Configuring SSL Initiation

This chapter describes the steps required to configure a CSS to initiate an SSL connection with an SSL server after receiving clear text from a client. It contains the following major sections:

- Overview of SSL Initiation
- Creating an SSL Initiation Proxy List
- Adding a Description to an SSL Initiation Proxy List
- Configuring Back-End SSL Servers in an SSL Initiation Proxy List
- Activating and Suspending an SSL Proxy List
- Modifying an SSL Proxy List
- Configuring a Service for SSL Initiation
- Configuring a Content Rule for SSL Initiation
- Troubleshooting SSL Initiation

Overview of SSL Initiation

SSL initiation allows a CSS configured with an SSL module to:

- Receive clear text from a client
- Load balance the content
- Encrypt the clear text

- Originate an SSL connection with either an SSL server or another CSS configured with SSL termination (see Chapter 4, Configuring SSL Termination).

Use this feature for secure site-to-site data transfers. SSL initiation allows you to send clear text within a site for maximum speed, while sending encrypted text through the Internet between sites or to an SSL server for maximum security. For each SSL server or CSS to which you want to establish an SSL connection from a clear-text connection, you must configure an SSL initiation service on the CSS that maps to that SSL server or CSS. This service uses an SSL proxy list to properly direct the flows within the CSS.

Figure 6-1 illustrates a single SSL initiation flow with an SSL server.

Figure 6-1 SSL Initiation with an SSL Server

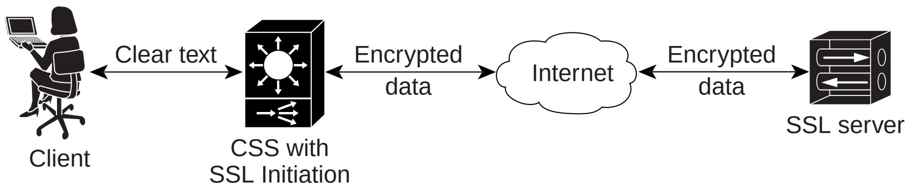
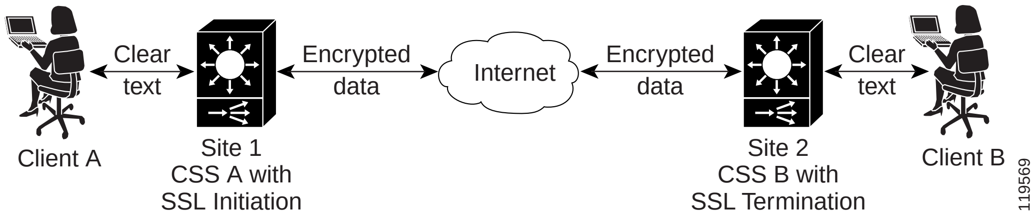


Figure 6-2 illustrates an SSL initiation flow with another CSS configured with SSL termination. In this case, the second CSS acts as a virtual front-end SSL server.

Figure 6-2 SSL Initiation with a Second CSS Running SSL Termination



An SSL proxy list determines the flow of SSL information among the client, SSL module, and the SSL server. An SSL proxy list comprises one or more back-end SSL servers (virtual servers that you create on the CSS SSL module) related by index entry. An SSL module in the CSS uses the back-end SSL server to initiate the connection to an SSL server. You can define a maximum of 256 back-end SSL servers in 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 an initiation service to enable the transfer of SSL configuration data to the SSL module. When you activate the service, the CSS transfers the configuration data to the module. You can then add each SSL initiation service to an SSL content rule.

Creating an SSL Initiation Proxy List

An SSL initiation proxy list is a group of related back-end SSL servers that are associated with an SSL initiation 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 Initiation Proxy List

To specify a description for an SSL initiation 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 proxy list, ssl_list1, 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 Back-End SSL Servers in an SSL Initiation Proxy List

This section discusses the creation of one or more back-end SSL servers in an SSL proxy list for use with the SSL initiation feature. In SSL proxy-list mode, use the **backend-server** command to create an SSL initiation back-end server on the CSS and define an index entry in the SSL proxy list. You then use this index to configure specific back-end SSL parameters associated with the SSL proxy list. An SSL module in the CSS uses the back-end SSL server defined in the SSL proxy list to initiate a connection to an SSL server. You must define a back-end SSL server index number before configuring SSL proxy-list parameters. You can define a maximum of 256 back-end SSL initiation servers in a single SSL proxy list.

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.

Once you create a back-end server in an SSL proxy list, configure a IP address that corresponds to the address of the service and a server IP address that corresponds to the IP address of the SSL initiation server. Configure the other optional proxy-list parameters if desired, and then activate the SSL proxy list. To make the back-end server work for SSL initiation, you must configure the back-end server type as **initiation**.

After you configure and activate the SSL proxy list, add the list to an SSL initiation service. When you activate the service, the CSS sends the configuration data to the SSL module.

The following sections describe:

- Creating a Back-End Server in an SSL Initiation Proxy List
- Configuring the Back-End Server as an SSL Initiation Server
- Configuring an IP Address for the SSL Initiation Server
- Configuring a Port for the SSL Initiation Server
- Configuring the SSL Server IP Address
- Configuring the SSL Server Port
- Configuring SSL Version
- Configuring the Available Cipher Suites
- Configuring SSL Session Cache Timeout
- Configuring SSL Session Handshake Renegotiation
- Configuring TCP Virtual Client Connections Timeout Values
- Configuring TCP Server-Side Connection Timeout Values on the SSL Module
- Specifying the TCP Buffering for SSL TCP Connections
- Configuring the Retransmission Timer for SSL TCP Connections
- Changing the Acknowledgement Delay for SSL TCP Connections
- Configuring Client Certificates and Keys
- Configuring CA Certificates for Server Authentication

Creating a Back-End Server in an SSL Initiation Proxy List

Before you can configure SSL initiation proxy list parameters, you must create a back-end server in an SSL proxy list. To create a back-end server in the SSL proxy list, use the **backend-server** *number* command. By default, this command creates a back-end server of type **backend-ssl** and assigns it a number (index entry you enter) in the SSL proxy list that you use to configure specific SSL parameters associated with the back-end SSL server (for example, IP address, certificate name, and key pair). Enter a value from 1 to 256.

For example, to create back-end server 1 in the proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1
```

To remove back-end server 1 and all its configured parameters from proxy list ssl_list1, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1
```

Configuring the Back-End Server as an SSL Initiation Server

To use the back-end server for SSL initiation, you must configure it as an initiation server for use with services of type **ssl-init**. By default, the back-end server is a server of type **backend-ssl** for use with services of type **ssl-accel-backend**.

To configure the back-end server as an initiation server (accepts clear traffic from a client and initiates SSL traffic with the SSL server) for use with services of type **ssl-init**, use the **backend-server** *number* **type** command. The syntax for this command is:

backend-server *number* **type** {**backend-ssl**|**initiation**}

For example, to configure back-end server 1 as an SSL initiation server in proxy list ssl_list1, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 type initiation
```

To reconfigure the SSL initiation server as a back-end SSL server without having to configure all the back-end server parameters, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 type backend-ssl
```

Configuring an IP Address for the SSL Initiation Server

To configure an IP address for the SSL initiation server, use the **backend-server** *number* **ip address** command. The IP address corresponds to the IP address of the SSL initiation service.

For example, to configure the IP address 192.168.2.3 for SSL initiation back-end server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 ip address
192.168.2.3
```

To remove the IP address from SSL initiation server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 ip address
```

Note If you have not configured the SSL initiation back-end server IP address when you issue the **active** command, the following error message appears and the CSS does not activate the list.

```
%% Error in backend-server 1: SSL-server/Backend-server must have valid
IP address
```

Configuring a Port for the SSL Initiation Server

By default, the port for the SSL initiation back-end server is port 80. This port accepts the clear text data traffic from the SSL initiation service and sends it to the SSL module. To configure a different port for the SSL initiation server, use the **backend-server** *number* **port** command. Enter a port number from 1 to 65535.

Note If you configure the **backend-server** *number* **ip address** and **server-ip** commands with the same address, configure the **backend-server** *number* **port** and **server-port** commands with different port numbers.

For example, to configure a port number of 1200, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 port 1200
```

To reset the port to the default value of 80, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 port
```

Configuring the SSL Server IP Address

The server IP address is the IP address for the real SSL server. To configure the IP address for the real SSL server, use the **backend-server** *number* **server-ip** command.

For example, to configure the server IP address 192.168.2.3, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-ip
192.168.2.3
```

To remove the real server IP address from the proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 server-ip
```

Note If you have not configured the real server IP address when you issue the **active** command, the following error message appears and the CSS does not activate the list.

```
%% Error in backend-server 1: SSL-server/Backend-server must have valid
IP address
```

Configuring the SSL Server Port

By default the port number for the real SSL server is port 443. To configure a different server port for the SSL server, use the **backend-server** *number* **server-port** command. Enter a port number from 1 to 65535.

Note If you configure the **backend-server** *number* **ip address** and **server-ip** commands with the same address, configure the **backend-server** *number* **port** and **server-port** commands with different port numbers.

For example, to configure the server port number 155, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-port 155
```

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

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 server-port
```

Configuring SSL Version

The SSL module initiates the connection to the real SSL server. The version in the ClientHello message sent to the server indicates the highest supported 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 **backend-server** *number* **version** command to specify which version of SSL the back-end server supports:

- **ssl3** - SSL version 3.
- **tls1**- TLS version 1.
- **ssl-tls** - 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.

For example, to configure the SSL version 3, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 version ssl3
```

To reset the default SSL version, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 version
```

Configuring the Available Cipher Suites

To configure one or more specific cipher suites to be used by the back-end SSL initiation server, use the **backend-server** *number* **cipher** command. By default, all supported hardware accelerated cipher suites are enabled.

For a list of all cipher suites that the SSL module supports and the corresponding cipher suite values, see Table 4-1 in the "Specifying Cipher Suites" section in Chapter 4, Configuring SSL Termination. These values match those defined for SSL version 3.0 and TLS version 1.0. Table 4-1 also lists those Cipher suites that are exportable in any version of the software.

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** option reenables all cipher suites for the back-end server. This option works only when you do not configure specifically-defined ciphers. To return to using the **all-cipher-suites** option, you must remove all specifically-defined ciphers.

For example, to configure a cipher of rsa-with-rc4-128-md5, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher
rsa-with-rc4-128-md5
```

When negotiating which cipher suite to use, the SSL module sends the ciphers in weighted order to the server with the highest weighted cipher first in the list.

By default, all configured cipher suites have a weight of 1. Optionally, you can assign a priority weight to the cipher suite, with 10 being the highest.

Note If two or more ciphers have the same weight (no weight has a value of 1), the ciphers appear in the Client Hello in the same order as they appear in the running-configuration file.

For example, to set a weight of 10 to a cipher suite, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher
rsa-with-rc4-128-md5 weight 10
```

To remove one or more of the configured cipher suites for the SSL initiation back-end server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 cipher
rsa-with-rc4-128-md5
```

Configuring SSL Session Cache Timeout

In SSL, every time a client and server go through a full key exchange and establish a new master secret key, a new session is created. Enabling a session cache timeout allows the reuse of the master key on subsequent connections by the client. When you disable the cache timeout, the full SSL handshake must occur on each new connection to the SSL module (the virtual client). Use the **backend-server** *number* **session-cache** command to configure the SSL module to resume connection with a back-end SSL server using a previously established secret key.

By default, the cache timeout is enabled with a timeout of 300 seconds (5 minutes). The timeout value can range from 0 to 72000 (0 seconds to 20 hours). A timeout value of 0 disables the session cache reuse.

For example, to configure the SSL session cache timeout of 500 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 session-cache 500
```

To reset the session cache ID reuse to the default of enabled with a timeout of 300 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 session-cache
```

To disable session cache ID reuse, enter a timeout value of 0 seconds:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 session-cache 0
```

Configuring 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 between the CSS and the back-end SSL server.

Use the **backend-server** *number* **handshake data** *kbytes* command to force an SSL rehandshake after the exchange of a certain amount of data between the CSS and the back-end SSL server, after which the CSS transmits the SSL handshake message and reestablishes the SSL session.

By default, the SSL rehandshake based on data (flow) is disabled (set to 0) for a back-end SSL server after the exchange of data. The data value is in kilobytes and is from 0 to 512000 kilobytes.

For example, to configure the SSL session rehandshake data value of 500 Kbytes, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 handshake data
500
```

To reset the rehandshake data value to 0, disable the rehandshake based on the exchange of data. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 handshake data
```

Use the **backend-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.

By default, the SSL rehandshake timeout is disabled (set to 0) for the back-end SSL server. The timeout value is from 0 to 72000 (0 seconds to 20 hours).

For example, to configure a 30-second timeout of an SSL session rehandshake, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 handshake timeout
30
```

To reset the timeout to 0, disable the rehandshake timeout period for the back-end server by entering:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 handshake
timeout
```

Configuring TCP Virtual Client Connections Timeout Values

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

To configure the parameters for the TCP connection with the client, see the following sections:

- Specifying a TCP SYN Timeout Value for the Virtual Client Connection
- Specifying a TCP Inactivity Timeout for a Virtual Client Connection
- Specifying the Nagle Algorithm for Client-Side Connections

Specifying a TCP SYN Timeout Value for the Virtual Client Connection

The CSS SYN timer counts the time difference 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 **backend-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 and the SSL module 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 TCP connections.

To configure the TCP SYN timeout of 100 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual
syn-timeout 100
```

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

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp virtual
syn-timeout
```

Specifying a TCP Inactivity Timeout for a Virtual Client 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 **backend-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 and the SSL module 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.

Based on the default parameters for retransmission, the timer value should be larger than 60 seconds (1 minute).

For example, to configure the TCP inactivity timeout period of 100 seconds for the virtual client connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual
inactivity-timeout 0
```

To reset the timeout to the default value of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp virtual
inactivity-timeout
```

Specifying the Nagle Algorithm for Client-Side 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. 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 **backend-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:

backend-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])# backend-server 1 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])# backend-server 1 tcp virtual nagle
enable
```

Configuring TCP Server-Side Connection Timeout Values on the SSL Module

The TCP connection between the SSL module 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 the timeout values of a TCP connection with the server, see the following sections:

- Specifying a TCP SYN Timeout Value for a Server-Side Connection
- Specifying a TCP Inactivity Timeout for a Server-Side Connection
- Specifying the Nagle Algorithm for Server-Side Connections

Specifying a TCP SYN Timeout Value for a Server-Side Connection

The TCP SYN timer counts the time difference between the CSS initiating the back-end TCP connection by transmitting a SYN and the server replying with a SYN/ACK. Use the **backend-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 TCP connections.

For example, to configure the TCP SYN timeout of 100 seconds for the server-side connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server
syn-timeout 100
```

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

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp server
syn-timeout
```

Specifying a TCP Inactivity Timeout for a 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 **backend-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 the TCP inactivity timeout period of 100 seconds for the server-side connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server
inactivity-timeout 0
```

To reset the timeout to the default value of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp server
inactivity-timeout
```

Specifying the Nagle Algorithm for Server-Side Connections

The TCP Nagle algorithm automatically concatenates a number of small buffer messages transmitted over the TCP connection between a back-end 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 **backend-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:

backend-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])# backend-server 1 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])# backend-server 1 tcp server nagle
enable
```

Specifying the TCP Buffering for SSL TCP Connections

If you experience unacceptable latency in your network due to congestion, you can increase the buffer size that the CSS buffers for a given TCP connection before shutting down the TCP window to 0. Use the **backend-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:

backend-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])# backend-server 1 tcp buffer-share
rx 65536
```

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

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 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])# backend-server 1 tcp buffer-share
tx 131072
```

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

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp
buffer-share tx
```

Configuring the Retransmission Timer for SSL TCP Connections

On networks that experience a lot of packet loss, TCP transaction can take a long time. To adjust the retransmission timer for TCP transactions, use the following command:

backend-server *server-num* **tcp virtual|server retrans** *milliseconds*

The *milliseconds* variable is the minimum time in milliseconds for retransmission of TCP transactions. Enter a number form 50 to 500. The default value is 500.

For example, to set the retransmission timer to 100 milliseconds for client transactions, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 20 tcp virtual
retrans 100
```

To set the retransmission timer to 100 milliseconds for server transactions, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 20 tcp server
retrans 100
```

To reset the default value of 500 milliseconds for client transactions, enter:

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

To reset the default value of 500 milliseconds for server transactions, enter:

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

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:

backend-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])# backend-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])# backend-server 20 tcp server
ack-delay 400
```

Configuring Client Certificates and Keys

SSL servers frequently require that a client authenticate itself before a data transfer can occur. To allow the client (in this case, the SSL module) to authenticate itself to such a server, you must configure client certificates and keys on the CSS.

To obtain a client certificate and key pair, contact your authorized certificate authority (CA). Once the CA has prepared your client certificate and key pair, you must import them into the CSS. For information about importing a certificate and key pair, see the "Importing or Exporting Certificates and Private Keys" section in Chapter 3, Configuring SSL Certificates and Keys. Once you have imported the certificate and key pair, you must associate them with a filename. For information about associating a certificate and key pair with filenames, see the "Associating Certificate and Private Key Files with Names" section in Chapter 3, Configuring SSL Certificates and Keys.

If the SSL module originates a connection to an SSL server that requests a client certificate and no client certificate and key are configured on the CSS, the CSS increments the Requested Client Certificate Not Sent counter.

Note When the SSL server does not receive the requested client certificate, it may close the connection.

The following sections describe how to configure client certificates and keys.

Configuring the RSA Certificate Name

To configure the back-end server RSA certificate, use the **backend-server** *number* **rsacert** *name* command. The certificate must already be loaded on the SCM. If the certificate name does not exist, the CSS logs an error message. Enter a name for the RSA certificate as an unquoted text string from 1 to 31 characters.

For example, to configure an RSA certificate named myrsacert, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 rsacert myrsacert
```

To remove an RSA cert from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 rsacert
```

Configuring the RSA Key Name

To configure the back-end server RSA key name, use the **backend-server** *number* **rsakey** *name* command. The key pair must already be loaded on the SCM. If the key pair name does not exist, the CSS logs an error message. Enter a name for the RSA key pair as an unquoted text string from 1 to 31 characters.

For example, to configure an RSA key pair named myrsakey, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 rsakey myrsakey
```

To remove an RSA key pair from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 rsakey
```

Configuring Diffie Hellman Parameters

To configure the back-end server Diffie-Hellman (DH) parameter file, use the **backend-server** *number* **dhparam** *name* command. The DH parameters file must already be loaded on the SCM. If the parameter file does not exist, the CSS logs an error message. Enter a name for the DH parameter files as an unquoted text string from 1 to 31 characters.

For example, to configure a DH parameter file named dhparamfile2, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 dhparam
dhparamfile2
```

To remove the configured DH parameter file from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 dhparam
```

Configuring the DSA Certificate Name

To configure the back-end server DSA certificate, use the **backend-server** *number* **dsacert** *name* command. The certificate must already be loaded on the SCM. If the certificate name does not exist, the CSS logs an error message. Enter a name for the DSA certificate as an unquoted text string from 1 to 31 characters.

For example, to configure a DSA certificate named mydsacert, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 dsacert mydsacert
```

To remove a DSA cert from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 dsacert
```

Configuring the DSA Key Filename

To configure the back-end server DSA key name, use the **backend-server** *number* **dsakey** *name* command. The key pair must already be loaded on the SCM. If the key pair name does not exist, the CSS logs an error message. Enter a name for the DSA key pair as an unquoted text string from 1 to 31 characters.

For example, to configure a DSA key pair named mydsakey, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 dsakey mydsakey
```

To remove an DSA key pair from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 dsakey
```

Configuring CA Certificates for Server Authentication

If the it has the public key of a particular certificate authority (CA), the CSS can verify that the server certificate was signed by that CA. The CSS obtains the public key of the CA from the CA certificate. If you configure a CA certificate name in an SSL initiation proxy list, the CSS can use the public key in the certificate to verify the digital signature of the CA in the server certificate. Defining a CA certificate in the SSL initiation proxy list indicates to the CSS that you want to verify the server certificate.

Note By default, SSL servers are not authenticated.

Before you configure the CA certificate in an SSL initiation proxy list, you must import the certificate on the CSS and then associate the certificate with a filename. For information about importing a CA certificate, see the "Importing or Exporting Certificates and Private Keys" section in Chapter 3, Configuring SSL Certificates and Keys. For information about associating a certificate with a filename, see the "Associating Certificate and Private Key Files with Names" section in Chapter 3, Configuring SSL Certificates and Keys.

To enable the SSL module (the client) to authenticate the SSL server, you must configure at least one, with a maximum of four, CA certificates in the SSL initiation proxy list. If you attempt to configure more than four CA certificates, the CSS displays the following error message:

```
%% Max number of CA Certificates configured on server.
```

Use the **cacert** command to configure the CA certificates in the proxy list. The syntax for this command is:

backend-server *number* **cacert** {*name*}

The *name* variable specifies the filename with which you have previously associated the CA certificate. Enter a filename from 1 to 31 characters. The CA certificate must already be loaded on the SCM. You can define a maximum of four CA certificates for each SSL initiation proxy list. The CSS uses the CA certificates to verify the server certificate in the order in which you configure the CA certificates.

For example, to configure the mycert1 CA certificate in proxy list ssl_list1 for SSL initiation server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cacert mycert1
```

To remove a CA certificate from an SSL proxy list, use the **no** form of the command. For example, to remove the mycert1 CA certificate from the ssl_list1 proxy list for SSL initiation back-end server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 cacert mycert1
```

Activating and Suspending an SSL Proxy List

Before you can activate an SSL proxy list, ensure that you have created at least one back-end SSL server configured as type **initiation** in the list. See the "Configuring Back-End SSL Servers in an SSL Initiation Proxy List" section.

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 a valid 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 Initiation" section.

To display the back-end SSL servers configured in a proxy list, use the **show ssl-proxy-list** command (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 Initiation

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, which allow the CSS to direct clear content requests to the SSL module for encryption.

The requirements for the type of service to be added to the SSL initiation content rule is as follows:

- The service must have a configured IP address.
- The keepalive type for an SSL initiation service can be none, ICMP, TCP, named, scripted, or SSL. You can also configure an encrypted persistent or non-persistent HTTP keepalive to verify the full SSL handshake and data returned from the server.

  If you configure a TCP or SSL keepalive type, you must configure the keepalive port correctly for the service to work.

- You must configure an SSL proxy list that contains an SSL initiation back-end server configuration for a service of type **ssl-init**.

Note The CSS supports multiple active SSL services of type **ssl-init** for each SSL module in the CSS.

This section covers:

- Creating an SSL Service
- Configuring the SSL Service Type
- Configuring an IP Address for an SSL Initiation Service
- Adding an SSL Proxy List to an SSL Initiation Service
- Specifying the SSL Module Slot
- Configuring the SSL Initiation Service Keepalive Type
- SSL Session ID Cache Size
- Activating the SSL Service
- Suspending the SSL Service

Note If you do not configure a service port, the CSS uses the same port number as the content rule.

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. You can create multiple SSL services for use with an SSL initiation content rule. 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 to the service mode.

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

Configuring the SSL Service Type

You must configure the **ssl-init** service type for an SSL initiation service. To configure a service type for an SSL initiation service, enter:

```
(config-service[server1]# type ssl-init
```

Configuring an IP Address for an SSL Initiation Service

The IP address for an SSL initiation service must match the IP address configured in the SSL proxy list for the SSL initiation back-end server (the **backend server-ip** address).

For example, to configure the IP address 192.168.21.7 for the SSL initiation service, enter:

```
(config-service[server1])# ip address 192.168.21.7
```

To remove the IP address for the SSL initiation service, enter:

```
(config-service[server1])# no ip address
```

Adding an SSL Proxy List to an SSL Initiation Service

After you configure an SSL proxy list for an SSL initiation server, add the active list to one or more SSL initiation services to define how the CSS processes SSL requests for content from an SSL initiation back-end server. Configuring the SSL initiation service is similar to configuring a local service except that you must set the service type to **ssl-init**. Also, an SSL initiation service requires an SSL proxy list with a back-end server entry.

An SSL proxy list contains the parameters for the SSL initiation service. To add the proxy list to the service, use the **add ssl-proxy-list** command. For more information on configuring an SSL proxy list for SSL initiation, see the "Creating an SSL Initiation Proxy List" section.

Enter the name of the previously created SSL proxy list (see the "Creating an SSL Initiation Proxy List" section) that you want to add to the service.

For example, to add the SSL proxy list *ssl_list1* for an SSL initiation service, enter:

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

To remove an SSL proxy list for the SSL initiation 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 multiple active SSL services of type **ssl-init** for each SSL module in the CSS.

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

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

Configuring the SSL Initiation Service Keepalive Type

A service of type **ssl-init** supports the use of keepalives to periodically check the health of the SSL server. The CSS sends the keepalives to the IP address configured on the service.

To configure a keepalive, use the **keepalive type** command in service configuration mode. The syntax for this service configuration mode command is:

(config-service[server1])# **keepalive type** *type*

For the *type* variable, enter one of the following keepalive types:

- **icmp** - An ICMP echo message (ping). This is the default keepalive type.
- **none** - Do not send keepalive messages to this service.
- **ssl** - SSL HELLO keepalives for this service. The CSS sends a client HELLO to connect the SSL server. After the CSS receives a HELLO from the server, the CSS closes the connection with a TCP RST.
- **tcp** - A TCP session that determines service viability through a 3-way handshake and reset; SYN, SYN-ACK, ACK, RST-ACK.
- **http** {**non-persistent**} **encrypt** - Encrypted persistent or non-persistent HTTP keepalive to verify the full SSL handshake and data returned from the server.

Note If you configure a TCP, SSL, or encrypted keepalive, you must configure the keepalive port correctly for the service to work.

For more information about ICMP, SSL, TCP, and other CSS keepalives, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*. For information about encrypted HTTP keepalives, see the following section.

Configuring Encrypted HTTP Keepalives

Encrypted HTTP keepalives allow the verification of the full SSL handshake and the data returned from the server. The keepalives perform an HTTP GET or HEAD that is directed to the SSL module. Then, the module connects to the configured server.

For an SSL initiation service, encrypted keepalives use the SSL module in the slot configured for the service. For information on configuring the SSL module slot, see the "Specifying the SSL Module Slot" section.

The configured keepalive IP address and port must be the same as an SSL initiation server in the SSL proxy list.

Note An SSL proxy list contains a maximum of 256 SSL back-end or initiation servers. Therefore, the total number of encrypted keepalives on the CSS can only be 256.

Note When configuring an encrypted HTTP keepalive, make sure that the configured IP address for the back-end server and the real server are the same.

You can configure a service or global encrypted keepalive. The syntax for the service configuration mode command is:

(config-service)# **keepalive type http** {**non-persistent**} **encrypt**

For example, to configure a service encrypted HTTP HEAD non-persistent keepalive for an SSL initiation service, enter:

```
(config-service [ssl_serv1])# keepalive http non-persistent encrypt
```

The syntax for the global encrypted keepalive is:

(config-keepalive)# **type http** {**non-persistent**} **encrypt**

For global keepalives, no service or slot information is available to the keepalive. For a global encrypted keepalive to select an SSL module, the keepalive uses the slot of the SSL initiation service.

Note In order for encypted global keepalives to work properly, you should only add it to services that use the same SSL proxy list.

For example, to configure a service encrypted HTTP HEAD non-persistent keepalive for an SSL initiation service, enter:

```
(config-keepalive [SSL1])# http non-persistent encrypt
```

Then assign the global keepalive to the service for the SSL initiation service.

If you are configuring a SSLinitiation server to support session ID reuse and you want to use encrypted keepalives to perform a full SSL handshake, you must configure an additional SSL initiation server. This second server would have the same configuration as the original SSL initiation server except it would be configured with a different port number and a session cache timeout value. Then you can configure an encrypted keepalive with the port number of the second server. The encrypted keepalives are sent to this server on the SSL module. The module connects to the SSL back-end server and the keepalive performs a full SSL handshake.

Encrypted Keepalive Configuration Examples

The following configuration shows an encrypted HTTP non-persistent keepalive for an SSL initiation server. The keepalives with the destination address 192.168.7.10 and port 80 are sent to SSL module 3.

```
!********************** SSL PROXY LIST ***********************
ssl-proxy-list SSLInit_list
  backend-server 1
  backend-server 1 ip address 192.168.7.10
  backend-server 1 server-ip 192.168.7.10
  backend-server 1 type initiation
  active

!************************* SERVICE *************************

service DC-SSL1
  ip address 192.168.7.10
  port 80
  type ssl-init
  slot 2
  keepalive type http non-persistent encrypt
  keepalive method head
  keepalive uri "/index.html"
  active
```

The following configuration shows a global encrypted HTTP non-persistent keepalive for an SSL initiation server. The global keepalives with the destination address 192.168.7.10 and port 80 are sent to SSL module either 2 or 3. If that server goes down, all of the servers associated with the keepalive also go down.

```
!************************** KEEPALIVE ************************
keepalive SSL1
  IP address 192.168.7.10
  type http non-persistent encrypt
  method head
  uri "/index.html"
  active

!*********************** SSL PROXY LIST ***********************
ssl-proxy-list SSLInit_list
  backend-server 1
  backend-server 1 ip address 192.168.7.10
  backend-server 1 server-ip 192.168.7.10
  backend-server 1 type initiation
  backend-server 2
  backend-server 2 ip address 192.168.7.20
  backend-server 2 server-ip 192.168.7.20
  backend-server 2 type initiation
  backend-server 3
  backend-server 3 ip address 192.168.7.30
  backend-server 3 server-ip 192.168.7.30
  backend-server 3 type initiation
  backend-server 4
  backend-server 4 ip address 192.168.7.40
  backend-server 4 server-ip 192.168.7.40
  backend-server 4 type initiation
  active

!************************** SERVICE **************************

service DC1
  type ssl-init
  ip address 192.168.7.10
  protocol tcp
  port 80
  slot 2
  keepalive named SSL1
active

service DC2
```

```
  type ssl-init
  ip address 192.168.7.20
  protocol tcp
  port 80
  slot 3
  keepalive named SSL1
  active

service DC3
  type ssl-init
  ip address 192.168.7.30
  protocol tcp
  port 80
  slot 2
  keepalive named SSL1
  active

service DC4
  type ssl-init
  ip address 192.168.7.40
  protocol tcp
  port 80
  slot 3
  keepalive named SSL1
  active
```

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. For services of type **ssl-init**, the SSL session cache size is fixed at 4096 entries and is not configurable.

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 an initiation SSL server, you must add an SSL proxy list to an **ssl-init** 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-init 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 Initiation

For the CSS to encrypt clear client requests for content, apply the SSL initiation services to content rules. A content rule defines:

- Where the content physically resides
- Where to direct the request for content (which SSL initiation services)
- Which load-balancing method to use

For an HTTP server or back-end SSL server content rule, ensure that each configured service IP address matches an IP address configured for an SSL initiation server in the SSL proxy list (see the "Configuring an IP Address for the SSL Initiation Server" section).

For an SSL initiation content rule, you can specify a Layer 5 cookie or URL rule. The information in the rule enables the CSS to locate a sticky server to use or to load balance a new server for a new client request.

For more information on Layer 5 sticky and content rules, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Troubleshooting SSL Initiation

The following information is designed to assist you in troubleshooting issues that you may encounter when configuring SSL initiation.

For issues with the SSL proxy list:

- Verify that you have configured the back-end server as type initiation. See the "Configuring the Back-End Server as an SSL Initiation Server" section.
- Verify that you have added the SSL proxy list to a service of type **ssl-init** and you have activated the service. See the "Configuring a Service for SSL Initiation" section.
- Verify that you have added the SSL service to a content rule and you have activated the content rule. See the "Configuring a Content Rule for SSL Initiation" section.

For issues with client certificates:

- Verify that you have configured the client certificate and key on the appropriate back-end server in the SSL proxy list. See the "Configuring Client Certificates and Keys" section.
- Verify that you have added the SSL proxy list to a service of the type for which the back-end server will be used. Use the **type ssl-init** command for SSL initiation and the **type ssl-accel-backend** command for back-end SSL. See the "Configuring a Service for SSL Initiation" section.
- Verify that you have added the SSL service to a content rule and that the content rule is active. See the "Configuring a Content Rule for SSL Initiation" section.
- Ensure that the SSL server is configured to request a client certificate.
- Use a sniffer on the back-end connection to verify that the server is requesting a client certificate and that the CSS is sending the certificate.