CSS transmits the data as clear text either to an HTTP server or back to the SSL module for encryption to a configured back-end SSL server.

Figure 4-1 illustrates an SSL connection between a client and a CSS configured with an SSL module acting as an SSL server.

Figure 4-1 SSL Termination



An SSL proxy list determines the flow of SSL information between the SSL module, the client, and the server. An SSL proxy list comprises one or more virtual SSL servers (related by index entry). An SSL module in the CSS uses the virtual SSL servers to properly process and terminate SSL communications between the client and the server. You can define a maximum of 256 virtual SSL servers for a single SSL proxy list.

After you create and configure the entries in a proxy list, you must activate the list, and then add the SSL proxy list to a service to initiate the transfer of SSL configuration data to the SSL module. When you activate the service, the CSS transfers the data to the module. Then you can add each SSL service to an SSL content rule.

Creating an SSL Proxy List

An SSL proxy list is a group of related virtual SSL servers that are associated with an SSL service. To create an SSL proxy list, use the **ssl-proxy-list** command.

You can access the ssl-proxy-list configuration mode from most configuration modes except for ACL, boot, group, rmon, or owner configuration modes. You can also use this command from the ssl-proxy-list configuration mode to access another SSL proxy list. Enter the SSL proxy list name as an unquoted text string from 1 to 31 characters.

For example, to create the SSL proxy list, ssl_list1, enter:

```
(config)# ssl-proxy-list ssl_list1
Create ssl-list <ssl_list1>, [y/n]: y
```

Once you create an SSL proxy list, the CLI enters into the ssl-proxy-list configuration mode.

```
(config-ssl-proxy-list[ssl_list1])#
```

To delete an existing SSL proxy list, enter:

```
(config)# no ssl-proxy-list ssl_list1
Delete ssl-list <ssl_list1>, [y/n]: y
```

Note You cannot delete an SSL proxy list if an SSL service is in use and contains the active SSL proxy list. You must first suspend the SSL service to delete a specific SSL proxy list.

Adding a Description to an SSL Proxy List

To specify a description for an SSL proxy list, use the **description** command. Enter the description as a quoted text string with a maximum of 64 characters, including spaces.

For example, to add a description to the *ssl_list1* SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# description "This is the SSL list
for www.brandnewproducts.com"
```

To remove the description from a specific SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no description
```

Configuring Virtual SSL Servers for an SSL Proxy List

This section discusses creating one or more virtual SSL servers for an SSL proxy list. Use the **ssl-server** command to define an index entry in the SSL proxy list that you then use to configure specific SSL parameters associated with the SSL proxy list. An SSL module in the CSS uses the virtual SSL servers to properly process and terminate SSL communications between the client and the server. You must define an ssl-server index number before configuring SSL proxy list parameters. You can define a maximum of 256 virtual SSL servers for a single SSL proxy list.

For example, suppose the e-commerce vendor Brand New Products, Inc. wants to configure the CSS to perform SSL termination. They need to divert all traffic intended for https://www.brandnewproducts.com to the SSL module in the CSS. To do this, they must identify a VIP address for a virtual SSL server in the SSL proxy list and link the list to the same VIP address as a content rule. The VIP address requires the following additional SSL configuration parameters:

- Identification of a virtual TCP port number that corresponds with a content rule
- An existing RSA or DSA certificate for identification purposes
- An appropriate SSL key pair to perform encryption and signing (assuming you are using an RSA key pair)
- Diffie-Hellman parameters if your CSS SSL security requires the Diffie-Hellman key exchange algorithm
- Assignment of a cipher suite

Note No modifications to an SSL proxy list are permitted on an active list. Suspend the list prior to making changes, and then reactivate the SSL proxy list once the changes are complete. The CSS sends any additions or changes to any active SSL services using the proxy list. For more information, see the "Modifying an SSL Proxy List" section.

The following sections describe how to create a virtual SSL server for an SSL proxy list:

- Creating an SSL Server Index
- Specifying a Virtual IP Address
- Specifying a Virtual Port
- Assigning Certificate, Key, and Cipher Suites for Server Authentication
- Configuring Client Authentication
- Configuring HTTP Header Insertion
- Specifying SSL or TLS Version
- Terminating a Client Connection with a TCP FIN Message Only
- Specifying Secure URL Rewrite
- Specifying SSL Session Cache Timeout
- Specifying SSL Session Handshake Renegotiation
- Configuring the Delay Time for SSL Queued Data
- Specifying SSL TCP Client-Side Connection Timeout Values
- Specifying SSL TCP Server-Side Connection Timeout Values
- Changing the Acknowledgement Delay for SSL TCP Connections
- Specifying the Nagle Algorithm for SSL TCP Connections
- Specifying the TCP Buffering for SSL TCP Connections

To view configuration information on an SSL proxy list, see Chapter 7, Displaying SSL Configuration Information and Statistics.

Creating an SSL Server Index

You must create a virtual SSL server before you can configure SSL proxy list parameters. To identify SSL-specific parameters for the SSL proxy list, use the **ssl-server** *number* command. This command creates a number (index entry) in the SSL proxy list that you use to configure specific SSL parameters associated with the virtual SSL server (for example, VIP address, certificate name, and key pair). Enter an integer from 1 to 256.

For example, to specify virtual SSL server 20, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20
```

To remove the virtual SSL server from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20
```

Specifying a Virtual IP Address

The SSL module uses this VIP address as the means to know which traffic it should accept. Ensure that the VIP address matches a VIP address configured in a content rule. Use the **ssl-server** *number* **vip address** *ip_or_host* command to specify a virtual IP (VIP) address. Enter a VIP address for the virtual SSL server that corresponds to an SSL content rule. See the "Configuring a Content Rule for SSL Termination" section.

Enter a valid VIP address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com).

Note When you use the mnemonic host name format for the VIP address, the CSS uses its Domain Name Service (DNS) facility to translates host names such as myhost.mydomain.com to IP addresses such as 192.168.11.1. If the host name cannot be resolved, the VIP address setting is not accepted and an error message appears indicating host resolution failure. For details on configuring a Domain Name Service, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*.

If the VIP address has not been defined for the virtual SSL sever when you activate the SSL proxy list (see the "Specifying the Nagle Algorithm for SSL TCP Connections" section), the CSS logs an error message and does not activate the SSL proxy list. When you activate a content rule with a configured SSL service, the CSS verifies that each VIP address configured in the content rule matches at least one VIP address configured in the SSL proxy list in each of the added services. If a match is not found, the CSS logs an error message and does not activate the content rule.

For example, to specify a VIP address for the virtual SSL server that corresponds to a VIP address configured in a content rule, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 vip address
192.168.3.6
```

To remove a VIP address from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 vip address
```

Specifying a Virtual Port

The SSL module uses the virtual port to know which traffic it should accept. Use the **ssl-server** *number* **port** *number* command to specify a virtual TCP port number for the virtual SSL server. Enter a TCP port number that corresponds with an SSL content rule, which uses the specified TCP port number.

Specify a port number from 1 to 65535. The default port is 443. Ensure that the specified port number matches the port configured in a content rule (see the "Configuring a Content Rule for SSL Termination" section).

If the virtual port has not been defined for the virtual SSL server when you activate the SSL proxy list (see the "Specifying the Nagle Algorithm for SSL TCP Connections" section), the CSS logs an error message and does not activate the SSL proxy list. When you activate a content rule with a configured SSL service, the CSS verifies that each virtual port configured in the content rule matches at least one port configured in the SSL proxy list in each of the added services. If a match is not found, the CSS logs an error message and does not activate the content rule.

For example, to specify a virtual port of 444, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 port 444
```

To reset the virtual port to the default of 443, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 port
```

Assigning Certificate, Key, and Cipher Suites for Server Authentication

The CSS supports server certificates that it sends to all clients for authentication. To identify a certificate with a virtual SSL server, you must assign the certificates and key that you have either imported to or generated on the CSS described in Chapter 3, Configuring SSL Certificates and Keys. You must also assign the cipher suite that correlates to the certificates and keys.

The following sections provide information for configuring server authentication:

- Specifying the RSA Certificate Name
- Specifying the RSA Key Pair Name
- Specifying the DSA Certificate Name
- Specifying the DSA Key Pair Name
- Specifying the Diffie-Hellman Parameter Filename
- Specifying Cipher Suites

Specifying the RSA Certificate Name

To identify the name of an RSA certificate association to be used in the exchange of a public/private key pair for authentication and packet encryption, use the **ssl-server** *number* **rsacert** *name* command. To see a list of existing RSA certificate associations, use the **ssl-server** *number* **rsacert ?** command.

The specified RSA certificate must already be loaded on the CSS and an association made (see Chapter 3, Configuring SSL Certificates and Keys). If there is not a proper RSA certificate association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined RSA certificate association named *rsacert*, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert myrsacert1
```

To remove an RSA certificate association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 rsacert
```

Specifying the RSA Key Pair Name

To identify the name of an RSA key pair association. RSA key pairs are required before another device (client or server) can exchange an SSL certificate with the CSS, use the **ssl-server** *number* **rsakey** *name* command. To see a list of existing RSA key pair associations, use the **ssl-server** *number* **rsakey ?** command.

The RSA key pair must already be loaded on the CSS and an association made (see Chapter 3, Configuring SSL Certificates and Keys). If there is not a proper RSA key pair association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined RSA key pair association named *rsakey*, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey myrsakey1
```

To remove an RSA key pair association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 rsakey
```

Specifying the DSA Certificate Name

To identify the name of a DSA certificate association that is to be used in the exchange of digital signatures, use the **ssl-server** *number* **dsacert** *name* command. To see a list of existing DSA certificate associations, use the **ssl-server** *number* **dsacert ?** command.

The specified DSA certificate must already be loaded on the CSS and an association made (see Chapter 3, Configuring SSL Certificates and Keys). If there is not a proper RSA certificate association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined DSA certificate association named *dsacert*, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dsacert mydsacert1
```

To remove a DSA certificate association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dsacert
```

Specifying the DSA Key Pair Name

DSA key pairs are used to sign packet data, and they are required before another device (client or server) can exchange an SSL certificate with the CSS. Use the **ssl-server** *number* **dsakey** *name* command to identify the name of a DSA key pair association. To see a list of existing DSA key pair associations, use the **ssl-server** *number* **dsakey ?** command.

The DSA key pair must already be loaded on the CSS and an association made (see Chapter 3, Configuring SSL Certificates and Keys). If there is not a proper DSA key pair association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined DSA key pair association named *dsakey*, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dsakey mydsakey1
```

To remove a DSA key pair association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dsakey
```

Specifying the Diffie-Hellman Parameter Filename

The Diffie-Hellman key exchange parameter file ensures that the two devices in a data exchange cooperate to generate a shared key for packet encryption and authentication. Use the **ssl-server** *number* **dhparam** *name* command to identify the name of a Diffie-Hellman key exchange parameter file association. To see a list of existing Diffie-Hellman key exchange parameter files, use the **ssl-server** *number* **dhparam ?** command.

The Diffie-Hellman parameter file must already be loaded on the CSS and an association made (see Chapter 3, Configuring SSL Certificates and Keys). If there is not a proper Diffie-Hellman parameter file association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

To specify a previously defined Diffie-Hellman parameter file association, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dhparam mydhparams1
```

To remove a Diffie-Hellman parameter file association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dhparam
```

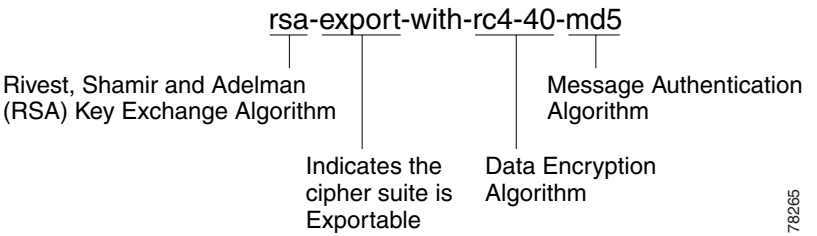
Specifying Cipher Suites

The SSL protocol supports a variety of different cryptographic algorithms, or ciphers, for use in operations such as authenticating the server and client to each other, transmitting certificates, and establishing session keys. Clients and servers may support different cipher suites, or sets of ciphers, depending on various factors such as the version of SSL they support, company policies regarding acceptable encryption strength, and government restrictions on export of SSL-enabled software. Among its other functions, the SSL handshake protocol determines how the server and client negotiate which cipher suites they will use to authenticate each other to transmit certificates and to establish session keys.

Note Exportable cipher suites are those cipher suites that are considered not to be as strong as some of the other cipher suites (for example, 3DES or RC4 with 128-bit encryption) as defined by U.S. export restrictions on software products. Exportable cipher suites may be exported to most countries from the United States, and provide the strongest encryption available for exportable products.

Each cipher suite specifies a set of key exchange algorithms. Figure 4-2 summarizes the algorithms associated with the rsa-export-with-rc4-40-md5 cipher suite.

Figure 4-2 Cipher Suite Algorithms



Use the **ssl-server** *number* **cipher** command to assign a cipher suite for the SSL proxy list. The cipher suite that you choose must correlate to the certificates and keys that you have either imported to or generated on the CSS. For example, if you choose **all-cipher-suites**, you must have an RSA certificate and key, a DSA certificate and key, and a Diffie-Hellman parameter file prior to activating the SSL proxy list.

For each available SSL version, there is a distinct list of supported cipher suites representing a selection of cryptographic algorithms and parameters. Your choice depends on your environment, certificates and keys in use, and security requirements. By default, no supported cipher suites are enabled.

The syntax for this command is:

ssl-server *number* **cipher** *name ip_address* or *hostname port* {**weight** *number*}

The options and variables are:

- **ssl-server** *number* - The number used to identify the virtual SSL server in the SSL proxy list.
- **cipher** *name* - The name of a specific cipher suite (as listed in Table 4-1).
- *ip_address* or *hostname* - The IP address to assign to the back-end content rule used with the cipher suite. Specify the IP address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com).
- *port* - The TCP port of the back-end content rule through which the back-end HTTP connections are sent.
- **weight** *number* - Optional parameter. Assigns a priority to the cipher suite, with 10 being the highest weight. By default, all configured cipher suites have a weight of 1. When negotiating which cipher suite to use, the SSL module selects from the client list based on the cipher suite configured with the highest weight. A higher weight will bias towards the specified cipher suite. To set the weight for a cipher suite, enter a number from 1 to 10. The default is 1.

For example, to select the *dhe-rsa-with-3des-ede-cbc-sha* cipher suite with an assigned weight of 5, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cipher
dhe-rsa-with-3des-ede-cbc-sha 192.168.11.1 80 weight 5
```

To remove a specific cipher suite from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 cipher
dhe-rsa-with-3des-ede-cbc-sha
```

Table 4-1 lists all supported cipher suites and values for the specific SSL server (and corresponding SSL proxy list). Table 4-1 also lists whether those cipher suites are exportable from the CSS, along with the authentication certificate and encryption key required by the cipher suite.

If you use the default setting or select the **all-cipher-suite** option, the CSS sends the suites in the same order as they appear in Table 4-1, starting with rsa-with-rc4-128-md5.

Note The **all-cipher-suites** setting works only when no specifically-defined ciphers are configured. To return to using the **all-cipher-suites** setting, you must remove all specifically-defined ciphers.

Caution The dh-anon series of cipher suites are intended for completely anonymous Diffie-Hellman communications in which neither party is authenticated. Note that this cipher suite is vulnerable to attacks.

Cipher suites with “export” in the title indicate that they are intended for use outside of the domestic United States and that they have encryption algorithms with limited key sizes.

Table 4-1 SSL Cipher Suites Supported by the CSS

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
all-cipher-suites	No	RSA certificate, DSA certificate	RSA key exchange, Diffie-Hellman
rsa-with-rc4-128-md5	No	RSA certificate	RSA key exchange
rsa-with-rc4-128-sha	No	RSA certificate	RSA key exchange
rsa-with-des-cbc-sha	No	RSA certificate	RSA key exchange
rsa-with-3des-ede-cbc-sha	No	RSA certificate	RSA key exchange
dhe-dss-with-des-cbc-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
dhe-dss-with-3des-ede-cbc-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
dhe-rsa-with-des-cbc-sha	No	RSA certificate	Ephemeral Diffie-Hellman key exchange

Table 4-1 SSL Cipher Suites Supported by the CSS (continued)

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
dhe-rsa-with-3des-ede-cbc-sha	No	RSA certificate	Ephemeral Diffie-Hellman key exchange
dh-anon-with-rc4-128-md5	No	Neither party is authenticated	Diffie-Hellman
dh-anon-with-des-cbc-sha	No	Neither party is authenticated	Diffie-Hellman
dh-anon-with-3des-ede-cbc-sha	No	Neither party is authenticated	Diffie-Hellman
dhe-dss-with-rc4-128-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
rsa-export-with-rc4-40-md5	Yes	RSA certificate	RSA key exchange
rsa-export-with-des40-cbc-sha	Yes	RSA certificate	RSA key exchange
dhe-dss-export-with-des40-cbc-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman key exchange
dhe-rsa-export-with-des40-cbc-sha	Yes	RSA certificate	Ephemeral Diffie-Hellman
dh-anon-export-with-rc4-40-md5	Yes	Neither party is authenticated	Diffie-Hellman
dh-anon-export-with-des40-cbc-sha	Yes	Neither party is authenticated	Diffie-Hellman
rsa-export1024-with-des-cbc-sha	Yes	RSA certificate	RSA key exchange
dhe-dss-export1024-with-des-cbc-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman
rsa-export1024-with-rc4-56-sha	Yes	RSA certificate	RSA key exchange
dhe-dss-export1024-with-rc4-56-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman

Configuring Client Authentication

For additional security, you can configure the SSL proxy server to request certificates from clients. By default, client certificate authentication is disabled. When you enable client authentication, the CSS requires the client to exchange a certificate during the SSL handshake. The CSS verifies that the:

- Client sending the certificate has a corresponding key
- Certificate has not expired
- Signature is valid
- Issuing CA has not revoked the certificate

You can configure how the CSS handles a certificate that has expired, is invalid, or has been revoked.

The following sections provide information on configuring client authentication:

- Enabling Client Authentication
- Specifying CA Certificates for Client Certificate Verification
- Configuring a CRL Record
- Assigning a CRL Record to the Virtual SSL Server
- Forcing the CSS to Download CRL Records
- Handling Client Authentication Failures

To view client authentication configuration information, use the **show ssl-proxy-list ssl-server** command. To view SSL counters for client authentication-related activities, use the **show ssl statistics** command. See Chapter 7, Displaying SSL Configuration Information and Statistics for more information.

Enabling Client Authentication

By default, client authentication is disabled on the CSS. The **authentication** option of the **ssl-server** command allows you to enable or disable client authentication. For example, to enable client authentication, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 authentication
enable
```

To reset the default setting of disabling client authentication, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 authentication
```

You can also reset the default setting of disabling client authentication by using the **disable** option. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 authentication
disable
```

After you enable client authentication on the CSS, you must specify a CA certificate that the CSS uses to verify client certificates.

Specifying CA Certificates for Client Certificate Verification

CA certificates contain the public key of the CA. If a server has the CA public key, it can verify that a client certificate was signed by the CA. If you assign a CA certificate to a virtual SSL server, the CSS uses the key in the certificate to verify the digital signature in the client certificate.

Note You must configure a CA certificate before you activate the SSL proxy list.

Before you configure the certificate on a virtual SSL server, you must import a CA certificate on the CSS and then associate it with a filename. For information on importing a CA certificate, see the "Importing or Exporting Certificates and Private Keys" section in Chapter 3, Configuring SSL Certificates and Keys. For information on associating a certificate with a filename, see the "Associating a Certificate with a File" also in Chapter 3, Configuring SSL Certificates and Keys.

You must configure at least one certificate; however, you can configure a maximum of four certificates. If you try to configure more than four certificates, the CSS displays an error message.

After you enable client authentication, you can assign the CA certificates to the virtual SSL server through the **ssl-server** *number* **cacert** command. For example, to specify the mycert1 CA certificate association to the virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cacert mycert1
```

To remove a certificate association from the virtual SSL server, use the **no** form of the **ssl-server** *number* **cacert** command. For example, to remove the mycert1 CA certificate association, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 cacert mycert1
```

Configuring a CRL Record

When a CA revokes a certificate, the CA places the certificate on a certificate revocation list (CRL) and publishes it for public availability. For the CSS to use a CRL, it must obtain the CRL from its current location and download it via HTTP. To do so, you must create a CRL record on the CSS. The CRL record contains the complete URL information for the CSS to obtain the CRL from its current location and import it periodically. After you configure the CRL record, you can assign it to the virtual SSL server.

Note The HTTP request to retrieve the CRL has a source IP address that is the VIP address of the virtual SSL server.

You can assign one to a virtual SSL server, however you can configure the CSS to store up to 10 CRL records. To configure the CRL record, use the **ssl crl-record** command in global configuration mode. The syntax for the command is:

ssl crl-record *crl_name url sign_cert hours*

The variables are:

- *crl_name* - The name for the CRL record. Enter a string with a maximum of 31 characters and no spaces.
- *url* - The URL where the CRL is located. Enter a string with a maximum of 168 characters and no spaces (for example, http://www.example.com/crl/clientcert.crl).

- *sign_cert* - The name of the CA certificate that signed the CRL. The CA certificate verifies that the CRL is authentic. You must import this certificate on the CSS before configuring the CRL. For information on importing a CA certificate, see the "Importing or Exporting Certificates and Private Keys" section in Chapter 3, Configuring SSL Certificates and Keys. For information on associating a certificate with a filename, see the "Associating a Certificate with a File" also in Chapter 3, Configuring SSL Certificates and Keys.
- *hours* - The number of hours to wait before retrieving an updated CRL. Enter a value from 0 to 2000. A value of 0 disables the retrieval of the CRL, which means that the CRL is not updated.

  The CSS SSL module keeps a list of all configured CRLs. The module only attempts to retrieve a CRL when:

  - The SSL proxy list containing CRL records is activated
  - The service or content rule associated with the SSL proxy list is activated
  - The CRL was previously retrieved and the time defined in the CRL record has now passed

The following example shows how to configure a CRL record named mycrl. The URL location of the CRL is crl.verisign.com. The CA certificate name on the CSS that authenticates the CRL is verisign_cacert. The CSS updates the CRL every 24 hours. Enter:

```
(config)# ssl crl-record mycrl http://crl.verisign.com/class1.crl
verisign_cacert 24
```

To remove the CRL record, enter:

```
(config)# no ssl crl-record mycrl
```

To view configuration information on a CRL, use the **show ssl crl-record** command. For more information on this command, see Chapter 7, Displaying SSL Configuration Information and Statistics.

Assigning a CRL Record to the Virtual SSL Server

After you configure the CRL record, you can assign it to the virtual SSL server. To assign the CRL record to the virtual SSL server, use the **ssl-server** *number* **crl** command. You can assign only one CRL record to a virtual SSL server. For example, to assign the mycrl CRL record, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 crl mycrl
```

To remove the mycrl CRL record from a virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 crl mycrl
```

Forcing the CSS to Download CRL Records

You can force the CSS to download all Certificate Revocation Lists (CRLs) or a specific CRL on any active SSL-proxy list on an active service configured with a service type of ssl-accel. To force the download, use the **ssl force-crl-reload** command in SuperUser mode.

To download a specific CRL, include its name when entering the command. For example, enter:

```
# ssl force-crl-reload mycrl
```

If you do not include a specific CRL name with this command, the CSS downloads any of the configured CRLs that are associated with an active SSL-proxy list on an active service. For example, enter:

```
# ssl force-crl-reload
```

You cannot download CRLs on a CSS in the backup state.

Handling Client Authentication Failures

A client certificate can fail if it is invalid, expired, or revoked by a CA. By default, when authentication of a client certificate fails on the CSS, the CSS rejects the client connection.

Note If a CSS cannot download the CRL, client connections will fail using a Revoked SSL alert. To verify that the CRL has successfully loaded, use the **show ssl statistics ssl** command.

You can configure how the CSS handles a failed client certificate through the **ssl-server** *number* **failure** command and the following options:

- **ignore** - The CSS ignores client authentication failures and allows both invalid and valid certificates to connect. For example, to configure the CSS to ignore client authentication failures, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure ignore
```

Note If you configure the **ignore** option, it may create a security risk.

- **reject** - Resets the CSS default behavior of rejecting the client connection when client authentication fails. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure reject
```

- **redirect** - The CSS sends connections of failed authentications to a configured URL.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure redirect
```

To configure the URL where the CSS redirects the client connection, use the **ssl-server** *number* **failure-url** command. Enter a URL with a maximum of 168 characters and no spaces. For example, to redirect the client connection to URL www.service_css.com when client authentication fails, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure-url
http://www.service_css.com
```

If you want to change an existing redirect URL, you must use the **no ssl-server** *number* **failure-url** command to remove it, and then reissue the **ssl-server** *number* **failure-url** command to configure the new URL. Note that you must suspend an activated virtual SSL server before modifying it.

For example, to remove the URL, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 failure-url
```

Note Regardless of the failure settings, the CSS logs a client authentication failure as an error message in syslog.

Configuring HTTP Header Insertion

During an SSL connection, a client may need to pass specific information to a back-end server. HTTP header insertion allows the embedding of information into an HTTP header during a client connection. For example, when a client connects to the virtual SSL server and the CSS decrypts the data, the CSS can insert information about the SSL session, and insert the client and server certificate into the HTTP request header, and then pass the header to the back-end server.

By default, HTTP header insertion only occurs on the first HTTP request for a persistent HTTP 1.1 connection. Subsequent requests within the same TCP connection are sent unmodified. For HTTP 1.0, in which persistence is not implemented, all HTTP requests contain the inserted header.

The CSS can insert one or more of the following information into the HTTP header after it decrypts the client data:

- Client certificate fields and associated information
- Server certificate fields and associated information
- SSL session fields and associated information
- Static text string

You can also:

- Add a prefix to the client certificate, server certificate, or SSL session fields inserted in the HTTP header. However, a prefix has no effect on insertion of a static text string.
- Modify a client certificate, server certificate, or SSL session HTTP header field.

The following sections provide information on HTTP header insertion:

- Inserting Client Certificate Information
- Inserting Server Certificate Information
- Inserting Session Information

- Adding a Prefix to the Fields Inserted in the HTTP Header
- Inserting a Static Text String
- Modifying an HTTP Header Insertion Field
- Inserting an HTTP Header in All HTTP Requests

To view configuration information on an HTTP header insertion, use the **show ssl-proxy-list** command. For more information on this command, see Chapter 7, Displaying SSL Configuration Information and Statistics.

Inserting Client Certificate Information

When you need to send client certificate information to the back-end server, you can configure the CSS to insert client certificate fields and associated information into the HTTP header. To add a prefix to the fields, see the "Adding a Prefix to the Fields Inserted in the HTTP Header" section.

Note If the SSL proxy list is active, suspend the s the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list.

To configure the CSS to insert client certificate fields in the HTTP header, use the **ssl-server** *number* **http-header client-cert** command. For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header
client-cert
```

To disable the insertion of client certificate fields and information in the HTTP header, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
client-cert
```

Table 4-2 lists the inserted client certificate fields, description, format, and an example. Depending on how the certificate was generated and what key algorithm was used, all of these fields may not be present for the certificate.

Table 4-2 Client Certificate Fields Inserted in the HTTP Header

Client Certificate Field	Description, Format, and Example
ClientCert-Fingerprint	Description: Hash Output Format: ASCII string of hexadecimal bytes separated by colons Example: ClientCert-Fingerprint: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1A:BA
ClientCert-Subject-CN	Description: X.509 subject's common name Format: String of characters representing the common name of the subject to whom the certificate has been issued Example: ClientCert-Subject-CN: www.cisco.com
ClientCert-Issuer-CN	Description: X.509 Certificate Issuer's Common Name Format: String of characters representing the common name for the certificate issuer Example: ClientCert-Issuer-CN: www.exampleca.com
ClientCert-Certificate-Version	Description: X.509 Certificate Version Format: Numerical X.509 version (3, 2, or 1), followed by the ASN.1 defined value for X.509 version (2, 1, or 0) in parentheses Example: ClientCert-Certificate-Version: 3 (0x2)
ClientCert-Serial-Number	Description: Certificate serial number Format: A whole integer value assigned by the certificate authority; this can be any arbitrary integer value Example: ClientCert-Serial-Number: 2
ClientCert-Data-Signature-Algorithm	Description: X.509 Hashing and Encryption Method Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 algorithm used to sign the certificate and algorithm parameters Example: ClientCert-Signature-Algorithm: md5WithRSAEncryption

Table 4-2 Client Certificate Fields Inserted in the HTTP Header (continued)

Client Certificate Field	Description, Format, and Example
ClientCert-DSA-Public-Key-Size	Description: The size in bits of the DSA public key, if the certificate contains a DSA key. Format: Number of bits as a whole integer followed by the word **bit** Example: ClientCert-DSA-Public-Key-Size: 1024 bit
ClientCert-DSA-Public-Key	Description: The DSA algorithm public key. Only used if the certificate contains a DSA key. Format: The key is printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character Example: ClientCert-DSA-Public-Key: 00:d8:1b:94:de:52:a1:20:51:b1:77
ClientCert-DSA-Private-Key-Size	Description: The size in bits of the DSA private key, if the certificate contains a DSA key. Format: Number of bits as a whole integer followed by the word **bit** Example: ClientCert-DSA-Private-Key-Size: 1024 bit
ClientCert-Subject	Description: X.509 subject's distinguished name Format: String of characters representing the subject that owns the private key being certified Example: ClientCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root
ClientCert-Issuer	Description: X.509 Certificate Issuer's Distinguished Name Format: String of characters representing the certificate authority that issued this certificate Example: ClientCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@exampleca.com, 0=Root

Table 4-2 Client Certificate Fields Inserted in the HTTP Header (continued)

Client Certificate Field	Description, Format, and Example
ClientCert-Not-After	Description: Certificate is not valid after this date Format: A universal time string or generalized time string in the Not After date of the Validity field Example: ClientCert-Not-After: 2003-1-27 23:59.59 UTC
ClientCert-Not-Before	Description: Certificate is not valid before this date Format: A universal time string or generalized time string in the Not Before date of the Validity field Example: ClientCert-Not-Before: 2002-1-27 00:00:00.00 UTC
ClientCert-Public-Key-Algorithm	Description: The algorithm used for the public key Format: The rsaEncryption, rsa, or dsaEncryption public key algorithm used to create the public key in the certificate Example: ClientCert-Public-Key-Algorithm: rsaEncryption
ClientCert-RSA-Modulus-Size	Description: Size of the RSA public key Format: Number of bits as a whole integer of the RSA modulus (typically 512, 1024, or 2048), followed by the word **bit** Example: ClientCert-RSA-Modulus-Size: 1024 bit
ClientCert-RSA-Modulus	Description: RSA modulus Format: The RSA algorithm modulus (n) printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character. Together with the exponent (e), this modulus forms the public key portion in the RSA certificate Example: ClientCert-RSA-Modulus: + 00:d8:1b:94:de:52:a1:20:51:b1:77
ClientCert-RSA-Exponent	Description: The public RSA exponent Format: Printed as a whole integer for the RSA algorithm exponent (e) Example: ClientCert-RSA-Exponent: 65537

Table 4-2 Client Certificate Fields Inserted in the HTTP Header (continued)

Client Certificate Field	Description, Format, and Example
ClientCert-Subject-Key-Identifier	Description: X.509 subject key identifier Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 subject key identifier Example: ClientCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99
ClientCert-Authority-Key -Identifier	Description: X.509 authority key identifier Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 Authority Key Identifier Example: ClientCert-Authority-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99
ClientCert-Basic-Constraints	Description: X.509 basic constraints Format: String listing whether the certificate subject can act as a certificate authority. Possible values are CA=TRUE or CA=FALSE basic constraints Example: ClientCert-Basic-Constraints: CA=TRUE
ClientCert-Signature-Algorithm	Description: Certificate signature algorithm Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 for the secure hash algorithm Example: ClientCert-Signature-Algorithm: md5WithRSAEncryption
ClientCert-Signature	Description: Certificate signature Format: Secure hash of the other fields in the certificate and a digital signature of the hash printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character Example: ClientCert-Signature: 33:75:8e:a4:05:92:65

Inserting Server Certificate Information

When you need to send server certificate information to the back-end server, you can configure the CSS to insert server certificate fields and associated information. The server certificate resides on the CSS and is configured with the **ssl-server** *number* **rsacert** or **dsacert** command. To add a prefix to the fields, see the "Adding a Prefix to the Fields Inserted in the HTTP Header" section.

Note If the SSL proxy list is active, suspend the proxy list before configuring or disabling HTTP header insertion. After, reactivate the SSL proxy list.

To configure the insertion server certificate information, use the **ssl-server** *number* **http-header server-cert** command. For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header
server-cert
```

To disable the insertion of server certificate fields and information in the HTTP header, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
server-cert
```

Table 4-3 lists the inserted server certificate fields and their descriptions. Depending on how the certificate was generated and what key algorithm was used, all of these fields may not be present for the certificate.

Table 4-3 Server Certificate Fields Inserted In the HTTP Header

Field	Description
ServerCert-Fingerprint	Description: Hash Output Format: ASCII string of hexadecimal bytes separated by colons Example: ServerCert-Fingerprint: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1A:BA
ServerCert-Subject-CN	Description: X.509 subject's common name Format: String of characters representing the common name of the subject to whom the certificate has been issued Example: ServerCert-Subject-CN: www.cisco.com

Table 4-3 Server Certificate Fields Inserted In the HTTP Header (continued)

Field	Description
ServerCert-Issuer-CN	Description: X.509 Certificate Issuer's Common Name Format: String of characters representing the common name for the certificate issuer Example: ServerCert-Issuer-CN: www.exampleca.com
ServerCert-Certificate-Version	Description: X.509 Certificate Version Format: Numerical X.509 version (3, 2, or 1), followed by the ASN.1 defined value for X.509 version (2, 1, or 0) in parentheses Example: ServerCert-Certificate-Version: 3 (0x2)
ServerCert-Serial-Number	Description: Certificate serial number Format: A whole integer value assigned by the certificate authority; this can be any arbitrary integer value Example: ServerCert-Serial-Number: 2
ServerCert-Data-Signature-Algorithm	Description: X.509 Hashing and Encryption Method Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 algorithm used to sign the certificate and algorithm parameters Example: ServerCert-Signature-Algorithm: md5WithRSAEncryption
ServerCert-DSA-Public-Key-Size	Description: The size in bits of the DSA public key, if the certificate contains a DSA key. Format: Number of bits as a whole integer followed by the word **bit** Example: ServerCert-DSA-Public-Key-Size: 1024 bit

Table 4-3 Server Certificate Fields Inserted In the HTTP Header (continued)

Field	Description
ServerCert-DSA-Public-Key	Description: The DSA algorithm public key. Only used if the certificate contains a DSA key. Format: The key is printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character Example: ServerCert-DSA-Public-Key: 00:d8:1b:94:de:52:a1:20:51:b1:77
ServerCert-DSA-Private-Key-Size	Description: The size in bits of the DSA private key, if the certificate contains a DSA key. Format: Number of bits as a whole integer followed by the word **bit** Example: ServerCert-DSA-Private-Key-Size: 1024 bit
ServerCert-Subject	Description: X.509 subject's distinguished name Format: String of characters representing the subject that owns the private key being certified Example: ServerCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root
ServerCert-Issuer	Description: X.509 Certificate Issuer's Distinguished Name Format: String of characters representing the certificate authority that issued this certificate Example: ServerCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@exampleca.com, 0=Root
ServerCert-Not-After	Description: Certificate is not valid after this date Format: A universal time string or generalized time string in the Not After date of the Validity field Example: ServerCert-Not-After: 2003-1-27 23:59.59 UTC

Table 4-3 Server Certificate Fields Inserted In the HTTP Header (continued)

Field	Description
ServerCert-Not-Before	Description: Certificate is not valid before this date Format: A universal time string or generalized time string in the Not Before date of the Validity field Example: ServerCert-Not-Before: 2002-1-27 00:00:00.00 UTC
ServerCert-Public-Key-Algorithm	Description: The algorithm used for the public key Format: The rsaEncryption, rsa, or dsaEncryption public key algorithm used to create the public key in a certificate Example: ServerCert-Public-Key-Algorithm: rsaEncryption
ServerCert-RSA-Modulus-Size	Description: Size of the RSA public key Format: Number of bits as a whole integer of the RSA modulus (typically 512, 1024, or 2048), followed by the word **bit** Example: ServerCert-RSA-Modulus-Size: 1024 bit
ServerCert-RSA-Modulus	Description: RSA modulus Format: The RSA algorithm modulus (n) printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character. Together with the exponent (e), this modulus forms the public key portion in the RSA certificate. Example: ServerCert-RSA-Modulus: + 00:d8:1b:94:de:52:a1:20:51:b1:77
ServerCert-RSA-Exponent	Description: The public RSA exponent Format: Printed as a whole integer for the RSA algorithm exponent (e) Example: ServerCert-RSA-Exponent: 65537

Table 4-3 Server Certificate Fields Inserted In the HTTP Header (continued)

Field	Description
ServerCert-Subject-Key-Identifier	Description: X.509 subject key identifier Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 subject key identifier Example: ServerCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99
ServerCert-Authority-Key-Identifier	Description: X.509 authority key identifier Format: ASCII string of hex bytes separated by colons for the X.509 version 3 Authority Key Identifier Example: ServerCert-Authority-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99
ServerCert-Basic-Constraints	Description: X.509 basic constraints Format: String listing whether the certificate subject can act as a certificate authority. Possible values are CA=TRUE or CA=FALSE Example: ServerCert-Basic-Constraints: CA=TRUE
ServerCert-Signature-Algorithm	Description: Certificate signature algorithm Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 for the secure hash algorithm Example: ServerCert-Signature-Algorithm: md5WithRSAEncryption
ServerCert-Signature	Description: Certificate signature Format: Secure hash of the other fields in the certificate and a digital signature of the hash printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters and separated by a colon (:) character Example: ServerCert-Signature: 33:75:8e:a4:05:92:65

Inserting Session Information

When you want to send SSL session information to the back-end server, you can configure the CSS to insert SSL session fields and associated information. To add a prefix to the fields, see the "Adding a Prefix to the Fields Inserted in the HTTP Header" section.

Note If the SSL proxy list is active, suspend the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list.

To configure the insertion of session information, use the **ssl-server** *number* **http-header session** command. For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header session
```

To disable the insertion of SSL session fields and information in the HTTP header, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
session
```

Table 4-4 lists the inserted SSL session fields and their descriptions.

Table 4-4 SSL Session Fields Inserted In the HTTP Header

Field	Description
Session-Cipher-Name	Description: Symmetric cipher suite Format: The OpenSSL version name of the cipher suite negotiated during this session Example: Session-Cipher-Name: EXP1024-RC4-SHA
Session-Cipher-Key-Size	Description: Symmetric cipher key size Format: Whole integer representing the length in bytes of the public key Example: Session-Cipher-Key-Size: 128

Table 4-4 SSL Session Fields Inserted In the HTTP Header (continued)

Field	Description
Session-Cipher-Use-Size	Description: Symmetric cipher use Format: Whole integer representing the length in bytes of the key used for symmetric encryption during this session Example: Session-Cipher-Use-Size: 56
Session-Protocol-Version	Description: Version of SSL or TLS Format: SSL or TLS protocol followed by version number Example: Session-Protocol-Version: TLSv1
Session-Id	Description: SSL Session ID Format: The 32-byte session ID negotiated during this session if a session ID is or has been negotiated, printed in big-endian format; hexadecimal without leading 0x and lowercase alphanumeric characters separated by a colon (:). Example: Session-Id: 75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25:75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25
Session-Verify-Result	Description: SSL session verify result Format: Numeric value indicating the SSL session verify result Example: Session-Verify-Result: 0

Adding a Prefix to the Fields Inserted in the HTTP Header

To add a prefix to the client certificate, server certificate, or session fields inserted in the HTTP header, use the **ssl-server** *number* **http-header prefix** command. Enter a quoted text string with a maximum of 16 characters.

Note If the SSL proxy list is active, suspend the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list.

For example, to add the Acme-SSL prefix to all inserted fields, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header prefix
"Acme-SSL"
```

The ClientCert-Certificate-Version field would now appear as Acme-SSL-ClientCert-Certificate-Version.

To reset the default of not including a prefix, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
prefix
```

Note The **ssl-server** *number* **http-header prefix** command affects only SSL fields that you configure to be inserted in the HTTP header. This command has no effect on the insertion of a static text string.

Inserting a Static Text String

The primary purpose of text string insertion in the HTTP header to a back-end server is to support Microsoft Outlook Web Access (OWA) applications; however, you may have other reasons to insert static text. To configure the insertion of a static text string, use the **ssl-server** *number* **http-header static** command. Enter a quoted text string with a maximum of 199 characters including spaces. For OWA support, enter the text string "FRONT-END-HTTPS: on".

Note If the SSL proxy list is active, suspend the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list.

For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header static
“FRONT-END-HTTPS: on”
```

You can also insert multiple strings on different lines by using the \r\n characters in between each line. The \r\n characters that terminate the lines use 4 of the 199 characters. The following example shows the insertion of three strings, "FRONT-END-HTTPS: on", "session cache: on", and "vip address: www.acme.com".

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header static
"FRONT-END-HTTPS: on\r\nsession cache: on\r\nvipaddress: www.acme.com"
```

To disable the insertion of the static string in the HTTP header and delete the string, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
static
```

Modifying an HTTP Header Insertion Field

Inserted HTTP header fields are not standard and vary based on the application and the implementation of the application. For this reason, the CSS allows you to configure the value of the HTTP header tag that is inserted into the session.

Note The HTTP header insertion functionality must be enabled for the insertion of the modified HTTP header field to occur. For information on enabling HTTP header insertion on the CSS, see the "Inserting Client Certificate Information", "Inserting Server Certificate Information", or "Inserting Session Information" sections.

To modify:

- A client-certificate field, use the **ssl-server** *number* **http-header client-cert-field** command.
- A server-certificate field, use the **ssl-server** *number* **http-header server-cert-field** command.
- A session field, use the **ssl-server** *number* **http-header session-field** command.

The syntax for these commands is:

ssl-server *number* [**http-header client-cert-field|server-cert-field|session-field**] *default_field* **"***configured_field***"**

The variables are:

- *default_field* - The default name of the client-certificate, server-certificate, or session field you want to modify.
  - For a list of default client certificate fields, see Table 4-2 (do not include the ClientCert- prefix) or enter:

    ```
    (config-ssl-proxy-list)# ssl-server 1 http-header
    client-cert-field ?
    ```

  - For a list of default server certificate fields, see Table 4-3 (do not include the ServerCert- prefix) or enter:

    ```
    (config-ssl-proxy-list)# ssl-server 1 http-header
    server-cert-field ?
    ```

  - For a list of default session fields, see Table 4-4 (do not include the Session- prefix) or enter:

    ```
    (config-ssl-proxy-list)# ssl-server 1 http-header session-field
    ?
    ```

- *configured_field* - The name you want to configure replacing the default field name. Enter the configured field as a quoted text string that contains from 1 to 36 characters. Do not enter a colon (:) character at the end of the text string. The CSS inserts the colon automatically.

  If you do not enter a text string in the quotes, the CSS sets the field to its default value.

For example, to change the ClientCert-Fingerprint client certificate field to Client-Fingerprint, enter:

```
(config-ssl-proxy-list)# ssl-server 1 http-header client-cert-field
Fingerprint "Client-Fingerprint"
```

To change the ServerCert-Fingerprint server-certificate field to Server-Fingerprint, enter:

```
(config-ssl-proxy-list)# ssl-server 1 http-header server-cert-field
Fingerprint "Server-Fingerprint"
```

To change the Session-Cipher-Name session field name to CipherName, enter:

```
(config-ssl-proxy-list)# ssl-server 1 http-header session-field
Cipher-Name "CipherName"
```

To reset a client-certificate, server-certificate, or session field to its default value, use the following command:

no ssl-server *number* [**http-header client-cert-field|server-cert-field| session-field**] *default_field*

The *default_field* variable is the default name of the client-certificate, server-certificate, or session field you modified. To view a list of default HTTP header insertion fields and their associated configured value, use the **show ssl-proxy-list** *list_name* command. For more information on this command, see Chapter 7, Displaying SSL Configuration Information and Statistics. Enter the default field as an unquoted text string.

Note Do not include the prefixes for the client-certificate, server certificate, and session fields. For example, for the ClientCert-Fingerprint field, enter Fingerprint.

For example, to reset the default ClientCert-Fingerprint client-certificate field, enter:

```
[(config-ssl-proxy-list)# no ssl-server 1 http-header
client-cert-field Fingerprint
```

To reset the default ServerCert-Fingerprint server-certificate field, enter:

```
(config-ssl-proxy-list)# no ssl-server 1 http-header server-cert-field
Fingerprint
```

To reset the default Session-Cipher-Name session field, enter:

```
(config-ssl-proxy-list)# no ssl-server 1 http-header session-field
Cipher-Name
```

Inserting an HTTP Header in All HTTP Requests

By default, HTTP header insertion only occurs on the first HTTP request for a persistent HTTP connection. Subsequent requests within the same TCP connection are sent unmodified. To insert HTTP headers in all HTTP requests within the same TCP connection, use the use the **ssl-server** *number* **http-header insert-per-request** command.

For example, enter:

```
(config-ssl-proxy-list)# ssl-server 1 http-header insert-per-request
```

To reset the default behavior of inserting an HTTP header only on the first HTTP request for a persistent HTTP connection, enter:

```
(config-ssl-proxy-list)# no ssl-server 1 http-header
insert-per-request
```

Specifying SSL or TLS Version

By default, the SSL version is SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

Use the **ssl-server** *number* **version** *protocol* command to specify the SSL or Transport Layer Security (TLS) protocol version. The options include:

- **ssl-tls** - SSL protocol version 3.0 and TLS protocol version 1.0 (default)
- **ssl** - SSL protocol version 3.0
- **tls** - TLS protocol version 1.0

For example, to specify SSL version 3.0, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 version ssl
```

To reset the SSL version to the default of SSL version 3.0 and TLS version 1.0, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 version
```

Terminating a Client Connection with a TCP FIN Message Only

Normally, the SSL Close-Notify alert terminates a client connection without an error. However, some versions of MSIE browsers do not close the connection upon receiving the Close-Notify alert. The browser may attempt to reuse the connection even though it appears to be closed to the CSS. Because the CSS cannot reply to a new request on this connection, the browser may display an error.

The **unclean-shutdown** option for the **ssl-server** command instructs the CSS to send only a TCP FIN message to terminate a client connection. The CSS does not send a Close-Notify alert to close a client connection.

For example, to configure the CSS to send only a TCP FIN message to terminate a client connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 unclean-shutdown
```

The **no** version of this command resets the CSS default behavior of sending both a Close-Notify alert and TCP FIN message to close the client connection. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 unclean-shutdown
```

Specifying Secure URL Rewrite

Client HTTPS connections can become HTTP connections when sent to a back-end server through a virtual SSL server in the SSL proxy list. The back-end server receives data as clear text from the client in the HTTP connection. If the server performs an HTTP 300-series redirect to another HTTP URL, the redirect causes the client to perform an HTTP request even though the client originally had been performing an HTTPS request. Because the client's connection changes to HTTP, the requested data may not be available from the server using a clear text connection.

Note Do not specify secure URL rewrite as a configuration parameter for the virtual SSL server if you plan to include one or more back-end SSL servers in the SSL proxy list (as described in the "Specifying the Nagle Algorithm for SSL TCP Connections" section).

You can avoid problems with nonsecure HTTP redirects from the back-end server by configuring one or more URL rewrite rules. Each rewrite rule is associated with a virtual SSL server in the SSL proxy list. URL rewrite rules resolve the problem of a web site redirecting the user to a nonsecure HTTP URL by rewriting the domain from http:// to https://. By using URL rewrite, all client connections to the Web server will be SSL, ensuring the secure delivery of HTTPS content back to the client.

Use the **ssl-server** *number* **urlrewrite** command to add a URL rewrite rule to the virtual SSL server to avoid nonsecure HTTP 300-series redirects. This command instructs the CSS, through the SSL module, to examine every HTTP header field received from the server for a 300-series redirection response (such as 302 Found or 304 Not Modified). If the CSS finds a 300-series return code, it searches the Location Response-Header field in the HTTP header to determine if the field matches the hostname defined in a URL rewrite rule. If there is a match, the CSS rewrites the Location field to contain an HTTPS location and the SSL port for the response.

For example, to define the following URL rewrite rule, keeping the default of port 443 for the SSL port and port 80 for the clear text port, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22
www.website.com
```

In this case, all HTTP redirects to http://www.website.com/ are rewritten in the SSL module as https://www.website.com/ and forwarded to the client.

The CSS supports the use of wildcards in domain hostnames as part of the matching criteria for a URL redirect rule. Include an asterisk (*) wildcard character in the domain name to identify more than one host in a single domain. You can specify a wildcard-only hostname (for example, *), a prefix wildcard (for example, *.mydomain.com), or a suffix wildcard (for example, www.mydomain.*). When using a wildcard-only hostname, the entire domain name is the * (asterisk) character and all HTTP redirects that come through this VIP address from the server are rewritten to HTTPS. In this case, there is no need to have additional URL rewrite rules for the SSL server.

Note Use care when specifying wildcards to avoid unwanted rewriting of all URL references by the SSL module. Review your redirects and ensure that every URL that matches a specified wildcard rule needs to be rewritten.

The CSS performs a URL rewrite search in the follow order:

1. Exact match.
2. Postfix wildcard match using the shortest prefix (for example, will match on "ssl-server 1 urlrewrite 7 cis*" before matching on "ssl-server 1 urlrewrite 12 cisco.*").
3. Prefix wildcard match using the shortest match (for example, will match on "ssl-server 1 urlrewrite 7 *.cis" before matching on "ssl-server 1 urlrewrite 12 *.cisco".
4. Wildcard match (for example, ssl-server 1 urlrewrite 7 *).

The syntax for the **ssl-server** *number* **urlrewrite** command is:

ssl-server *number* **urlrewrite** *number hostname* [**sslport** *port* {**clearport** *port*}]

The options and variables are:

- **ssl-server** *number* - The number used to identify the virtual SSL server in the SSL proxy list.
- **urlrewrite** *number* - The number of the URL rewrite rule to be added to the virtual SSL server. Enter a value from 1 to 32 corresponding to the URL rewrite rule. You can add a maximum of 32 URL rewrite rules to each SSL server for handling HTTP to HTTPS redirects.
- *hostname* - The domain name of the URL to be redirected (for example, www.mydomain.com). Enter an unquoted text string with a maximum length of 240 characters that corresponds to the domain name of the URL rewrite host. Do not include the directory path as part of the hostname. If you intend to use wildcards in domain names to identify and match on more than one host in a single domain, insert an asterisk (*) wild card character in the domain name.
- **sslport** *port* - (Optional) Specifies the port used for SSL network traffic. Enter a TCP port number that corresponds with an SSL content rule, which uses the specified TCP port number. The SSL module rewrites an HTTP redirect matching the URL redirect rule with the specified SSL port (or default port 443 if no port number is specified). Enter a port value from 1 to 65535. The default value is 443.

- **clearport** *port* - (Optional) Specifies the port used for clear text network traffic. The SSL module matches redirects in the Location Response-Header field with the specified clear text port (or default port 80 if no port number is specified). Enter a port value from 1 to 65535. The default value is 80.

For example, to specify URL rewrite 22 for *www.mydomain.com* using port 444 for SSL traffic and port 81 for clear text, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22
www.mydomain.com sslport 444 clearport 81
```

To remove URL rewrite rule 22, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 urlrewrite 22
```

For example, for the HTTP URLs www.sales.acme.com and www.services.acme.com, you could include the wildcard asterisk (*) character as follows to match on the two URLs (keeping the default of port 443 for the SSL port and port 80 for the clear text port):

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 1
*.acme.com
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 2
*.acme.com
```

Or, you could include the wildcard asterisk (*) character for the HTTP URLs www.acmesales.com and www.acmeservices.com as follows:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 1
www.acme*
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 2
www.acme*
```

To view statistical information on SSL URL rewrite, see Chapter 7, Displaying SSL Configuration Information and Statistics.

Specifying SSL Session Cache Timeout

In SSL, a new session ID is created every time the client and the CSS SSL module go through a full key exchange and establish a new master secret key. Specifying an SSL session cache timeout allows the SSL module to reuse the master key on subsequent connections with the client, which can speed up the SSL negotiation process. You can specify a timeout value to set the total amount of time an SSL session ID remains valid before the SSL module requires the full SSL handshake to establish a new SSL session.

The selection of an SSL session cache timeout value is important when using the **advanced-balance ssl** load-balancing method for a Layer 5 content rule to help fine-tune the SSL session ID that is used to stick the client to the server.

Use the **ssl-server** *number* **session-cache** *seconds* command to configure the SSL module to resume connection with a client using a previously established secret key. Enter an SSL session cache timeout value in seconds, from 0 (SSL session ID reuse disabled) to 72000 (20 hours). The default is 300 seconds (5 minutes). By disabling this option (entering a value of 0), the full SSL handshake occurs for each new connection between the client and the SSL module.

Note We do not recommend specifying a zero value for the **ssl-server** *number* **session-cache** *seconds* command. A non-zero value ensures that the SSL session ID is reused to improve CSS performance.

For example, to configure the reuse of an SSL session ID with a client using a timeout value of 10 hours, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 session-cache 36000
```

To reset the SSL session reuse timeout to the default of 300 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 session-cache
```

Specifying SSL Session Handshake Renegotiation

The SSL session handshake commands send the SSL HelloRequest message to a client to restart SSL handshake negotiation. SSL rehandshake is useful when a connection has been established for a lengthy period of time and you want to ensure security by reestablishing the SSL session.

Use the **ssl-server** *number* **handshake data** *kbytes* command to specify the maximum amount of data to be exchanged between the CSS and the client, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. By setting the data value, you force the SSL session to renegotiate a new session key after a session has transferred the specified amount of data. Specify an SSL handshake data value in Kbytes, from 0 (handshake disabled) to 512000. The default is 0.

For example, to configure an SSL rehandshake message for the SSL proxy list after a data exchange of 125000 Kbytes is reached with the client, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 handshake data
125000
```

To disable the rehandshake data option, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake data
```

Use the **ssl-server** *number* **handshake timeout** *seconds* command to specify a maximum timeout value, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. Setting a timeout value forces the SSL session to renegotiate a new session key after a session has lasted the defined number of seconds. The selection of an SSL rehandshake timeout value is important when using the **advanced-balance ssl** load-balancing method for a Layer 5 content rule to fine-tune the SSL session ID used to stick the client to the server. Specify an SSL handshake timeout value in seconds, from 0 (handshake disabled) to 72000 (20 hours). The default is 0.

For example, to configure an SSL rehandshake message after a timeout value of 10 hours has elapsed, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 handshake timeout
36000
```

To disable the rehandshake timeout option, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake timeout
```

Note If a connection is stuck using SSL sticky, be aware that the connection loses SSL sticky persistence each time that the CSS performs handshake renegotiation because the SSL session ID regenerates within an existing TCP flow. Because of this situation, the CSS is not aware of the new SSL session ID. When the next TCP connection comes in for this SSL flow, the CSS considers it as a new SSL session and load balances the connections to an SSL service. If there is more than one service and multiple SSL modules, the CSS may send the connection to a different SSL module. The connection will be a new SSL session to that SSL module, which causes the connection to be renegotiated for a second time. After the second renegotiation, the CSS is aware of the SSL session ID and the SSL session sticks to the other SSL module.

In this case, turning on SSL rehandshaking can cause SSL sessions to require additional resources to perform handshake renegotiation. If you are operating in a high traffic environment, this may impact overall SSL performance.

Configuring the Delay Time for SSL Queued Data

SSL on the CSS queues packet data from the server and encrypts it for transmission to the client. SSL empties the data from the queue and encrypts it for transmission to the client when:

- The queue fills to 16,400 bytes (the maximum SSL record size)
- The server sends a TCP FIN packet
- When the delay time on the CSS has passed, even though the queue has less than 16,400 bytes

For efficiency, SSL encrypts data into SSL records with a maximum size of 16,400 bytes. In an attempt to fully queue 16,400 bytes for encryption, SSL delays the emptying of the queue data for encryption.

You can use the **ssl-server** *number* **ssl-queue-delay** *ms* command to set the amount of time for the CSS virtual SSL server to wait before emptying the queued data for encryption. The default delay is 200 milliseconds. Enter a delay time value in milliseconds from 0 (disabled) to 10000.

Setting the delay value to 0 disables the queuing of data. The virtual SSL server on the CSS encrypts the data as soon as it arrives from the server and then sends the data to the client.

For example, to configure a delay time value of 400 milliseconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 ssl-queue-delay 400
```

To reset the delay time to the default of 200 milliseconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 ssl-queue-delay
```

Specifying SSL TCP Client-Side Connection Timeout Values

The TCP connection between the CSS and a client is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over the TCP connection between the CSS SSL module and a client.

To configure an SSL proxy list virtual SSL server for termination of a TCP connection with the client, see the following sections:

- Specifying a TCP SYN Timeout Value (Client-Side Connection)
- Specifying a TCP Inactivity Timeout Value (Client-Side Connection)

Specifying a TCP SYN Timeout Value (Client-Side Connection)

The CSS SYN timer counts the delta between the CSS sending the SYN/ACK and the client replying with an ACK as the means to terminate the TCP three-way handshake. Use the **ssl-server** *number* **tcp virtual syn-timeout** *seconds* command to specify a timeout value that the CSS uses to terminate a TCP connection with a client that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN inactivity timeout value in seconds, from 1 to 3600 (1 hour). The default is 30 seconds.

Note The connection timer should always be less than the retransmit termination time for new SSL and TCP connections.

For example, to configure a TCP SYN timeout of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
syn-timeout 1800
```

To reset the TCP SYN timeout to the default of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual
syn-timeout
```

Specifying a TCP Inactivity Timeout Value (Client-Side Connection)

The TCP inactivity timeout begins once the CSS receives an ACK from the client to terminate the TCP three-way handshake. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the **ssl-server** *number* **tcp virtual inactivity-timeout** *seconds* command to specify a timeout value that the CSS uses to terminate a TCP connection with the client when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure a TCP inactivity time of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
inactivity-timeout 1800
```

To reset the TCP inactivity timer to the default of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual
inactivity-timeout
```

Specifying SSL TCP Server-Side Connection Timeout Values

The TCP connection between the CSS and a server is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over TCP connections between the CSS SSL module and a server.

To configure an SSL proxy list virtual SSL server for termination of a TCP connection with the server, see the following sections:

- Specifying a TCP SYN Timeout Value (Server-Side Connection)
- Specifying a TCP Inactivity Timeout Value (Server-Side Connection)

Specifying a TCP SYN Timeout Value (Server-Side Connection)

The TCP SYN timer counts the delta between the CSS initiating the back-end TCP connection by transmitting a SYN and the server replying with a SYN/ACK. Use the **ssl-server** *number* **tcp server syn-timeout** *seconds* command to specify

a timeout value that the CSS uses to end a TCP connection with a server that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN timeout value in seconds, from 1 to 3600 (1 hour). The default is 30 seconds.

Note The connection timer should always be less than the retransmit termination time for new SSL and TCP connections.

For example, to configure a TCP SYN timeout of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server
syn-timeout 1800
```

To reset the TCP SYN timeout to the default of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server
syn-timeout
```

Specifying a TCP Inactivity Timeout Value (Server-Side Connection)

The TCP inactivity timeout begins once the CSS receives a SYN/ACK from the server. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the **ssl-server** *number* **tcp server inactivity-timeout** *seconds* command to specify a timeout value that the CSS uses to terminate a TCP connection with a server when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure a TCP inactivity time of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server
inactivity-timeout 1800
```

To reset the TCP inactivity timer to the default of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server
inactivity-timeout
```

Changing the Acknowledgement Delay for SSL TCP Connections

By default, the acknowledgement delay on a client or server connection is 200 milliseconds (ms). You can disable or adjust the SSL TCP timer length for delayed acknowledgements by using the following command:

ssl-server *server-num* **tcp virtual|server ack-delay** *value*

The *value* variable is the timer length in milliseconds (ms) for delayed acknowledgements. The default value is 200. Enter a value from 0 to 10000. A value of 0 disables the acknowledgement delay in receiving SSL traffic from the client. Disabling the timer improves the performance for sessions using the SSL session cache (Session ID Reuse).

For example, to set an acknowledgement delay of 400 ms for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
ack-delay 400
```

To set an acknowledgement delay of 400 ms for the TCP connection between the server and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server ack-delay
400
```

Specifying the Nagle Algorithm for SSL TCP Connections

The TCP Nagle algorithm automatically concatenates a number of small buffer messages transmitted over the TCP connection between a client and the SSL module or between a server and the SSL module. This process increases the throughput of your CSS by decreasing the number of packets sent over each TCP connection. However, the interaction between the Nagle algorithm and the TCP delay acknowledgment may increase latency in your TCP connection. Disable the Nagle algorithm when you observe an unacceptable delay in a TCP connection (clear-text or SSL).

- Use the **ssl-server** *number* **tcp virtual nagle** command to disable or reenable the Nagle algorithm for the TCP connection between the client and the SSL module. The syntax for this command is:

  ssl-server *number* **tcp virtual nagle enable|disable**

To disable the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
nagle disable
```

To reenable the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
nagle enable
```

- Use the **ssl-server** *number* **tcp server nagle** command to disable or reenable the Nagle algorithm for the TCP connection between the server and the SSL module. The syntax for this command is:

  ssl-server *number* **tcp server nagle enable**|**disable**

  To disable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server nagle
  disable
  ```

  To reenable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

  ```
  (config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server nagle
  enable
  ```

Specifying the TCP Buffering for SSL TCP Connections

If the network is slow and congested, you can increase the buffer size that the CSS buffers for a given TCP connection before shutting down the TCP window to 0. Increasing the buffer size decreases latency on slow connections to a client but increases buffering in the CSS. Use this feature with caution, because if there are many slow clients, the CSS memory may run low.

Use the **ssl-server** *number* **tcp buffer-share** command to set the TCP buffering from the client or server on a given connection. The syntax for this command is:

ssl-server *number* **tcp buffer-share rx** *number1*|**tx** *number2*

- To set the amount of data in bytes that a given connection can buffer from the client traffic, use the **rx** *number1* keyword and variable. By default, the buffer size is 32768. The buffer size can range from 16400 to 262144. For example, to set the value to 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp buffer-share
rx 65536
```

  To reset the reset the buffer size to the default of 32768, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp
buffer-share rx
```

- To set the amount of data in bytes that a given connection can buffer from the server to the client, use the **tx** *number2* keyword and variable. By default, the buffer size is 65536. The buffer size can range from 16400 to 262144. For example, to set the value to 131072, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp buffer-share
tx 131072
```

  To reset the reset the buffer size to the default of 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp
buffer-share tx
```

Activating and Suspending an SSL Proxy List

Before you can activate an SSL proxy list, ensure that you have created at least one virtual or back-end SSL server in the list (see the "Configuring Virtual SSL Servers for an SSL Proxy List" section or the "Specifying the Nagle Algorithm for SSL TCP Connections" section earlier in this chapter).

The CSS checks the SSL proxy list to verify that all of the necessary components are configured, including verification of the certificate and key pair against each other. If the verification fails, the certificate name is not accepted and the CSS logs the error message `Certificate and key pair do not match` and does not activate the SSL proxy list. You must either remove the configured key pair or configure an appropriate certificate.

Use the **active** command to activate the new or modified SSL proxy list. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# active
```

After you activate an SSL proxy list, you can add it to a service. See the "Configuring a Service for SSL Termination" section later in this chapter.

To view the virtual or back-end SSL servers in a list, use the **show ssl-proxy-list** (see Chapter 7, Displaying SSL Configuration Information and Statistics).

Use the **suspend** command to suspend an active SSL proxy list.

To suspend an active SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# suspend
```

Modifying an SSL Proxy List

You cannot modify an SSL proxy list when the list is active. Suspend the list prior to making changes, and then reactivate the SSL proxy list once you complete the changes.

After you modify the proxy list, you do not need to suspend and reactivate the SSL services using the list. The SSL module:

- Sends any additions or changes to any active SSL services using the proxy list.
- Clears the connections for the SSL-related services that were modified or deleted. If the SSL module receives any packets for those connections, the module sends a TCP RST.

Caution If you are using front-end and back-end servers for a flow, both servers must be active for the end-to-end connection to work. If you modify an SSL proxy list , do not delete the back-end server from the list when the service is still active. The end-to-end connection will fail when you reactivate the SSL proxy list.

Configuring a Service for SSL Termination

An SSL proxy list may belong to multiple SSL services (one SSL proxy list per service), and an SSL service may belong to multiple content rules. You can apply the services to content rules that allow the CSS to direct SSL requests for content.

Note The CSS supports one active SSL service for each SSL module in the CSS (one SSL service per slot). You can configure more than one SSL service for a slot but only a single SSL service can be active at a time.

The following sections cover:

- Creating an SSL Service
- Specifying the SSL Acceleration Service Type
- Adding an SSL Proxy List to an SSL Termination Service
- Specifying the SSL Module Slot
- Disabling Keepalive Messages for the SSL Module
- Specifying the SSL Session ID Cache Size
- Activating the SSL Service
- Suspending the SSL Service

Creating an SSL Service

When creating a service for use with an SSL module, you must identify it as an SSL service for the CSS to recognize it. For additional details on creating a service, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Enter the SSL service name, from 1 to 31 characters.

To create service *ssl_serv1*, enter:

```
(config)# service ssl_serv1
Create service <ssl_serv1>, [y/n]: y
```

The CSS transitions into the newly created service mode.

```
(config-service[ssl_serv1])#
```

Specifying the SSL Acceleration Service Type

After you create the SSL service and the CSS enters into service mode, you must specify **ssl-accel** as the service type to:

- Configure the service as an SSL acceleration service.
- Add the SSL proxy list to an SSL service.

Use the **type** command to specify the SSL acceleration service type. For details on specifying an SSL service type, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

To specify the SSL acceleration service type, enter:

```
(config-service[ssl_serv1])# type ssl-accel
```

Adding an SSL Proxy List to an SSL Termination Service

After you configure a virtual SSL server on an SSL proxy list for an SSL module, add the active list to an SSL service. The active list explains how the CSS processes SSL requests for content through the specific SSL module.To include an SSL proxy list as part of an SSL service, use the **add ssl-proxy-list** command in service mode. Enter the name of the previously created SSL proxy list (see the "Creating an SSL Proxy List" section in this chapter) that you want to add to the service.

To add SSL proxy list *ssl_list1* to service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1
```

To remove the SSL proxy list from the service, enter:

```
(config-service[ssl_serv1])# remove ssl-proxy-list ssl_list1
```

Specifying the SSL Module Slot

The CSS 11501 supports a single integrated SSL module. The CSS 11503 and CSS 11506 support multiple SSL modules; a maximum of two in a CSS 11503 and a maximum of four in a CSS 11506. The SSL service requires the SSL module slot number to correlate the SSL proxy list and virtual SSL server(s) to a specific SSL module. Use the **slot** command to specify the slot in the CSS chassis where the SSL module is located.

The valid slot entries are:

- CSS 11501 - 2
- CSS 11503 - 2 and 3
- CSS 11506 - 2 to 6

Slot 1 is reserved for the SCM.

Note The CSS supports one active SSL service for each SSL module in the CSS (one SSL service per slot). You can configure more than one SSL service for a slot but only a single SSL service can be active at a time.

For example, to identify an SSL module in slot 3 of the CSS chassis, enter:

```
(config-service[ssl_serv1])# slot 3
```

Disabling Keepalive Messages for the SSL Module

The SSL module is an integrated device within the CSS chassis and, therefore, does not require the use of keepalive messages for the service. Use the **keepalive type none** command to instruct the CSS not to send keepalive messages to a service. For details on specifying a keepalive type, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

To disable sending keepalive messages for an SSL service, enter:

```
(config-service[ssl_serv1])# keepalive type none
```

Specifying the SSL Session ID Cache Size

The cache size is the maximum number of SSL session IDs that can be stored in a dedicated session cache on an SSL module. By default, the SSL session cache can hold 10000 sessions. If necessary for your SSL service, you can increase the SSL session cache size to 100000. Use the **session-cache-size** command to reconfigure the size of the SSL session ID cache for a service. Valid entries are 0 (SSL session cache disabled) to 100000 sessions.

Note We do not recommend specifying a zero value for the **session-cache-size** command to ensure that the SSL session ID is reused. Specifying an SSL session cache and cache timeout allows the reuse of the master key on subsequent connections between the client and the CSS SSL module, which can speed up the SSL negotiation process and improve CSS performance.

The back-end session ID cache is 4096 entries and is not configurable.

If you specify 0 as the SSL session cache size, the SSL module associated with the SSL service does not cache any SSL session IDs. If you choose to disable the SSL session cache, ensure the following parameters are properly configured to disable the use of SSL session ID:

- Set the **ssl-server** *number* **session-cache timeout** setting in the SSL proxy list to 0 (disabled) for a virtual SSL server.
- Disable the **advanced-balance ssl** command in the content rule to disable SSL sticky.

For example, to specify an SSL session cache size of 20000 sessions, enter:

```
(config-service[ssl_serv1])# session-cache-size 20000
```

To reset the SSL session cache size to the default of 10000 sessions, enter:

```
(config-service[ssl_serv1])# no session-cache-size
```

Activating the SSL Service

Once you configure an SSL proxy list service, use the **active** command to activate the service. Activating a service puts it into the resource pool for load-balancing SSL content requests between the client and the server.

Before activating an SSL service:

- For a virtual SSL server, you must add an SSL proxy list to an **ssl-accel** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the following error message and does not activate the service.

  ```
  Must add at least one ssl-proxy-list to an ssl-accel type service
  ```

- For a back-end SSL server, you must add an SSL proxy list to an **ssl-accel-backend** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the following error message and does not activate the service.

  ```
  Must add at least one ssl-proxy-list to an ssl-accel type service
  ```

- The SSL proxy list added to the service must be active before you can activate the service. If the list is suspended, the CSS logs the following error message and does not activate the service.

  ```
  No ssl-lists on service, service not activated
  ```

Once the service is ready to activate, the CSS initiates the transfer of appropriate SSL configuration data for each SSL proxy list to a specific SSL module and activates the service. If there is an error in transfer, the CSS logs the appropriate error and does not activate the service.

To activate service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# active
```

Suspending the SSL Service

To suspend an SSL service and remove it from the pool for future load-balancing SSL content requests, use the **suspend** command. Suspending an SSL service does not affect existing content flows, but it prevents additional connections from accessing the service for its content.

To suspend service ssl_*serv1*, enter:

```
(config-service[ssl_serv1])# suspend
```

Configuring a Content Rule for SSL Termination

For the CSS to direct SSL requests for content, apply the virtual services to content rules. No network traffic is sent to an SSL module until you activate an SSL content rule to define where the content physically resides, where to direct the request for content (which SSL service), and which load-balancing method to use.

For a virtual SSL server content rule, ensure that the VIP address and port number configured for the rule match the VIP address and port number for the server entry in the SSL proxy list.

When you activate a content rule with a configured SSL service, the CSS verifies that there is a VIP address and port match. If a match is not found, the CSS logs the following error message and does not activate the content rule.

```
Not all content VIP:Port combinations are configured in an
ssl-proxy-list for sslAccel type of service
```

Verify the configured VIP addresses used in the content rule and SSL proxy list, and modify as necessary.

When a CSS uses two or more SSL modules, we recommend that you use stickiness based on SSL version 3 session ID for a Layer 5 content rule. For a virtual SSL server rule, specify the following:

- Enable the content rule to be sticky based on SSL using the **advanced-balance ssl** command.
- Specify the SSL application type using the **application ssl** command.

The Layer 5 SSL sticky content rule ensures SSL session ID reuse to eliminate the rehandshake process (which speeds up the SSL negotiation process) and to increase overall performance.

Note If the 32K sticky table becomes full (which means that 32K simultaneous users are on the site) the table wraps and the first users in the table become "unstuck." This may be due to a combination of number of flows and the duration of the sticky period, which can quickly use up the available space in the sticky table. This problem can typically occur in a CSS that contains multiple SSL modules. An SCM with 288M memory module can support a 128K sticky table.

Note If you specify the **sticky-inact-timeout** command for a Layer 5 content rule using SSL sticky, the SSL sessions continue even if the sticky table is full. However, the CSS does not maintain stickiness on the new sessions.