Configuring the CSM-S SSL Services

This chapter describes the Command Line Interface (CLI) commands to configure, monitor, and debug the CSM-S software for SSL. These configuration commands are the same commands that are valid in the SSL Services Module.

Note Except where specifically differentiated, the term *Content Switching Module* and its acronym *CSM* refer to both the Content Switching Module and the Content Switching Module with SSL. The term *Content Switching Module with SSL* and its acronym *CSM-S* are used to refer to the CSM-S only.

This chapter describes configuration additions made to the CSM-S to support the SSL daughter card and contains these sections:

Note You must create a separate server farm for all back-end servers. Although the CSM-S is not associated with a virtual server, the CSM-S performs address resolution on each real server. A sample configuration is shown in the "Configuring the Real Server as the Back-End Server" section on page 1-17.

Initial SSL Daughter Card Configuration

This section describes how to make the initial configurations for the SSL daughter card.

Note You must make the following initial SSL daughter card configurations through a direct connection to the CSM-S Certificate Management Port connector (see Figure 1-2 on page 1-7). After the initial configurations, you can make an SSH or Telnet connection to the module in order to make further configurations for the module.

The initial SSL daughter card configuration consists of these tasks:

Configuring VLANs on the SSL Daughter Card

When you configure VLANs on the SSL daughter card, configure one of the VLANs as an administrative VLAN. The administrative VLAN is used for all management traffic, including SSH, public key infrastructure (PKI), secure file transfer (SCP), and TFTP operations. The system adds the default route through the gateway of the administrative VLAN.

Note Configure only one VLAN on the SSL daughter card as the administrative VLAN.

Note All VLANs configured on the SSL daughter card must also be configured on the CSM. All VLANs must match for the CSM virtual servers and the SSL real servers.

Note The VLAN IDs for the switch and the module must be identical. Refer to the “Configuring VLANs” chapter in the *Catalyst 6500 Series Switch Software Configuration Guide* for details.

Note The SSL software supports only the normal-range VLANs (2 through 1005). You must limit the SSL daughter card configuration to the normal-range VLANs.

To configure VLANs on the SSL daughter card, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy vlan vlan`	Configures the VLANs and enters VLAN mode.
Step 2	`ssl-proxy(config-vlan)# ipaddr ip_addr netmask`	Configures an IP address for the VLAN.

	Command	Purpose
Step 3	`ssl-proxy(config-vlan)# gateway gateway_addr`	Configures the client-side gateway IP address. **Note** Configure the gateway IP address in the same subnet as the VLAN IP address.
Step 4	`ssl-proxy(config-vlan)# route ip_addr netmask gateway ip_addr`	(Optional) Configures a static route for servers that are one or more Layer 3 hops away from the CSM-S.
Step 5	`ssl-proxy(config-vlan)# admin`	(Optional) Configures the VLAN as the administrative VLAN[1].

1. The administrative VLAN is for management traffic (PKI, SSH, SCP and TFTP). Specify only one VLAN as the administrative VLAN.

This example shows how to configure the VLAN and specify the IP address, the subnet mask, and the global gateway, and how to specify the VLAN as the administrative VLAN:

```
ssl-proxy(config)# ssl-proxy vlan 100
ssl-proxy(config-vlan)# ipaddr 10.1.0.20 255.255.255.0
ssl-proxy(config-vlan)# gateway 10.1.0.1
ssl-proxy(config-vlan)# admin
ssl-proxy(config-vlan)# ^Z
ssl-proxy#
```

Configuring Telnet Remote Access

To configure the SSL daughter card for Telnet remote access, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# enable password password`	Specifies a local enable password.
Step 2	`ssl-proxy(config)# line vty starting-line-number ending-line-number`	Identifies a range of lines for configuration and enters line configuration mode.
Step 3	`ssl-proxy(config-line)# login`	Enables password checking at login.
Step 4	`ssl-proxy(config-line)# password password`	Specifies a password on the line.

This example shows how to configure the SSL daughter card for remote access:

```
ssl-proxy(config)# enable password cisco
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)#login
ssl-proxy(config-line)#password cisco
ssl-proxy(config-line)#end
ssl-proxy#
```

Configuring the Fully Qualified Domain Name

If you are using the SSL daughter card to enroll for certificates from a certificate authority, you must configure the Fully Qualified Domain Name (FQDN) on the module. The FQDN is the host name and domain name of the module.

To configure the FQDN, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# hostname name`	Configures the host name.
Step 2	`ssl-proxy(config)# ip domain-name name`	Configures the domain name.

This example shows how to configure the FQDN on the SSL daughter card:

```
ssl-proxy(config)# hostname ssl-proxy2
ssl-proxy2(config)# ip domain-name example.com
ssl-proxy2(config)# end
ssl-proxy2(config)#
```

Configuring SSH

After you complete the initial configuration for the module, enable SSH on the module, and then configure the username and password for the SSH connection using either a simple username and password or using an authentication, authorization, and accounting (AAA) server.

These sections describe how to enable and configure SSH:

- Enabling SSH on the Module, page 7-4
- Configuring the Username and Password for SSH, page 7-5
- Configuring Authentication, Authorization, and Accounting for SSH, page 7-6

Enabling SSH on the Module

SSH uses the first key pair generated on the module. In the following task, you generate a key pair used specifically for SSH.

Note If you generate a general-purpose key pair (as described in the "Generating the RSA Key Pairs" section on page 8-5) without specifying the SSH key pair first, SSH is enabled and uses the general-purpose key pair. If this key pair is later removed, SSH is disabled. To reenable SSH, generate a new SSH key pair.

To generate an SSH key pair and enable SSH, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 2	`ssl-proxy(config)# ip ssh rsa keypair-name ssh_key_name`	Assigns the key pair name to SSH.
Step 3	`ssl-proxy(config)# crypto key generate rsa general-keys label ssh_key_name`	Generates the SSH key pair. SSH is now enabled.
Step 4	`ssl-proxy(config)# end`	Exits configuration mode.
Step 5	`ssl-proxy# show ip ssh`	Shows the current state of SSH.

This example shows how to enable SSH on the module, and how to verify that SSH is enabled:

```
ssl-proxy(config)# ip ssh rsa keypair-name ssh-key
Please create RSA keys to enable SSH.
ssl-proxy(config)# crypto key generate rsa general-keys label ssh-key
The name for the keys will be: ssh-key
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys ...[OK]

ssl-proxy(config)#
*Aug 28 11:07:54.051: %SSH-5-ENABLED: SSH 1.5 has been enabled
ssl-proxy(config)# end

ssl-proxy# show ip ssh
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
ssl-proxy#
```

Configuring the Username and Password for SSH

To configure the username and password for the SSH connection, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 2	`ssl-proxy(config)# enable password` *password*	Specifies a local enable password, if not already specified.
Step 3	`ssl-proxy(config)# username` *username* {**password** | **secret**} *password*	Specifies the username and password.
Step 4	`ssl-proxy(config)# line vty` *line-number ending-line-number*	Identifies a range of lines for configuration and enters line configuration mode.
Step 5	`ssl-proxy(config-line)# login local`	Enables local username authentication.

This example shows how to configure the username and password for the SSH connection to the SSL daughter card:

```
ssl-proxy# configure terminal
ssl-proxy(config)# enable password cisco
ssl-proxy(config)# username admin password admin-pass
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)# login local
ssl-proxy(config-line)# end
```

After you configure the username and password, see the "Recovering a Lost Password" section on page 3-14 to configure the switch.

Configuring Authentication, Authorization, and Accounting for SSH

To configure authentication, authorization, and accounting (AAA) for SSH, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 2	`ssl-proxy(config)# username` *username* `secret {0 | 5}` *password*	Enables enhanced password security for the specified, unretrievable username.
Step 3	`ssl-proxy(config)# enable password` *password*	Specifies a local enable password, if not already specified.
Step 4	`ssl-proxy(config)# aaa new-model`	Enables authentication, authorization, and accounting (AAA).
Step 5	`ssl-proxy(config)# aaa authentication login default local`	Specifies the module to use the local username database for authentication.
Step 6	`ssl-proxy(config)# line vty` *line-number ending-line-number*	Identifies a range of lines for the configuration and enters line configuration mode.
Step 7	`ssl-proxy(config-line)# transport input ssh`	Configures SSH as the only protocol used on a specific line (to prevent non-SSH connections).

This example shows how to configure AAA for the SSH connection to the SSL daughter card:

```
ssl-proxy# configure terminal
ssl-proxy(config)# username admin secret admin-pass
ssl-proxy(config)# enable password enable-pass
ssl-proxy(config)# aaa new-model
ssl-proxy(config)# aaa authentication login default local
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)# transport input ssh
ssl-proxy(config-line)# end
ssl-proxy#
```

After you configure AAA, see the "Recovering a Lost Password" section on page 3-14 to configure the switch.

Configuring SSL for Client-Side and Server-Side Operation

This section describes how to configure the CSM-S. These topics are discussed in this section:

When you configure the server farm, if the real server is the SSL daughter card you must use the **local** keyword when defining the real server.

This example shows how to configure the CSM to support SSL:

```
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
```

```
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# inservice
```

The local keyword on the real server is the only CSM configuration change. Additional configuration is required on the CSM-S for proper CSM-SSL daughter card communication.

Configuring the Client Side

This example shows how to configure the SSL proxy service on the SSL daughter card:

```
ssl-proxy(config)# ssl-proxy service S1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 10.2.0.100 protocol TCP port 80
ssl-proxy(config-ssl-proxy)# inservice
```

This example shows how to configure the CSM virtual server:

```
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# inservice
```

You can perform SSL load balancing between the SSL daughter card and an SSL Services Module in mixed mode.

The CSM uses SSL-ID sticky functionality to stick SSL connections to the same SSL Services Module. The CSM must terminate the client-side TCP connection in order to inspect the SSL-ID. The CSM must then initiate a TCP connection to either the SSL daughter card or the SSL Services Module when a load-balancing decision has been made.

The traffic flow has the CSM passing all traffic received on a virtual server to the SSL daughter card with TCP termination performed on the SSL daughter card itself. When you enable the SSL sticky function, the connection between the CSM and SSL daughter card becomes a full TCP connection.

This example shows how to configure mixed-mode SSL load balancing:

```
Cat6k-2(config-module-csm)# sticky 10 ssl timeout 60
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-slb-sfarm)# real 10.2.0.21
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
Cat6k-2(config-slb-vserver)# sticky 60 group 10
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# persistent rebalance
Cat6k-2(config-slb-vserver)# inservice
```

Additionally, you must make an internally generated configuration to direct traffic at the SSL daughter card when the CSM must terminate the client-side TCP connection. You must create a virtual server with the same IP address or port of each local real server in the server farm *SSLfarm*. Internally, this virtual server is configured to direct all traffic that is intended for the virtual server to the SSL daughter card.

You must make an internally generated configuration because the IP address of the local real server and the SSL daughter card virtual server address must be the same. When the CSM initiates a connection to this local real server, the SYN frame is both sent and received by the CSM. When the CSM receives the SYN, and the destination IP address or port is the same as the virtual server VS1, it matches VS1 unless a more-specific virtual server is added.

Configuring the Server Side

A standard virtual server configuration is used for Layer 4 and Layer 7 load balancing when the SSL daughter card uses the CSM as the back-end server.

To restrict this virtual server to only receive traffic from the SSL daughter card, use the VLAN local virtual server submode command as follows:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-sfarm)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

You can configure the real server as the back end as shown in this example:

```
Cat6k-2(config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-sfarm)# predictor forward

Cat6k-2(config-module-csm)# vserver VS3
Cat6k-2(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-vserver)# inservice
```

Configuring the CSM as the Back-End Server

The virtual server and server farm configurations permits the SSL daughter card to use real servers as the back-end servers. Use the configuration that is described in the "Configuring the Client Side" section on page 7-7 and then configure the SSL daughter card to use the CSM as the back-end server:

This example shows the CSM virtual server configuration for Layer 7 load balancing:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# serverfarm SLBjpgfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.21

Cat6k-2(config-module-csm)# map JPG url
Cat6k-2(config-slb-map-cookie)# match protocol http url *jpg*

Cat6k-2(config-module-csm)# policy SLBjpg
Cat6k-2(config-slb-policy)# url-map JPG
Cat6k-2(config-slb-policy)#serverfarm SLBjpgfarm
```

```
Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# slb-policy SLBjpg
Cat6k-2(config-slb-vserver)# inservice
```

This example shows the CSM virtual server configuration for Layer 4 load balancing:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

Configuring the Real Server as the Back-End Server

The server side configuration traffic flow with the real server as the back end is similar to the client-side configuration. Use the configuration that is described in the "Configuring the Client Side" section on page 7-7 and then configure the SSL daughter card to use a real server as the back-end server:

No new configuration is required for the SSL daughter card proxy service configuration. This example shows how the configuration is internally initiated and hidden from the user:

```
ssl-proxy(config)# ssl-proxy service S1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 10.2.0.20 protocol TCP port 80
ssl-proxy(config-ssl-proxy)# inservice
```

This example shows how to configure the CSM virtual server:

```
Cat6k-2(config-module-csm)# serverfarm SSLreals

Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-sfarm)# inservice

Cat6k-2(config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-sfarm)# predictor forward

Cat6k-2(config-module-csm)# vserver VS3
Cat6k-2(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-vserver)# inservice
```

Configuring Policies

This section describes how to configure the SSL and TCP policies:

Configuring SSL Policy

Note The SSL commands for the SSL daughter card apply either globally or to a particular proxy server. See the "SSL Server Proxy Services" section on page 7-18 for procedures for applying policies to a proxy service.

The SSL policy template allows you to define parameters that are associated with the SSL stack.

One parameter that you can configure is the SSL close-protocol behavior. The SSL close protocol specifies that each of the SSL peers (client and server) should send a close-notify alert and receive a close-notify alert before closing the connection properly. If the SSL connection is not closed properly, the session is removed so that the peers cannot use the same SSL session ID in future SSL connections.

However, many SSL implementations do not follow the SSL close protocol strictly (for example, an SSL peer sends a close-notify alert but does not wait for the close-notify alert from the remote SSL peer before closing the connection).

When an SSL peer initiates the close-connection sequence, the SSL daughter card expects a close-notify alert message. If an SSL peer does not send a close-notify alert, the SSL daughter card removes the session from the session cache so that the same session ID cannot be used for future SSL connections.

When the SSL daughter card initiates the close-connection sequence, you can configure the following close-protocol options:

- strict—The SSL daughter card sends a close-notify alert message to the SSL peer, and the SSL daughter card expects a close-notify alert message from the SSL peer. If the SSL daughter card does not receive a close-notify alert, SSL resumption is not allowed for that session.
- none—The SSL daughter card does not send a close-notify alert message to the SSL peer and the SSL daughter card does not expect a close-notify alert message from the SSL peer. If the SSL daughter card receives a close-notify alert from the SSL peer, the SSL daughter card preserves the session information so that SSL resumption can be used for future SSL connections. However, if the SSL daughter card does not receive a close-notify alert from the SSL peer, SSL resumption is not allowed for that session.
- disabled (default)—The SSL daughter card sends a close-notify alert to the SSL peer; however, the SSL peer does not expect a close-notify alert before removing the session. Whether the SSL peer sends a close-notify alert or not, the session information is preserved allowing session resumption for future SSL connections.

If you do not associate an SSL policy with a particular proxy server, the proxy server enables all the supported cipher suites and protocol versions by default.

To define an SSL policy template and associate an SSL policy with a particular proxy server, perform this task:

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy policy ssl` *ssl_policy_name*	Defines SSL policy templates.
Step 2	`ssl-proxy (config-ssl-policy)# cipher {rsa-with-rc4-128-md5 | rsa-with-rc4-128-sha | rsa-with-des-cbc-sha | rsa-with-3des-ede-cbc-sha | others...}`	Configures a list of cipher-suite names acceptable to the proxy server. The cipher-suite names follow the same convention as that of existing SSL stacks.

	Command	Purpose
Step 3	`ssl-proxy (config-ssl-policy)# protocol {ssl3 | tls1 | all}`	Defines the various protocol versions supported by the proxy server.
Step 4	`ssl-proxy (config-ssl-policy)# timeout handshake time`	Configures how long the module keeps the connection in the handshake phase. The valid range is 0 to 65535 seconds.
Step 5	`ssl-proxy (config-ssl-policy)# close-protocol {strict | none}`	Configures the SSL close-protocol behavior. The close-protocol behavior is disabled by default.
Step 6	`ssl-proxy (config-ssl-policy)# session-cache`	Enables the session-caching feature. Session caching is enabled by default.
Step 7	`ssl-proxy (config-ssl-policy)# timeout session timeout [absolute[1]]`	Configures the amount of time that an entry is kept in the session cache. The valid range is 1 to 72000 seconds. **Note** You must use the **absolute** keyword to configure the session-cache size. The **absolute** keyword specifies that the session entry is kept in the session cache for the specified *timeout*. When the **absolute** keyword is specified, the new incoming connections are rejected and no free entries are available in the session cache.
Step 8	`ssl-proxy (config-ssl-policy)# session-cache size size`	(Optional) Specifies the size of the session cache[1]. The valid range is from 1 to 262143 entries. **Note** Specify the session cache size when you enter the **absolute** keyword with the **timeout session** command. If you do not enter this command or specify a *size*, the session cache size is the maximum size (262,144).

1. When the **absolute** keyword is configured, the session entry is not reused until the configured session timeout expires. When **absolute** is configured, the number of session entries required is equal to (new_connection_rate * absolute_timeout). Depending on the timeout configuration and the new connection rate, the number of session entries might be very large.You can limit the number of session entries by configuring the session-cache size.

Configuring TCP Policy

Note The TCP commands for the SSL daughter card apply either globally or to a particular proxy server.

The TCP policy template allows you to define parameters that are associated with the TCP stack.

To define an TCP policy template and associate a TCP policy with a particular proxy server, perform this task:

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy policy tcp tcp_policy_name`	Defines TCP policy templates. All defaults are assumed unless otherwise specified.
Step 2	`ssl-proxy (config-tcp-policy)# timeout syn time`	Configures the connection establishment timeout. The default is 75 seconds. The valid range is from 5 to 75 seconds.

	Command	Purpose
Step 3	`ssl-proxy (config-tcp-policy)#` **`mss`** *`max_segment_size`*	Configures the maximum segment size (MSS) in bytes that the connection will identify in the SYN packet that it generates. **Note** This command allows you to configure a different MSS for the client side and server side of the proxy server. The default is 1460 bytes. The valid range is from 256 to 2460 bytes[1].
Step 4	`ssl-proxy (config-tcp-policy)#` **`timeout reassembly`** *`time`*	Configures the amount of time in seconds before the reassembly queue is cleared. If the transaction is not complete within the specified time, the reassembly queue is cleared and the connection is dropped. The default is 60 seconds. The valid range is from 0 to 960 seconds (0 = disabled).
Step 5	`ssl-proxy (config-tcp-policy)#` **`timeout inactivity`** *`time`*	Configures the amount of time in seconds that an established connection can be inactive. The default is 600 seconds. The valid range is 0 to 960 seconds (0 = disabled).
Step 6	`ssl-proxy (config-tcp-policy)#` **`timeout fin-wait`** *`time`*	Configures the FIN wait timeout in seconds. The default value is 600 seconds. The valid range is from 75 to 600 seconds.
Step 7	`ssl-proxy (config-tcp-policy)#` **`buffer-share rx`** *`buffer_limit`*	Configures the maximum receive buffer share per connection in bytes. The default value is 32768 bytes. The valid range is from 8192 to 262144 bytes.
Step 8	`ssl-proxy (config-tcp-policy)#` **`buffer-share tx`** *`buffer_limit`*	Configures the maximum transmit buffer share per connection in bytes. The default value is 32768 bytes. The valid range is from 8192 to 262144 bytes.

1. If fragmentation occurs, decrease the MSS value until there is no fragmentation.

HTTP Header Insertion

In an SSL offloading environment, an SSL offloader terminatesthe secure client HTTP (HTTPS) connections, decrypts the SSL traffic into clear text, and forwards the clear text to a Web server through an HTTP connection. The HTTPS connections become nonsecure HTTP connections at the back-end server because it does not know that the client connection came in as a secure connection.

These reasons list why you should configure the HTTP header insertion:

- The HTTP header insertion allows the SSL daughter card to embed information into an HTTP header during a client connection. When the back-end server recognizes this header, the server returns all the URLs as HTTPS.
- You can have a back-end application that logs information per connection by configuring an SSL offloader to insert the client certificate information into the HTTP header received from the client.
- When you use the SSL daughter card in a site-to-site configuration to send traffic over a secured channel, the server end of the connection may need to know the client IP address and port information, which gets removed during NAT.

The HTTP header insertion is performed for the following methods: GET, HEAD, PUT, TRACE, POST, and DELETE. HTTP header insertion is not performed for the CONNECT method.

The custom headers and client IP and port headers are inserted in every HTTP request packet. Full session headers and decoded client certificate fields are inserted in the first HTTP request packets; only the session ID is inserted in subsequent HTTP requests that use the same session ID. The servers are expected to cache the session or client certificate headers based on the session ID and use the session ID in subsequent requests to get the session and client certificate headers.

You can configure up to 100 HTTP header insertion policies, with each policy consisting of up to 32 prefixes or headers. The prefix and custom headers can include up to 240 characters.

These sections describe the information that can be inserted in the HTTP header:

- Prefix, page 7-13
- Client Certificate Headers, page 7-13
- Client IP and Port Address Headers, page 7-14
- Custom Headers, page 7-14
- SSL Session Headers, page 7-14

Prefix

When you specify **prefix** *prefix_string*, the SSL daughter card adds the specified prefix to every inserted HTTP header. Adding a prefix enables the server to identify connections as coming from the SSL daughter card, and not from other appliances. A prefix is not added to standard HTTP headers from the client. The *prefix_string* can be up to 240 characters.

Client Certificate Headers

The client certificate header insertion allows the back-end server to see the attributes of the client certificate that the SSL daughter card has authenticated and approved. The client certificate headers are sent only once per session. The server is expected to cache these values using the session ID, which is also inserted with the headers. In subsequent requests, the server uses the session ID to look up the cached client certificate headers on the server itself.

Note If the client does not send a certificate, the SSL handshake fails. There is no data phase or header insertion.

When you specify **client-cert**, the SSL daughter card passes the following headers to the back-end server.

Field To Insert	Description
ClientCert-Valid	Certificate validity state
ClientCert-Error	Error conditions
ClientCert-Fingerprint	Hash output
ClientCert-Subject-CN	X.509 subject's common name
ClientCert-Issuer-CN	X.509 certificate issuer's common name
ClientCert-Certificate-Version	X.509 certificate version
ClientCert-Serial-Number	Certificate serial number
ClientCert-Data-Signature-Algorithm	X.509 hashing and encryption method

Field To Insert	Description
ClientCert-Subject	X.509 subject's distinguished name
ClientCert-Issuer	X.509 certificate issuer's distinguished name
ClientCert-Not-Before	Certificate is not valid before this date
ClientCert-Not-After	Certificate is not valid after this date
ClientCert-Public-Key-Algorithm	Algorithm used for the public key
ClientCert-RSA-Public-Key-Size	Size of the RSA public key
ClientCert-RSA-Modulus-Size	Size of the RSA private key
ClientCert-RSA-Modulus	RSA modulus
ClientCert-RSA-Exponent	Public RSA exponent
ClientCert-X509v3-Authority-Key-Identifier	X.509 authority key identifier
ClientCert-X509v3-Basic-Constraints	X.509 basic constraints
ClientCert-Signature-Algorithm	Certificate signature algorithm
ClientCert-Signature	Certificate signature

Client IP and Port Address Headers

Network address translation (NAT) changes the client IP address and destination TCP port number information. When you specify **client-ip-port,** the SSL daughter card inserts the client IP address and TCP destination port information in the HTTP header, allowing the server to see the client IP address and destination port number.

Custom Headers

When you specify **custom** *custom_string*, the SSL daughter card inserts the user-defined header verbatim in the HTTP header. You can configure up to 16 custom headers per HTTP header policy. The *custom_string* can include up to 240 characters.

Note The syntax for *custom_string* is in the form *name***:***value*. The *custom_string* must be enclosed in quotation marks if it contains spaces as follows:

“SOFTWARE VERSION : 2.1(1)”

SSL Session Headers

The session headers, including the session ID, are used to cache client certificates based on the session ID. The session headers are also cached based on the session ID if the server wants to track connections based on a particular cipher suite. The SSL daughter card inserts the full session headers in the HTTP request during the full SSL handshake but inserts only the session ID when the session resumes.

When you configure the SSL daughter card as a client, the SSL daughter card inserts the session ID of the connection between the module and the back-end SSL server.

When you specify **session**, the SSL daughter card passes information specific to an SSL connection to the back-end server in the form of the following session headers.

Field to insert	Description
Session-Id	SSL session ID
Session-Cipher-Name	Symmetric cipher suite
Session-Cipher-Key-Size	Symmetric cipher key size
Session-Cipher-Use-Size	Symmetric cipher use

Configuring the HTTP Header Insertion

To configure the HTTP header insertion, perform this task:

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy policy http-header` *policy_name*	Configures HTTP header insertion.
Step 2	`ssl-proxy(config-http-header-policy)#` {**prefix** *prefix_string* | **client-cert** | **client-ip-port** | **custom** *custom_string*} | **session**]	Specifies the prefix or type of header.
Step 3	`ssl-proxy(config-http-header-policy)# exit`	Returns to config mode.
Step 4	`ssl-proxy(config)# ssl-proxy service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case sensitive.
Step 5	`ssl-proxy(config-ssl-proxy)# policy http-header` *http_header_policy_name*	Applies the HTTP header policy to the proxy server, for request.

This example shows how to configure the SSL daughter card to insert a prefix and session headers:

```
ssl-proxy (config)# ssl-proxy policy http-header ssl-offload
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# session
ssl-proxy(config-http-header-policy)# custom "SOFTWARE VERSION:2.1(1)"
ssl-proxy(config-http-header-policy)# custom "module:SSL MODULE - CATALYST 6500"
ssl-proxy(config-http-header-policy)# custom type-of-proxy:server_proxy_1024_bit_key_size
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
```

In addition to the standard HTTP headers, the following header information is inserted:

```
SSL-OFFLOAD-SOFTWARE VERSION:2.1(1)
SSL-OFFLOAD-module:SSL MODULE - CATALYST 6500
SSL-OFFLOAD-type-of-proxy:server_proxy_1024_bit_key_size
SSL-OFFLOAD-Session-Id:33:FF:2C:2D:25:15:3C:50:56:AB:FA:5A:81:0A:EC:E9:00:00:0A:03:00:60:
 2F:30:9C:2F:CD:56:2B:91:F2:FF
SSL-OFFLOAD-Session-Cipher-Name:RC4-SHA
SSL-OFFLOAD-Session-Cipher-Key-Size:128
SSL-OFFLOAD-Session-Cipher-Use-Size:128
```

Configuring URL Rewrite

In a typical SSL offloading environment, an SSL offloader terminates secure client HTTP (HTTPS) connections, decrypts the SSL traffic into clear text, and forwards the clear text to a Web server through an HTTP connection. The HTTPS connections become nonsecure HTTP connections at the back-end server. The back-end server does not know that the client connection came in as a secure connection.

If the data returned to the client contains an HTTP redirection link, and the client follows this link, the client leaves the secure domain and no longer has a secure connection. The redirected link may not be available from the server using a clear text connection.

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 service in the SSL proxy list. The URL rewrite rules resolve the problem of a website redirecting you to a nonsecure HTTP URL by rewriting the domain from http:// to https://. By configuring URL rewrite, all client connections to the Web server are SSL connections, ensuring the secure delivery of HTTPS content back to the client.

Note URL rewrite supports the rewriting of redirection links. The system scans only the "Location:" HTTP header field in the response from the server and rewrites the rules accordingly. URL rewrite does not support embedded links.

URL rewrite rewrites the protocol and the nondefault port (default ports are port 80 for clear text and port 443 for SSL).

You can configure up to 100 URL rewrite policies with each policy consisting of up to 32 rewrite rules per SSL proxy service and up to 200 characters per rule.

The guidelines for URL rewrite are as follows:

- An exact URL match takes precedence over a wildcard rule. A suffix wildcard rule takes precedence over a prefix wildcard rule.

  For example, **www.cisco.com** takes precedence, then **www.cisco.***, and then ***.cisco.com**.
- Enter only one suffix or prefix wildcard rule at one time. For example, do not enter **www.cisco.*** and **www.cisco.c*** in the same policy or ***w.cisco.com** and ***.cisco.com** in the same policy.
- Do not enter two exact URL match rules in the same policy. For example, do not enter **www.cisco.com clearport 80 sslport 443** and **www.cisco.com clearport 81 sslport 444** in the same policy. In this case, the second rule overwrites the first rule.
- URL rewrite is performed for both offload and back-end servers (HTTP to HTTPS and HTTPS to HTTP). This includes port rewrites.

To configure URL rewrite, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy policy url-rewrite` *policy_name*	Configures the URL rewrite policy.
Step 2	`ssl-proxy(config-url-rewrite-policy)# url` *url* [**clearport** *port_number*[1]] [**sslport** *port_number*[2]]	Specifies the URL rewrite rules. You can configure up to 32 rewrite rules per SSL proxy service and up to 240 characters per rule. **Note** You should enter only one suffix or prefix wildcard character (*) once per rewrite rule.

	Command	Purpose
Step 3	`ssl-proxy(config-url-rewrite-policy)# exit`	Returns to config mode.
Step 4	`ssl-proxy(config)# ssl-proxy service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case sensitive.
Step 5	`ssl-proxy(config-ssl-proxy)# policy url-rewrite` *policy_name*	Applies the URL rewrite policy.

1. The **clearport** *port_number* specifies the port portion of the URL to be rewritten. Specify the **cleartext** *port_number* if it is not the default cleartext port 80.
2. The **sslport** *port_number* specifies the port portion of the URL that should be rewritten. Specify the **ssltext** *port_number* if it is not the default SSL port 443.

This example shows how to configure URL rewrite policy and apply the policy to a proxy service:

```
ssl-proxy(config)# ssl-proxy policy url-rewrite cisco_url
ssl-proxy(config-ssl-proxy)# url www.cisco.*
ssl-proxy(config-ssl-proxy)# url www.cisco.com clearport 81 sslport 444
ssl-proxy(config-ssl-proxy)# url wwwin.cisco.com clearport 81 sslport 440
ssl-proxy(config-ssl-proxy)# url 10.1.1.10 clearport 81 sslport 444
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)# ssl-proxy service cisco_service
ssl-proxy(config-ssl-proxy)# policy url-rewrite cisco_url
```

See Table 7-1 for examples that show URL rewrite.

Table 7-1 Rules and Outcome for Server Proxy

URL Rewrite Rule	URLs that Match	URL Rewrite
url www.cisco.com	http://www.cisco.com/	https://www.cisco.com/
url www.cisco.com clearport 81	http://www.cisco.com:81/	https://www.cisco.com/
url www.cisco.com sslport 444	http://www.cisco.com/	https://www.cisco.com:444/
url www.cisco.com clearport 81 sslport 444	http://www.cisco.com:81/	https://www.cisco.com:444/

Configuring the SSL Proxy Services

You define the SSL proxy services using the **ssl-proxy service** *ssl_proxy_name* command. You can configure the virtual IP address and port that is associated with the proxy service and the associated target IP address and port.

You define the TCP and SSL policies for both client (**virtual**) and server (**server**) sides of the proxy.

These sections describe how to configure the proxy services:

- SSL Server Proxy Services, page 7-18
- SSL Version 2.0 Forwarding, page 7-20
- SSL Client Proxy Services, page 7-20

SSL Server Proxy Services

To configure the SSL server proxy services, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy service service_name`	Defines the name of the SSL proxy service. **Note** The *service_name* value is case sensitive.
Step 2	`ssl-proxy(config-ssl-proxy)# virtual ipaddr ip_addr [mask_addr]`[1] ``[2]protocol tcp port port {secondary}`	Defines the virtual server IP address, transport protocol (TCP), and port number for which the CSM-S is the proxy. **Note** The **secondary** keyword is always required.
Step 3	`ssl-proxy(config-ssl-proxy)# server ipaddr ip_addr protocol tcp port port`	Defines the IP address, port number, and transport protocol of the target server for the proxy. **Note** The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.
Step 4	`ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp_policy_name`[3]	(Optional) Applies a TCP policy to the client side of the proxy server. See the "Configuring TCP Policy" section on page 7-11 for TCP policy parameters.
Step 5	`ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl_policy_name`[3]	(Optional) Applies an SSL policy to the client side of the proxy server. See the "Configuring SSL Policy" section on page 7-10 for SSL policy parameters.
Step 6	`ssl-proxy(config-ssl-proxy)# server policy tcp tcp_policy_name`	(Optional) Applies a TCP policy to the server side of the proxy server. See the "Configuring TCP Policy" section on page 7-11.
Step 7	`ssl-proxy(config-ssl-proxy)# policy http-header http_header_policy_name`	(Optional) Applies the HTTP header policy to the proxy server. See the "HTTP Header Insertion" section on page 7-12.
Step 8	`ssl-proxy(config-ssl-proxy)# policy url-rewrite url_rewrite_policy_name`	(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 7-16.
Step 9	`ssl-proxy(config-ssl-proxy)# trusted-ca ca_pool_name`	(Optional) Associates the trusted certificate authority pool with the proxy service. See the "Client Certificate Authentication" section on page 8-41 for information on the certificate authority pools.
Step 10	`ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only`[4] `| all`[5]`}`	(Optional) Enables the server certificate authentication and specifies the form of verification. See the "Server Certificate Authentication" section on page 8-43 for information on the server certificate authentication.
Step 11	`ssl-proxy(config-ssl-proxy)# nat {server | client natpool_name}`	(Optional) Specifies the usage of either server NAT[6] or client NAT for the server-side connection opened by the CSM-S. See the "Configuring NAT" section on page 7-22 and "Configuring NAT" section on page 7-22.

	Command	Purpose
Step 12	`ssl-proxy(config-ssl-proxy)#` **`certificate rsa general-purpose trustpoint`** *`trustpoint_label`*	Applies a trustpoint configuration to the proxy server[7]. **Note** The trustpoint defines the certificate authority server, the key parameters and key-generation methods, and the certificate enrollment methods for the proxy server. See the "Declaring the Trustpoint" section on page 8-7 for information on configuring the trust point.
Step 13	`ssl-proxy(config-ssl-proxy)#` **`inservice`**	Sets the proxy server as administratively Up.

1. Configure the mask address to specify a wildcard proxy service. You must enter the **secondary** keyword to configure a wildcard proxy service.
2. When you enter the secondary keyword, the SSL daughter card does not respond to the ARP requests of the virtual IP address.
3. If you create a policy without specifying any parameters, the policy is created using the default values.
4. When you verify signature-only, authentication stops at the level that corresponds to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
5. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL daughter card authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint for the highest level certificate authority, and this trustpoint should be authenticated.
6. NAT = network address translation
7. If the key (modulus) size is other than 512, 768, 1024, 1536, or 2048, you will receive an error and the trustpoint configuration is not applied. Replace the key by generating a key (using the same *key_label*) and specifying a supported modulus size, and then repeat Step 12.

This example shows how to configure SSL proxy services:

```
ssl-proxy(config)# ssl-proxy service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# nat client t2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint tp1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

If you have many virtual and server IP addresses to manage and configure, you can configure a wildcard proxy service.

This example shows how to configure a wildcard SSL proxy service, so that **proxy1** accepts virtual IP addresses 10.0.0.1 through 10.25.255.254:

```
ssl-proxy(config)# ssl-proxy service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.0.0.0 255.0.0.0 protocol tcp port 443
secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 20.1.2.3 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

SSL Version 2.0 Forwarding

The SSL daughter card is not able to terminate SSL version 2.0 (SSLv2) connections. However, you can configure the SSL daughter card to forward SSLv2 connections to another server by entering the **sslv2** keyword at the **server** command. When you configure the SSLv2 server IP address, the SSL daughter card transparently forwards all SSLv2 connections to that server. If you require SSLv2 forwarding, you need to configure the SSLv2 server IP address in addition to the IP address of the server that is used for offloading SSL version 3.0 or Transport Layer Security (TLS) version 1.0 connections.

To configure SSLv2 forwarding, perform this task:

Command	Purpose
`ssl-proxy(config-ssl-proxy)#` **`server ipaddr`** *`ip_addr`* **`protocol tcp port`** *`port`* **`sslv2`**[1]	Defines the IP address, port number, and the transport protocol of the target server for the proxy. **Note** The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.

1. Enter the **sslv2** keyword to forward SSL version 2.0 client connections to a SSL v2.0 server. When you enter **sslv2**, configure another server IP address to offload SSL version 3.0 or Transport Layer Security (TLS) version 1.0 connections.

This example shows how to configure the SSL proxy services to forward SSL v2.0 connections:

```
ssl-proxy(config)# ssl-proxy service frontend
ssl-proxy(config-ssl-proxy)# virtual ipaddr 35.200.200.102 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.2 protocol tcp port 443 sslv2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

SSL Client Proxy Services

You configure SSL client proxy services to specify that the proxy service accepts clear text traffic, encrypts the traffic into SSL traffic, and forwards the traffic to the back-end SSL server.

While you are required to configure a certificate for the SSL server proxy, you are not required to configure a certificate for the SSL client proxy. If you configure the certificate for the SSL client proxy, that certificate is sent in response to the certificate request message that is sent by the server during the client authentication phase of the handshake protocol.

Note The SSL policies are configured at the **server** subcommand for the SSL client proxy services; the SSL policies are configured at the **virtual** subcommand for the SSL server proxy services.

To configure SSL client proxy services, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy service` *proxy_name* **client**	Defines the name of the SSL proxy service. The **client** keyword configures the SSL client proxy service. **Note** The *proxy-name* value is case sensitive.
Step 2	`ssl-proxy(config-ssl-proxy)#` **virtual ipaddr** *ip_addr* [*mask_addr*][1] **protocol tcp port** *port* **secondary**	Defines the virtual server IP address, transport protocol (TCP), and port number for which the CSM-S is the proxy. **Note** The **secondary** keyword is required.
Step 3	`ssl-proxy(config-ssl-proxy)#` **server ipaddr** *ip_addr* **protocol tcp port** *port*	Defines the IP address, port number, and transport protocol of the target server for the proxy. **Note** The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.
Step 4	`ssl-proxy(config-ssl-proxy)#` **virtual policy tcp** *tcp_policy_name*[2]	(Optional) Applies a TCP policy to the client side of the proxy server. See the "Configuring TCP Policy" section on page 7-11 for the TCP policy parameters.
Step 5	`ssl-proxy(config-ssl-proxy)#` **server policy ssl** *ssl_policy_name*[2]	(Optional) Applies an SSL policy to the server side of the proxy server. See the "Configuring SSL Policy" section on page 7-10 for the SSL policy parameters.
Step 6	`ssl-proxy(config-ssl-proxy)#` **server policy tcp** *tcp_policy_name*	(Optional) Applies a TCP policy to the server side of the proxy server. See the "Configuring TCP Policy" section on page 7-11.
Step 7	`ssl-proxy(config-ssl-proxy)#` **policy http-header** *http_header_policy_name*	(Optional) Applies the HTTP header policy to the proxy server. See the "HTTP Header Insertion" section on page 7-12.
Step 8	`ssl-proxy(config-ssl-proxy)#` **policy url-rewrite** *url_rewrite_policy_name*	(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 7-16.
Step 9	`ssl-proxy(config-ssl-proxy)#` **trusted-ca** *ca_pool_name*	Associates the trusted certificate authority pool with the proxy service. See the "Client Certificate Authentication" section on page 8-41 for information on the certificate authority pools.
Step 10	`ssl-proxy(config-ssl-proxy)#` **authenticate verify** {**signature-only**[3] | **all**[4]}	Enables the client certificate authentication and specifies the form of verification. See the "Client Certificate Authentication" section on page 8-41 for information on the client certificate authentication.
Step 11	`ssl-proxy(config-ssl-proxy)#` **nat** {**server** | **client** *natpool_name*}	(Optional) Specifies the usage of either server NAT[5] or client NAT for the server-side connection opened by the SSL daughter card. See the "Configuring NAT" section on page 7-22.

	Command	Purpose
Step 12	`ssl-proxy(config-ssl-proxy)#` **`certificate rsa general-purpose trustpoint`** *`trustpoint_label`*	(Optional) Applies a trustpoint configuration to the client proxy. **Note** The trustpoint defines the certificate authority server, the key parameters and key-generation methods, and the certificate enrollment methods for the proxy server. See the "Declaring the Trustpoint" section on page 8-7 for information on configuring the trust point.
Step 13	`ssl-proxy(config-ssl-proxy)#` **`inservice`**	Sets the proxy server as administratively Up.

1. Configure the mask address to specify a wildcard proxy service. You must enter the **secondary** keyword to configure a wildcard proxy service.
2. If you create a policy without specifying any parameters, the policy is created using the default values.
3. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
4. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL daughter card authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint for the highest level certificate authority, and this trustpoint should be authenticated.
5. NAT = network address translation

This example shows how to configure SSL client proxy services:

```
ssl-proxy(config)# ssl-proxy service proxy1 client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy ssl ssl1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

Configuring NAT

The client connections originate from the client and are terminated on the SSL daughter card. The server connections originate from the SSL daughter card.

You can configure client NAT, server NAT, or both, on the server connection.

Note If Client NAT is configured on the SSL daughter card, then you must also configure it on the CSM side for it to work.

Server NAT

The server IP address configured with the **ssl-proxy service** command specifies the IP address and port for the destination device, either the SSL daughter card or the real server for which the SSL daughter card acts as a proxy. If you configure server NAT, the server IP address is used as the destination IP address for the server connection. If the server NAT is not configured, the destination IP address for the

server connection is the same as the **virtual ipaddress** for which the SSL daughter card is a proxy. The SSL daughter card always performs the port translation by using the port number entered in the **server ipaddress** subcommand.

To configure server NAT, perform this task:

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy service ssl_proxy_name`	Defines the SSL proxy service.
Step 2	`ssl-proxy (config-ssl-proxy)# nat server`	Enables a NAT server address for the server connection of the specified service SSL offload.

Client NAT

If you configure client NAT, the server connection source IP address and port are derived from a NAT pool. If client NAT is not configured, the server connection source IP address and port are derived from the source IP address and source port of the client connection.

Allocate enough IP addresses to satisfy the total number of connections supported by the SSL daughter card (256,000 connections). Assuming that you have 32,000 ports per IP address, configure 8 IP addresses in the NAT pool. If you try to configure fewer IP addresses than required by the total connections supported by the SSL daughter card, the command is rejected.

To configure a NAT pool and assign the NAT pool to the proxy service, perform this task:

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy natpool natpool_name start_ip_addr end_ip_addr netmask`	Defines a pool of IP addresses that the SSL daughter card uses for implementing the client NAT.
Step 2	`ssl-proxy (config-ssl-proxy)# ssl-proxy service ssl_proxy_name`	Defines the SSL proxy service.
Step 3	`ssl-proxy (config-ssl-proxy)# nat client natpool_name`	Configures a NAT pool for the client address used in the server connection of the specified service SSL offload.

Configuring TACACS, TACACS+, and RADIUS

For information on configuring TACACS, TACACS+, and RADIUS, refer to these URLs:

- "Configuring RADIUS" chapter in the *Cisco IOS Security Configuration Guide, Release 12.2*:
  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/fsecsp/scfrad.htm
- "Configuring TACACS+" chapter in the *Cisco IOS Security Configuration Guide, Release 12.2*:
  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/fsecsp/scftplus.htm

Configuring SNMP Traps

For a list of supported MIBs, refer to this URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

Note The Cisco product MIB ID for the CSM-S is ciscoproducts.610. This is different than the SSLM, which is ciscoproducts.554.

To enable SNMP traps, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# snmp-server host` *addr* `traps version` *version* `ssl-proxy`	Specifies the IP address of an external network management device to which traps are sent.
Step 2	`ssl-proxy(config)# snmp-server enable traps ssl-proxy cert-expiring`	(Optional) Enables the SSL proxy certificate expiration notification trap. **Note** If you set the certificate check-expiring interval to **0**, expiration notification traps are not sent. See the "Configuring Route Health Injection" section on page A-13 for information on enabling certificate expiration warnings. **Note** Expiration notification traps are sent only for proxy service certificates that are currently configured.
Step 3	`ssl-proxy(config)# snmp-server enable traps ssl-proxy oper-status`	(Optional) Enables the SSL proxy operation status notification trap.
Step 4	`ssl-proxy(config)# snmp-server queue-length` *length*	(Optional) Specifies the number of trap events that are held before the queue must be emptied. The default *length* is 10; valid values are 1 through 1000.
Step 5	`ssl-proxy# show snmp`	Displays the SNMP information.

This example shows how to enable SNMP traps:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# snmp-server host 10.1.1.1 traps version 2c ssl-proxy
ssl-proxy(config)# snmp-server enable traps ssl-proxy cert-expiring
*Nov 27 03:47:10.739:%STE-6-PROXY_CERT_EXPIRING_TRAP_ENABLED:SNMP trap for proxy service
certificate expiration warning has been enabled.
ssl-proxy(config)# snmp-server enable traps ssl-proxy oper-status
*Nov 27 03:46:59.607:%STE-6-PROXY_OPER_STATUS_TRAP_ENABLED:SNMP trap for proxy service
operational status change has been enabled.
ssl-proxy(config)# snmp-server queue-length 256
ssl-proxy(config)# end

ssl-proxy# show snmp
0 SNMP packets input
    0 Bad SNMP version errors
    0 Unknown community name
    0 Illegal operation for community name supplied
```

```
    0 Encoding errors
    0 Number of requested variables
    0 Number of altered variables
    0 Get-request PDUs
    0 Get-next PDUs
    0 Set-request PDUs
8 SNMP packets output
    0 Too big errors (Maximum packet size 1500)
    0 No such name errors
    0 Bad values errors
    0 General errors
    0 Response PDUs
    8 Trap PDUs

SNMP logging:enabled
    Logging to 10.1.1.1.162, 0/256, 0 sent, 0 dropped.
ssl-proxy#
```

Enabling the Cryptographic Self-Test

Note The power-on crypto chip self-test and key test are run only once at bootup.

Note Use the self-test for troubleshooting only. Running this test will impact run-time performance.

To run the self-test, perform this task:

Command	Purpose
ssl-proxy(config)# **ssl-proxy crypto self-test time-interval** *time*	Enables the crytographic self-test. The default value for *time* is 3 seconds; valid values are 1 though 8.

This example shows how to enable the cryptographic self-test and display cryptographic information:

```
ssl-proxy(config)# ssl-proxy crypto self-test time-interval 1
ssl-proxy(config)# end
```

Displaying Statistics Information

To display statistics information, perform this task:

Command	Purpose
ssl-proxy(config)# **show ssl-proxy stats** {**crypto** | **hdr** | **ipc** | **pki** [**auth** | **cache** | **cert-header** | **database** | **expiring** | **history** | **ipc** | **memory**] | **service** | **ssl** | **tcp** | **url**}	Displays specified statistics information.

This example shows how to display header insertion information:

```
ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
    Session Headers Inserted :1          Custom Headers Inserted :0
    Session Id's Inserted    :2          Client Cert. Inserted   :0
    Client IP/Port Inserted  :0
    No End of Hdr Detected   :0          Payload no HTTP header  :0
    Desc Alloc Failed        :0          Buffer Alloc Failed     :0
    Client Cert Errors       :0          No Service              :0
```

This example shows how to display crypto information:

```
ssl-proxy# show ssl-proxy stats crypto
Crypto Statistics from SSL Module:1
  Self-test is running
  Current device index is 1
  Time interval between tests is 1 seconds
  Device 0 statistics:
    Total Number of runs:50
    Runs all passed:50
    Number of timer error:0
    ------------------------------------------------------
    Test Name                     Passed  Failed  Did-not-run
    ------------------------------------------------------
      0  Power-on Crypto chip sel    1       0       0
      1  Power-on Crypto chip key    1       0       0
      2  Hash Test Case 1           50       0       0
      3  Hash Test Case 2           50       0       0
      4  Hash Test Case 3           50       0       0
      5  Hash Test Case 4           50       0       0
      6  SSL3 MAC Test Case 1       50       0       0
      7  SSL3 MAC Test Case 2       50       0       0
      8  TLS1 MAC Test Case 1       50       0       0
      9  TLS1 MAC Test Case 2       50       0       0
     10  DES Server Test            50       0       0
     11  DES Encrypt Test 1         50       0       0
     12  DES Decrypt Test 1         50       0       0
     13  DES Encrypt Test 2         50       0       0
     14  DES Decrypt Test 2         50       0       0
     15  ARC4 Test Case 1           50       0       0
     16  ARC4 Test Case 2           50       0       0
     17  ARC4 Test Case 3           50       0       0
     18  ARC4 State Test Case 1     50       0       0
     19  ARC4 State Test Case 2     50       0       0
     20  ARC4 State Test Case 3     50       0       0
     21  ARC4 State Test Case 4     50       0       0
     22  HMAC Test Case 1           50       0       0
     23  HMAC Test Case 2           50       0       0
     24  Random Bytes Generation    50       0       0
     25  RSA Encrypt/Decrypt Test   50       0       0
     26  Master Secret Generation   50       0       0
     27  Key Material Generation    50       0       0
     28  SSL3 Handshake Hash Test   50       0       0
     29  TLS1 Handshake Hash Test   50       0       0

  Device 1 statistics:
    Total Number of runs:49
    Runs all passed:49
    Number of timer error:0
    ------------------------------------------------------
    Test Name                     Passed  Failed  Did-not-run
    ------------------------------------------------------
      0  Power-on Crypto chip sel    1       0       0
```

```
 1  Power-on Crypto chip key    1      0      0
 2  Hash Test Case 1           50      0      0
 3  Hash Test Case 2           50      0      0
 4  Hash Test Case 3           50      0      0
 5  Hash Test Case 4           50      0      0
 6  SSL3 MAC Test Case 1       50      0      0
 7  SSL3 MAC Test Case 2       50      0      0
 8  TLS1 MAC Test Case 1       50      0      0
 9  TLS1 MAC Test Case 2       50      0      0
10  DES Server Test            50      0      0
11  DES Encrypt Test 1         50      0      0
12  DES Decrypt Test 1         50      0      0
13  DES Encrypt Test 2         50      0      0
14  DES Decrypt Test 2         50      0      0
15  ARC4 Test Case 1           50      0      0
16  ARC4 Test Case 2           50      0      0
17  ARC4 Test Case 3           50      0      0
18  ARC4 State Test Case 1     49      0      0
19  ARC4 State Test Case 2     49      0      0
20  ARC4 State Test Case 3     49      0      0
21  ARC4 State Test Case 4     49      0      0
22  HMAC Test Case 1           49      0      0
23  HMAC Test Case 2           49      0      0
24  Random Bytes Generation    49      0      0
25  RSA Encrypt/Decrypt Test   49      0      0
26  Master Secret Generation   49      0      0
27  Key Material Generation    49      0      0
28  SSL3 Handshake Hash Test   49      0      0
29  TLS1 Handshake Hash Test   49      0      0
```

This example shows how to display PKI certificate authentication and authorization statistics:

```
ssl-proxy# show ssl-proxy stats pki auth
Authentication request timeout:240 seconds
Max in process:100 (requests)
Max queued before dropping:0 (requests)
Certificate Authentication & Authorization Statistics:
  Requests started:2
  Requests finished:2
  Requests pending to be processed:0
  Requests waiting for CRL:0
  Signature only requests:0
  Valid signature:0
  Invalid signature:0
  Total number of invalid certificates:0
  Approved with warning (no crl check):2
  Number of times polling CRL:0
  No certificates present:0
  Failed to get CRL:0
  Not authorized (e.g. denied by ACL):0
  Root certificates not self-signed:0
  Verify requests failed (e.g. expired or CRL operation failed):0
  Unknown failure:0
  Empty certificate chain:0
  No memory to process requests:0
  DER encoded certificates missing:0
  Bad DER certificate length:0
  Failed to get key from certificate:0
  Issuer CA not in trusted CA pool:0
  Issuer CA certificates not valid yet:0
  Expired issuer CA certificates:0
  Peer certificates not valid yet:0
  Expired peer certificates:0
```

This example shows how to display PKI peer certificate cache statistics:

```
ssl-proxy# show ssl-proxy stats pki cache
Peer certificate cache size:0 (entries), aging timeout:30 (minutes)
Peer certificate cache statistics:
  In use:0 (entries)
  Cache hit:0
  Cache miss:0
  Cache allocated:0
  Cache freed:0
  Cache entries expired:0
  Cache error:0
  Cache full (wrapped around):0
  No memory for caching:0
```

This example shows how to display the forwarding data unit statistics:

```
ssl-proxy# show ssl-prox stats fdu
FDU Statistics:
    IP Frag Drops       : 0             IP Version Drops    : 0
    IP Addr Discards    : 0             Serv_Id Drops       : 0
    Conn Id Drops       : 0             Bound Conn Drops    : 0
    Vlan Id Drops       : 0             TCP Checksum Drops  : 0
    Hash Full Drops     : 0             Hash Alloc Fails    : 0
    Flow Creates        : 536701        Flow Deletes        : 536701
    Conn Id allocs      : 268354        Conn Id deallocs    : 268354
    Tagged Pkts Drops   : 0             Non-Tagg Pkts Drops : 0
    Add ipcs            : 3             Delete ipcs         : 0
    Disable ipcs        : 1             Enable ipcs         : 0
    Unsolicited ipcs    : 1345          Duplicate Add ipcs  : 0
    IOS Broadcast Pkts  : 43432         IOS Unicast Pkts    : 12899
    IOS Multicast Pkts  : 0             IOS Total Pkts      : 56331
    IOS Congest Drops   : 0             SYN Discards        : 0
FDU Debug Counters:
    Inv. Conn Drops     : 0             Inv. Conn Pkt Drops : 0
    Inv. TCP opcodes    : 0
    Inv. Fmt Pkt Drops  : 0             Inv. Bad Vlan ID    : 0
    Inv. Bad Ctl Command: 0             Inv. TCP Congest    : 0
    Inv. Bad Buffer Fmt : 0             Inv. Buf Undersized : 0
ssl-proxy#
```

Collecting Crash Information

The crash-info feature collects information for developers to fix software-forced resets. Enter the **show ssl-proxy crash-info** command to collect software-forced reset information. You can retrieve only the latest crash-info in case of multiple software-forced resets. The **show ssl-proxy crash-info** command takes 1 to 6 minutes to complete the information collection process.

Note The **show stack** command is not a supported command to collect software-forced reset information on the SSL daughter card.

This example shows how to collect software-forced reset information:

```
ssl-proxy# show ssl-proxy crash-info

===== SSL daughter card - START OF CRASHINFO COLLECTION =====
```

```
------------- COMPLEX 0 [FDU_IOS] ----------------------

NVRAM CHKSUM:0xEB28
NVRAM MAGIC:0xC8A514F0
NVRAM VERSION:1

++++++++++ CORE 0 (FDU) ++++++++++++++++++++++

   CID:0
   APPLICATION VERSION:2003.04.15 14:50:20 built for cantuc
   APPROXIMATE TIME WHEN CRASH HAPPENED:14:06:04 UTC Apr 16 2003
   THIS CORE DIDN'T CRASH
   TRACEBACK:222D48 216894
   CPU CONTEXT  -----------------------------

$0 :00000000, AT :00240008, v0 :5A27E637, v1 :000F2BB1
a0 :00000001, a1 :0000003C, a2 :002331B0, a3 :00000000
t0 :00247834, t1 :02BFAAA0, t2 :02BF8BB0, t3 :02BF8BA0
t4 :02BF8BB0, t5 :00247834, t6 :00000000, t7 :00000001
s0 :00000000, s1 :0024783C, s2 :00000000, s3 :00000000
s4 :00000001, s5 :0000003C, s6 :00000019, s7 :0000000F
t8 :00000001, t9 :00000001, k0 :00400001, k1 :00000000
gp :0023AE80, sp :031FFF58, s8 :00000019, ra :00216894
LO :00000000, HI :0000000A, BADVADDR :828D641C
EPC :00222D48, ErrorEPC :BFC02308, SREG :34007E03
Cause 0000C000 (Code 0x0):Interrupt exception

CACHE ERROR registers  -------------------

CacheErrI:00000000, CacheErrD:00000000
ErrCtl:00000000, CacheErrDPA:0000000000000000

   PROCESS STACK -----------------------------
      stack top:0x3200000

   Process stack in use:

   sp is close to stack top;

   printing 1024 bytes from stack top:

031FFC00:06405DE0 002706E0 0000002D 00000001  .@]`.'.`...-....
031FFC10:06405DE0 002706E0 00000001 0020B800  .@]`.'.`..... 8.
031FFC20:031FFC30 8FBF005C 14620010 24020004  ..|0.?.\.b..$...
...........
...........
...........
FFFFFFD0:00000000 00000000 00000000 00000000 ................
FFFFFFE0:00627E34 00000000 00000000 00000000 .b~4............
FFFFFFF0:00000000 00000000 00000000 00000006 ................


===== SSL daughter card - END OF CRASHINFO COLLECTION =======
```

Enabling VTS Debugging

A virtual terminal server (VTS) is built into the SSL daughter card for debugging different processors (FDU, TCP, SSL) on the module.

Note Use the TCP debug commands only to troubleshoot basic connectivity issues under little or no load conditions (for instance, when no connection is being established to the virtual server or real server).

If you use TCP debug commands, the TCP module displays large amounts of debug information on the console, which can significantly slow down module performance. Slow module performance can lead to delayed processing of TCP connection timers, packets, and state transitions.

From a workstation or PC, make a Telnet connection to one of the VLAN IP addresses on the module to port 2001 to view debug information.

To display debugging information, perform this task:

Command	Purpose
ssl-proxy# [**no**] **debug ssl-proxy** {**fdu** | **ssl** | **tcp**} [*type*]	Turns on or off the debug flags for the specified system component.

After you make the Telnet connection, enter the **debug ssl-proxy** {**tcp** | **fdu** | **ssl**} command from the SSL Certificate Management console. One connection is sent from a client and displays the logs found in TCP console.

This example shows how to display the log for TCP states for a connection and verify the debugging state:

```
ssl-proxy# debug ssl-proxy tcp state
ssl-proxy# show debugging
STE Mgr:
  STE TCP states debugging is on
```

This example shows the output from the workstation or PC:

```
Conn 65066 state CLOSED --> state SYN_RECEIVED
Conn 65066 state SYN_RECEIVED --> state ESTABLISHED
Conn 14711 state CLOSED --> state SYN_SENT
Conn 14711 state SYN_SENT --> state ESTABLISHED
Conn 14711 state ESTABLISHED --> state CLOSE_WAIT
Conn 65066 state ESTABLISHED --> state FIN_WAIT_1
Conn 65066 state FIN_WAIT_1 --> state FIN_WAIT_2
Conn 65066 state FIN_WAIT_2 --> state TIME_WAIT
Conn 14711 state CLOSE_WAIT --> state LAST_ACK
Conn 14711 state LAST_ACK --> state CLOSED
###############Conn 65066 state TIME_WAIT --> state CLOSED
```