



SSL Configuration Examples

This appendix contains these sections:

- [CSM-S Configuration Example \(Bridge Mode, No NAT\)](#), page B-1
- [CSM-S Configuration Example \(Router Mode, Server NAT\)](#), page B-7
- [CSM-S and SSLM Configuration Example \(Router Mode, Server NAT\)](#), page B-12
- [Integrated Secure Content-Switching Service Example](#), page B-16
- [Certificate Security Attribute-Based Access Control Examples](#), page B-19
- [HTTP Header Insertion Examples](#), page B-21
- [URL Rewrite Examples](#), page B-26

CSM-S Configuration Example (Bridge Mode, No NAT)

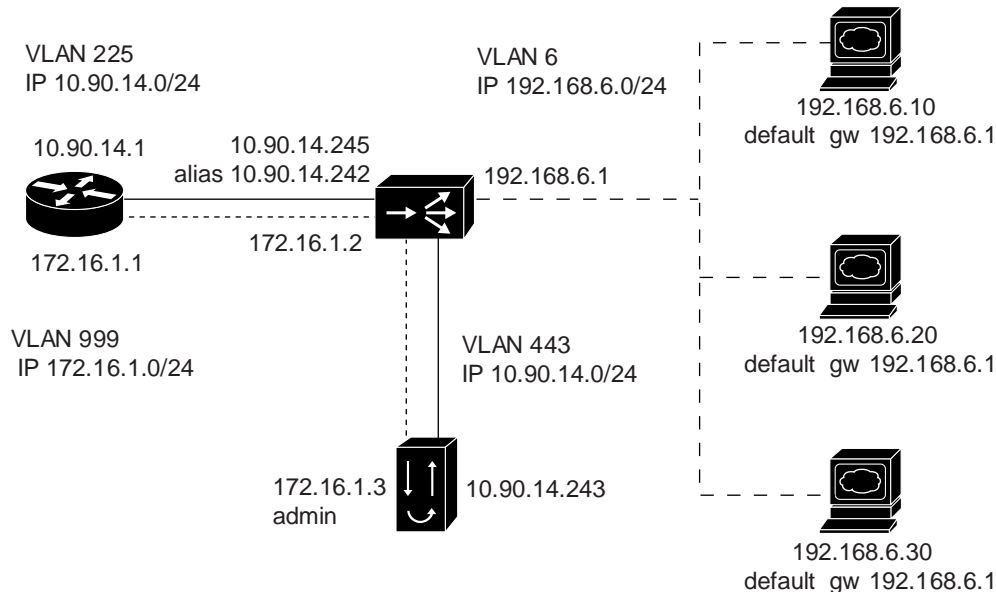
This section describes a CSM-S configuration, which allows a client to load balance HTTP to three web servers (IP addresses 192.168.6.10, 192.168.6.20, and 192.168.6.30) and offload HTTPS, and then load balance to the same three web servers.

In this example, the CSM-S client VLAN and the server VLAN for the SSL daughter card are configured in the same IP subnet (bridge mode), while the web servers are on a private IP network and reside in a separate VLAN. (See [Figure B-1](#).)

The CSM-S is configured so that it does not perform NAT operations when it is directing encrypted traffic to the SSL daughter card. The SSL daughter card is also configured not to perform NAT operations when it is sending decrypted traffic back to the CSM-S for load balancing the decrypted traffic. The CSM-S is then configured to perform NAT for the decrypted traffic to the selected destination server.

The administration network is separate from the client traffic networks and must reside in its own administration VLAN, which must be configured on both the CSM-S and SSL daughter card.

Figure B-1 Bridge Mode, No NAT Configuration Example



120262

The following addresses are configured on the CSM-S:

- Client clear text traffic—10.90.14.181:80
- Client SSL traffic—10.90.14.181:443
- Decrypted traffic from SSL daughter card—10.90.14.181:80
- Client VLAN 225 with IP address 10.90.14.245 for client communication
- Server VLAN 443 with IP address 10.90.14.245, and alias 10.90.14.242 for SSL daughter card communication
- Server VLAN 999 with IP address 172.16.1.2 to allow Administrative communication to reach the SSL daughter card
- Server VLAN 6 with IP address 192.168.6.2, and an alias 192.168.6.1 for real server communications.

The following address is configured on the SSL daughter card:

- 10.90.14.181:443 (This IP address is configured with the **secondary** keyword, which is a CSM-S and bridge mode requirement.)
- VLAN 443 with IP address 10.90.14.243 and a gateway of 10.90.14.1.
- VLAN 999 with IP address 172.16.1.3, a gateway of 172.16.1.1, and admin enabled.

Figure B-1 shows VLAN 225 and VLAN 443 in the same subnet and VLAN 6 in a separate subnet.

Add all the VLANs (listed above) to the VLAN database, and configure the IP address on the VLAN interface for VLAN 999, VLAN 225, and VLAN 6 on the MSFC.



Note

While VLAN 999 (172.16.1.1) and VLAN 225 (10.90.14.1) exist as Layer 3 interfaces on the MSFC, VLAN 443 and VLAN 6 (192.168.6.1) exist as VLANs in the VLAN database, but they do not have corresponding Layer 3 interfaces on the MSFC.

This example shows how to create the Layer 2 and Layer 3 VLANs on the switch MSFC:

```
Cat6k# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Cat6k(config)# vlan 6
Cat6k(config-vlan)# name Server_communications
Cat6k(config-if)# vlan 225
Cat6k(config-vlan)# name Client_communications
Cat6k(config-vlan)# interface Vlan225
Cat6k(config-if)# ip address 10.90.14.1 255.255.255.0
Cat6k(config-if)# no shutdown
Cat6k(config-if)# vlan 443
Cat6k(config-vlan)# name SSL-DC_communications
Cat6k(config-if)# vlan 999
Cat6k(config-vlan)# name SSL-DC_administrative
Cat6k(config-vlan)# interface Vlan999
Cat6k(config-if)# ip address 172.16.1.1 255.255.255.0
Cat6k(config-if)# no shutdown
```

This example shows how to create the client and server VLANs on the CSM-S installed in slot number 5:

```
Cat6k# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Cat6k(config-module-csm)# module ContentSwitchingModule 5
Cat6k(config-module-csm)# vlan 999 server
Cat6k(config-slb-vlan-server)# ip address 172.16.1.2 255.255.255.0
Cat6k(config-slb-vlan-server)# vlan 225 client
Cat6k(config-slb-vlan-client)# description Client Traffic
Cat6k(config-slb-vlan-client)# ip address 10.90.14.245 255.255.255.0
Cat6k(config-slb-vlan-client)# gateway 10.90.14.1
Cat6k(config-slb-vlan-client)# !
Cat6k(config-slb-vlan-client)# vlan 6 server
Cat6k(config-slb-vlan-server)# description Server Traffic
Cat6k(config-slb-vlan-server)# ip address 192.168.6.2 255.255.255.0
Cat6k(config-slb-vlan-server)# alias 192.168.6.1 255.255.255.0
Cat6k(config-slb-vlan-server)# !
Cat6k(config-slb-vlan-server)# vlan 443 server
Cat6k(config-slb-vlan-server)# ip address 10.90.14.245 255.255.255.0
```

This example shows how to create real servers with names.

```
Cat6k(config-slb-vlan-server)# real LINUX
Cat6k(config-slb-module-real)# address 192.168.6.10
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real WIN2K
Cat6k(config-slb-module-real)# address 192.168.6.20
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real SUN
Cat6k(config-slb-module-real)# address 192.168.6.30
Cat6k(config-slb-module-real)# inservice
```

This example shows how to create the server farm of web servers (configured with server NAT) and the server farm of the SSL daughter card (configured with no server NAT and local):

```
Cat6k(config-slb-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-sfarm)# no nat server
Cat6k(config-slb-sfarm)# real 10.90.14.243 local
```



Note

The keyword `local` is required to configure the CSM-S to send traffic to this real over the local VLAN to the SSL daughter card.

```
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# serverfarm WEB
Cat6k(config-slb-sfarm)# real name LINUX
```

```

Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name WIN2K
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name SUN
Cat6k(config-slb-real)# inservice

```

This example shows how to configure the two virtual servers to direct HTTPS traffic to the SSL daughter card for off loading and to load balance HTTP to web servers. In this example, the web servers are receiving traffic to port 80 only, either directly from the clients or as decrypted traffic from the SSL daughter cards (since no port translation is configured).

```

Cat6k(config-slb-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-sfarm)# no nat server
Cat6k(config-slb-sfarm)# real 10.90.14.243 local
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# serverfarm WEB
Cat6k(config-slb-sfarm)# real name LINUX
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name WIN2K
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name SUN
Cat6k(config-slb-real)# inservice

```

This example shows how to configure the two virtual servers to direct HTTP traffic to the SSL daughter card for off loading and to load balance HTTP to web servers. In this example, the web servers are receiving traffic to port 80 only, either directly from the clients or as decrypted traffic from the SSL daughter cards (since no port translation is configured).

```

Cat6k(config-slb-real)# vserver SSLTERMINATION
Cat6k(config-slb-vserver)# virtual 10.90.14.181 tcp https
Cat6k(config-slb-vserver)# vlan 225
Cat6k(config-slb-vserver)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice
Cat6k(config-slb-vserver)# vserver WEBSERVERS
Cat6k(config-slb-vserver)# virtual 10.90.14.181 tcp www
Cat6k(config-slb-vserver)# serverfarm WEB
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice
Cat6k(config-slb-vserver)# exit
Cat6k(config-module-csm)# exit
Cat6k(config)# exit

```

This example shows how to configure the administration VLAN on the SSL daughter card to communicate over the VLAN 999:

```

SSL-DC# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SSL-DC(config)# ssl-proxy vlan 999
SSL-DC(config-vlan)# ipaddr 172.16.1.3 255.255.255.0
SSL-DC(config-vlan)# gateway 172.16.1.1
SSL-DC(config-vlan)# admin

```

Next the VLAN 443 is configured to allow communication with clients for off loading client SSL connections:

```

SSL-DC(config-vlan)# ssl-proxy vlan 443
SSL-DC(config-vlan)# ipaddr 10.90.14.243 255.255.255.0
SSL-DC(config-vlan)# gateway 10.90.14.1

```

To complete the configuration, enter the **ssl-proxy service** command to create a new service on the SSL daughter card (**sslterm**). This example shows how to configure a virtual IP address that matches the virtual server created on the CSM-S. (This virtual IP address is configured with the **secondary** keyword

so that the SSL daughter card does not reply to ARP requests for this IP address. This configuration is also a requirement for bridging network designs) The service is configured to send decrypted traffic back to the CSM-S without performing NAT.

```
SSL-DC(config-vlan)# ssl-proxy service sslterm
SSL-DC(config-ssl-proxy)# virtual ipaddr 10.90.14.181 protocol tcp port 443 secondary
SSL-DC(config-ssl-proxy)# server ipaddr 10.90.14.245 protocol tcp port 80
SSL-DC(config-ssl-proxy)# no nat server
SSL-DC(config-ssl-proxy)# certificate rsa general-purposetrustpoint certs-key
```

```
*Aug 19 20:52:11.487: %STE-6-PKI_SERVICE_CERT_INSTALL: Proxy: sslterm, Trustpoint:
t: certs-key, Key: RSAKEY, Serial#: 1A65, Index: 2
```

```
*Aug 19 20:52:11.487: %STE-6-PKI_CA_CERT_INSTALL: Root, Subject Name: CN = Thawte
Test CA Root, OU = TEST TEST TEST, O = Thawte Certification, ST = FOR TESTING
PURPOSES ONLY, C = ZA, Serial#: 00, Index: 3
```

```
SSL-DC(config-ssl-proxy)# inservice
```

```
*Aug 19 20:52:11.515: %STE-5-UPDOWN: ssl-proxy service sslterm changed state to
UP
```

These examples show the output of the various **show** commands on the MSFC and CSM:

```
Cat6k# show module csm 5 vlan detail
vlan  IP address      IP mask      type
-----
6     192.168.6.2        255.255.255.0  SERVER
      Description: Server Traffic
      ALIASES
      IP address      IP mask
      -----
      192.168.6.1      255.255.255.0
225   10.90.14.245       255.255.255.0  CLIENT
      Description: Client Traffic
      GATEWAYS
      10.90.14.129
443   10.90.14.245       255.255.255.0  SERVER
999   172.16.1.2         255.255.255.0  SERVER
```

```
Cat6k# show module csm 5 real
```

```
real          server farm      weight  state          conns/hits
-----
10.90.14.243  SSLOFFLOADERS   8       OPERATIONAL    0
LINUX         WEB              8       OPERATIONAL    0
WIN2K         WEB              8       OPERATIONAL    0
SUN           WEB              8       OPERATIONAL    0
```

```
Cat6k# show module csm 5 vserver detail
```

```
SSLTERMINATION, type = SLB, state = OPERATIONAL, v_index = 12
virtual = 10.90.14.181/32:443 bidir, TCP, service = NONE, advertise = FALSE
idle = 3600, replicate csrp = none, vlan = 225, pending = 30, layer 4
max parse len = 2000, persist rebalance = TRUE
ssl sticky offset = 0, length = 32
conns = 1, total conns = 4
Default policy:
  server farm = SSLOFFLOADERS, backup = <not assigned>
  sticky: timer = 0, subnet = 0.0.0.0, group id = 0
Policy          Tot matches  Client pkts  Server pkts
-----
(default)       4            32           21
```

```

WEBSERVERS, type = SLB, state = OPERATIONAL, v_index = 13
  virtual = 10.90.14.181/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 1, total conns = 7
Default policy:
  server farm = WEB, backup = <not assigned>
  sticky: timer = 0, subnet = 0.0.0.0, group id = 0
Policy          Tot matches  Client pkts  Server pkts
-----
(default)       7             45           35

```

These examples show the output of the various **show** commands on the SSL daughter card:

```

SSL-DC# show ssl-proxy service sslterm
Service id: 1, bound_service_id: 257
Virtual IP: 10.90.14.181, port: 443 (secondary configured)
Server IP: 10.90.14.245, port: 80
rsa-general-purpose certificate trustpoint: certs-key
Certificate chain for new connections:
Certificate:
  Key Label: RSAKEY, 1024-bit, not exportable
  Key Timestamp: 02:03:11 UTC Aug 19 2004
  Serial Number: 1A65
Root CA Certificate:
  Serial Number: 00
Certificate chain complete
Admin Status: up
Operation Status: up

SSL-DC# show ssl-proxy stats
TCP Statistics:
  Conns initiated      : 4           Conns accepted      : 4
  Conns established   : 8           Conns dropped       : 4
  Conns Allocated     : 4           Conns Deallocated  : 4
  Conns closed        : 8           SYN timeouts       : 0
  Idle timeouts       : 0           Total pkts sent    : 43
  Data packets sent   : 19          Data bytes sent    : 5875
  Total Pkts rcvd     : 48           Pkts rcvd in seq  : 21
  Bytes rcvd in seq   : 3264

SSL Statistics:
  conns attempted     : 4           conns completed    : 4
  full handshakes     : 2           resumed handshakes : 2
  active conns        : 0           active sessions    : 0
  renegs attempted   : 0           conns in reneg    : 0
  handshake failures  : 0           data failures      : 0
  fatal alerts rcvd   : 0           fatal alerts sent  : 0
  no-cipher alerts    : 0           ver mismatch alerts: 0
  no-compress alerts  : 0           bad macs received  : 0
  pad errors          : 0           session fails      : 0

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        : 8           Flow Deletes       : 8
  Conn Id allocs      : 4           Conn Id deallocs   : 4
  Tagged Pkts Drops   : 0           Non-Tagg Pkts Drops: 0
  Add ipcs            : 3           Delete ipcs        : 0
  Disable ipcs        : 2           Enable ipcs        : 0

```

```

Unsolicited ipcs      : 127
IOS Broadcast Pkts    : 613
IOS Multicast Pkts    : 0
IOS Congest Drops     : 0
Duplicate Add ipcs    : 0
IOS Unicast Pkts     : 1110
IOS Total Pkts        : 1723
SYN Discards          : 0

```

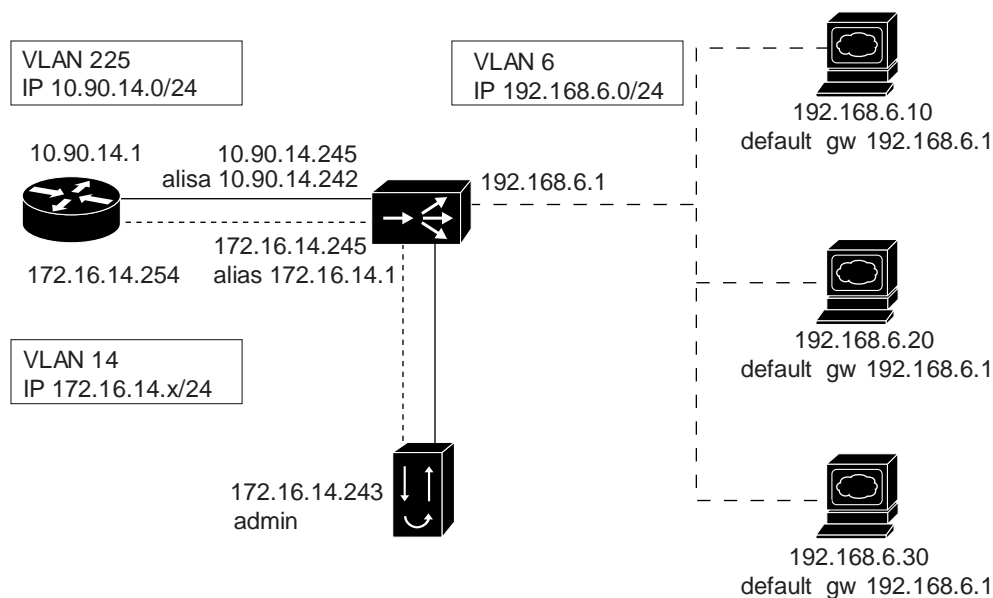
CSM-S Configuration Example (Router Mode, Server NAT)

This section describes a CSM-S configuration which allows a client to load balance HTTP to three web servers (IP addresses 192.168.6.10, 192.168.6.20, and 192.168.6.30) and offload HTTPS then load balance to the same three web servers.

In this example, the CSM-S client VLAN is on a public network, the server VLAN for the SSL daughter card is in the a private IP subnet, and the web servers are in a different private IP network and reside in a separate VLAN. (See [Figure B-2](#).)

The CSM-S is configured to perform the default server NAT operations to direct encrypted client traffic to the SSL daughter card. The SSL daughter card is also configured to perform server NAT operations when sending decrypted traffic back to the CSM-S. The CSM-S is then configured to perform another NAT on the decrypted traffic to the selected destination server.

Figure B-2 Configuration Example—Router Mode, Server NAT



The following addresses are configured on the CSM-S virtual servers:

- Client clear text traffic—10.90.14.182:80
- Client SSL traffic—10.90.14.182:443
- Decrypted traffic from SSL daughter card—10.90.14.182:80
- Client VLAN 225 with IP address 10.90.14.245 for client communication
- Server VLAN 14 with IP address 172.16.14.245, and alias 172.16.14.1 for SSL daughter card communication

- Server VLAN 6 with IP address 192.168.6.2, and an alias 192.168.6.1 for real server communications

The following address is configured on the SSL daughter card:

- 172.16.14.182:443 (this IP address is configured with the **secondary** keyword a CSM-S requirement)
- VLAN 14 with IP address 172.16.14.243, a route for client traffic to the CSM-S VLAN interface, and a gateway of 172.16.14.254 for routing administrative traffic.

Figure B-2, shows VLAN 225, VLAN 14 and VLAN 6 are each in separate subnets.

Add all the VLANs (listed above) to the VLAN database, and configure the IP address on the VLAN interface for VLAN 14 and VLAN 225 on the MSFC.



Note

VLAN 225 (10.90.14.1) exists as a Layer 3 interface on the MSFC to route Client traffic to the CSM-S. VLAN 14 (172.16.1.254) is also configured on the MSFC to allow administrative traffic to be routed to the SSL daughter card. VLAN 14 (172.16.14.1) is configured on the CSM-S to send and received SSL traffic to/from the SSL daughter card. VLAN 6 (192.168.6.1) exists only as a VLAN in the VLAN database and as CSM-S and SSL daughter card VLANs, but it does not have corresponding Layer 3 interfaces on the MSFC.

This example creates the Layer 2 and Layer 3 VLANs on the switch MSFC:

```
Cat6k# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Cat6k(config)# vlan 6
Cat6k(config-vlan)# name Server_communications
Cat6k(config)# vlan 14
Cat6k(config-vlan)# name SSL-DC_communications
Cat6k(config-vlan)# interface Vlan14
Cat6k(config-if)# ip address 172.16.14.254 255.255.255.0
Cat6k(config-if)# no shutdown
Cat6k(config-if)# vlan 225
Cat6k(config-vlan)# name Client_communications
Cat6k(config-vlan)# interface Vlan225
Cat6k(config-if)# ip address 10.90.14.1 255.255.255.0
Cat6k(config-if)# no shutdown
```

This example shows how to create the client and server VLANs on the CSM installed in slot number 5:

```
Cat6k# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Cat6k(config)# module ContentSwitchingModule 5
Cat6k(config-module-csm)# vlan 225 client
Cat6k(config-slb-vlan-client)# description Client Traffic
Cat6k(config-slb-vlan-client)# ip address 10.90.14.245 255.255.255.0
Cat6k(config-slb-vlan-client)# gateway 10.90.14.1
Cat6k(config-slb-vlan-client)# vlan 6 server
Cat6k(config-slb-vlan-server)# description Server Traffic
Cat6k(config-slb-vlan-server)# ip address 192.168.6.2 255.255.255.0
Cat6k(config-slb-vlan-server)# alias 192.168.6.1 255.255.255.0
Cat6k(config-slb-vlan-server)# vlan 14 server
Cat6k(config-slb-vlan-server)# ip address 172.16.14.245 255.255.255.0
Cat6k(config-slb-vlan-server)# alias 172.16.14.1 255.255.255.0
```

This example shows how to create real servers with names.

```
Cat6k(config-slb-vlan-server)# real LINUX
Cat6k(config-slb-module-real)# address 192.168.6.10
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real WIN2K
```

```

Cat6k(config-slb-module-real)# address 192.168.6.20
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real SUN
Cat6k(config-slb-module-real)# address 192.168.6.30
Cat6k(config-slb-module-real)# inservice

```

This example shows how to create the server farm of web servers (configured with server NAT) and the server farm of the SSL daughter card (configured with server NAT and local):

```

Cat6k(config-slb-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-sfarm)# nat server
Cat6k(config-slb-sfarm)# no nat client
Cat6k(config-slb-sfarm)# real 172.16.14.182 local

```

**Note**

The keyword **local** is required to configure the CSM-S to send traffic to this real over the local VLAN to the SSL daughter card.

```

Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# serverfarm WEB
Cat6k(config-slb-sfarm)# nat server
Cat6k(config-slb-sfarm)# no nat client
Cat6k(config-slb-sfarm)# real name LINUX
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name WIN2K
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name SUN
Cat6k(config-slb-real)# inservice

```

This example shows how to configure the two virtual servers. In this example, the web servers receive requests to port 80 directly from the clients. HTTPS traffic is received on port 443 and sent to the SSL daughter card for decryption. Upon decryption, the HTTP traffic is sent to the public HTTP virtual for load balancing:

```

Cat6k(config-slb-real)# vserver SSLTERMINATION
Cat6k(config-slb-vserver)# virtual 10.90.14.182 tcp https
Cat6k(config-slb-vserver)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice

Cat6k(config-slb-vserver)# vserver WEBSERVERS
Cat6k(config-slb-vserver)# virtual 10.90.14.182 tcp www
Cat6k(config-slb-vserver)# serverfarm WEB
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice

```

This example shows how to configure the SSL daughter card to communicate with the CSM-S over VLAN 14 for client traffic and administrative traffic:

```

SSL-DC# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SSL-DC(config)# ssl-proxy vlan 14
SSL-DC(config-vlan)# ipaddr 172.16.14.243 255.255.255.0
SSL-DC(config-vlan)# gateway 172.16.14.254
SSL-DC(config-vlan)# route 10.90.14.0 255.255.255.0 gateway 172.16.14.1
SSL-DC(config-vlan)# admin

```

**Note**

The **gateway** command is required for routing administrative communication. The router defined only effects traffic destined for the VLAN IP address (TELNET, SSH, and so on...)

The **route** statement is required to route traffic back to the CSM-S alias IP when the SSL daughter card is receiving encrypted traffic on one network and sending decrypted traffic to a different IP network.

**Note**

The administrative VLAN can be configured separately if required by adding a new VLAN and the appropriate IP address on both the CSM-S and SSL daughter card.

To complete the configuration, enter the **ssl-proxy service sslterm** command to create a new service on the SSL daughter card (**sslterm**). This example shows how to configure a virtual IP address that matches the virtual server created on the CSM-S. (This virtual IP address is configured with the **secondary** keyword so that the SSL daughter card does not reply to ARP requests for this IP address. This configuration is also a requirement for bridging network designs.) The service is configured to send decrypted traffic back to the CSM-S while performing NAT on the destination address:

```
SSL-DC(config-vlan)# ssl-proxy service sslterm
SSL-DC(config-ssl-proxy)# virtual 172.16.14.182 protocol tcp port 443 secondary
SSL-DC(config-ssl-proxy)# server ipaddr 10.90.14.182 protocol tcp port 80
SSL-DC(config-ssl-proxy)# certificate rsa general-purpose trustpoint certs-key

*Aug 22 14:44:47.395: %STE-6-PKI_SERVICE_CERT_INSTALL: Proxy: sslterm, Trustpoint:
certs-key, Key: RSAKEY, Serial#: 1A65, Index: 6
*Aug 22 14:44:47.395: %STE-6-PKI_CA_CERT_INSTALL: Root, Subject Name: CN = Thawte
Test CA Root, OU = TEST TEST TEST, O = Thawte Certification, ST = FOR TESTING
PURPOSES ONLY, C = ZA, Serial#: 00, Index: 7

SSL-DC(config-ssl-proxy)# inservice

*Aug 22 14:44:47.423: %STE-5-UPDOWN: ssl-proxy service sslterm changed state to UP

Cat6k # configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Cat6k(config)# interface Vlan14
Cat6k(config-if)# ip address 172.16.14.254 255.255.255.0
Cat6k(config-if)# no shutdown
```

These examples show the output of the various **show** commands on the MSFC and CSM:

```
cat6k# show mod csm 5 vlan detail
vlan  IP address      IP mask      type
-----
6      192.168.6.2        255.255.255.0  SERVER
  Description: Server Traffic
  ALIASES
  IP address      IP mask
  -----
  192.168.6.1      255.255.255.0
14      172.16.14.245     255.255.255.0  SERVER
  ALIASES
  IP address      IP mask
  -----
  172.16.14.1      255.255.255.0
225    10.90.14.245     255.255.255.0  CLIENT
  Description: Client Traffic
  GATEWAYS
  10.90.14.129

Cat6k# show mod csm 5 real
real      server farm      weight  state      conns/hits
-----
172.16.14.182  SSLOFFLOADERS  8      OPERATIONAL  0
LINUX      WEB              8      OPERATIONAL  0
WIN2K      WEB              8      OPERATIONAL  0
SUN        WEB              8      OPERATIONAL  0
```

```

Cat6k# show mod csm 5 vserver detail
SSLTERMINATION, type = SLB, state = OPERATIONAL, v_index = 20
  virtual = 10.90.14.182/32:443 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 8
  Default policy:
    server farm = SSLOFFLOADERS, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  -----
  (default)       8             75           46

WEBSERVERS, type = SLB, state = OPERATIONAL, v_index = 21
  virtual = 10.90.14.182/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 11
  Default policy:
    server farm = WEB, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  -----
  (default)       11            58           38

```

These examples show the output of the various **show** commands on the SSL daughter card:

```

SSL-DC# show ssl-proxy service sslterm
Service id: 4, bound_service_id: 260
Virtual IP: 172.16.14.182, port: 443 (secondary configured)
Server IP: 10.90.14.182, port: 80
rsa-general-purpose certificate trustpoint: certs-key
Certificate chain in graceful rollover, being renewed:
Certificate:
  Key Label: RSAKEY, 1024-bit, not exportable
  Key Timestamp: 02:03:11 UTC Aug 19 2004
  Serial Number: 1A65
Root CA Certificate:
  Serial Number: 00
Service certificate in graceful rollover
Admin Status: up
Operation Status: up

SSL-DC# show ssl-proxy stats
TCP Statistics:
  Conns initiated      : 12           Conns accepted      : 12
  Conns established    : 24           Conns dropped       : 12
  Conns Allocated     : 12           Conns Deallocated  : 12
  Conns closed        : 24           SYN timeouts        : 0
  Idle timeouts       : 0           Total pkts sent     : 129
  Data packets sent   : 59           Data bytes sent     : 23001
  Total Pkts rcvd     : 146          Pkts rcvd in seq   : 57
  Bytes rcvd in seq   : 9826

SSL Statistics:
  conns attempted     : 12           conns completed     : 12
  full handshakes     : 10           resumed handshakes  : 2
  active conns        : 0             active sessions     : 0
  renegs attempted    : 0             conns in reneg      : 0
  handshake failures  : 0             data failures       : 0
  fatal alerts rcvd   : 0             fatal alerts sent    : 0
  no-cipher alerts    : 0             ver mismatch alerts : 0

```

```

no-compress alerts : 0          bad macs received : 0
pad errors         : 0          session fails     : 0

FDU Statistics:
IP Frag Drops     : 0          IP Version Drops  : 0
IP Addr Discards  : 0          Serv_Id Drops     : 2
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      : 24         Flow Deletes      : 24
Conn Id allocs    : 12         Conn Id deallocs  : 12
Tagged Pkts Drops : 0          Non-Tagg Pkts Drops : 0
Add ipcs          : 7          Delete ipcs       : 0
Disable ipcs      : 6          Enable ipcs       : 0
Unsolicited ipcs  : 3579       Duplicate Add ipcs : 0
IOS Broadcast Pkts : 17881      IOS Unicast Pkts  : 31780
IOS Multicast Pkts : 0          IOS Total Pkts    : 49661
IOS Congest Drops : 0          SYN Discards      : 0

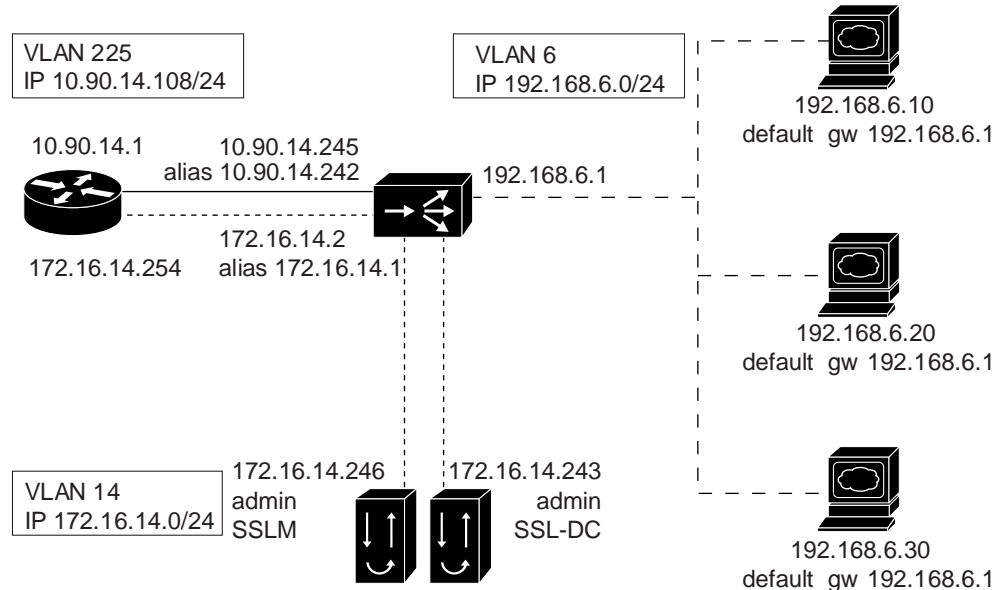
```

CSM-S and SSLM Configuration Example (Router Mode, Server NAT)

This section scales the previous CSM-S configuration [Appendix B, “CSM-S Configuration Example \(Router Mode, Server NAT\)”](#) by adding a SSL Services Module (SSLM) to the design. The SSLM is added using the same VLAN and IP network as the SSL daughter card. The CMS-S will use weighted round robin to load balance traffic between the SSLM and SSL-DC. Since the SSLM is approximately three times faster than the SSL daughter card weighted round robin is needed to spread the traffic across SSL off loaders according to the performance of the SSL off loader. The CSM-S applies SSL sticky to the client connections to ensure the same SSL session continue to use the same SSL off loader for the duration of the SSL session. In this example the duration is thirty minutes.

In this example, the SSLM is added to the previous CSM-S configuration. The SSLM will accept client connections on the same network as the SSL daughter card. (See [Figure B-3.](#))

Figure B-3 Configuration Example—CSM-S and SSLM Router Mode, Server NAT



The following address is configured on the SSLM:

- ssl-proxy service virtual of 172.16.14.10:443 and a server IP of 10.90.14.182:80
- VLAN 14 with IP address 172.16.14.246, a route for client traffic to the CSM-S VLAN interface, and a gateway of 172.16.14.254 for routing administrative traffic.

Along with the SSLM configuration the MSFC must be configured to allow VLAN 14 traffic to pass to the SSLM.

This example creates the Layer 2 and Layer 3 VLANs on the Cat6k MSFC:

```
Cat6k# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Cat6k(config)# ssl-proxy module 6 allowed-vlan 14
```

This example shows how to add the SSLM to the CSM-S.

```
Cat6k(config-slbf-vlan-server)# real SSLM
Cat6k(config-slbf-module-real)# address 172.16.14.10
Cat6k(config-slbf-module-real)# inservice
Cat6k(config-slbf-module-real)# inservice
```

This example shows how to add the SSLM real to the server farm of SSL off loaders and configure the weights for each real:

```
Cat6k(config-slbf-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slbf-sfarm)# real 172.16.14.182 local
Cat6k(config-slbf-sfarm)# weight 1
Cat6k(config-slbf-real)# inservice
Cat6k(config-slbf-sfarm)# real name SSLM
Cat6k(config-slbf-sfarm)# weight 3
Cat6k(config-slbf-real)# inservice
```

This example shows how to configure the CSM-S virtual server to apply SSL sticky for thirty minute sessions and use an SSL Session ID offset (SSL sticky sticky 10 ssl timeout 30).

```
Cat6k(config-slbf-real)# vservers SSLTERMINATION
Cat6k(config-slbf-vserver)# sticky 30 group 10
Cat6k(config-slbf-vserver)# ssl-sticky offset 20 length 6
```

This example shows how to configure the SSLM to communicate with the CSM-S over VLAN 14 for client traffic and administrative traffic:

```

SSLM# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SSLM(config)# ssl-proxy vlan 14
SSLM(config-vlan)# ipaddr 172.16.14.246 255.255.255.0
SSLM(config-vlan)# gateway 172.16.14.254
SSLM(config-vlan)# route 10.90.14.0 255.255.255.0 gateway 172.16.14.1
SSLM(config-vlan)# admin

```



Note

The **gateway** command is required for routing administrative communication. The router defined only effects traffic destined for the VLAN IP address (TELNET, SSH, and so on...)

The **route** statement is required to route traffic back to the CSM-S alias IP when the SSL daughter card is receiving encrypted traffic on one network and sending decrypted traffic to a different IP network.

To complete the configuration, enter the **ssl-proxy service** command to create a new service on the SSL daughter card (**sslterm**). This example shows how to configure a virtual IP address that matches the virtual server created on the CSM-S. (This virtual IP address is configured with the **secondary** keyword so that the SSL daughter card does not reply to ARP requests for this IP address. This is also a requirement for bridging network designs) The service is configured to send decrypted traffic back to the CSM-S while performing NAT on the destination address:

```

SSLM(config)# ssl-proxy service sslterm
SSLM(config-ssl-proxy)# virtual ipaddr 172.16.14.10 protocol tcp port 443
SSLM(config-ssl-proxy)# server ipaddr 10.90.14.182 protocol tcp port 80
SSLM(config-ssl-proxy)# certificate rsa general-purpose trustpoint certs-key

*Aug 24 01:40:17.581: %STE-6-PKI_SERVICE_CERT_INSTALL: Proxy: sslterm, Trustpoint:
certs-key, Key: RSAKEY, Serial#: 1C2B, Index: 2
*Aug 24 01:40:27.637: %STE-6-PKI_SERVICE_CERT_DELETE: Proxy: , Trustpoint: certs-key, Key:
RSAKEY, Serial#: 1C2B, Index: 0

SSLM(config-ssl-proxy)# inservice

*Aug 24 01:40:34.165: %STE-5-UPDOWN: ssl-proxy service sslterm changed state to UP

SSLM(config-ssl-proxy)# exit

Cat6k# show mod csm 5 real

real                server farm        weight  state           conns/hits
-----
SSLM                SSLOFFLOADERS     3       OPERATIONAL     0
172.16.14.182      SSLOFFLOADERS     1       OPERATIONAL     0
LINUX              WEB                8       OPERATIONAL     0
SUN                WEB                8       OPERATIONAL     0
WIN2K              WEB                8       OPERATIONAL     0

Cat6k# show mod csm 5 vserver detail
SSLTERMINATION, type = SLB, state = OPERATIONAL, v_index = 22
  virtual = 10.90.14.182/32:443 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 7
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 20, length = 6
  conns = 0, total conns = 12
  Default policy:
    server farm = SSLOFFLOADERS, backup = <not assigned>

```

```

    sticky: timer = 30, subnet = 0.0.0.0, group id = 10
Policy          Tot matches  Client pkts  Server pkts
-----
(default)      12           135         96

WEBSERVERS, type = SLB, state = OPERATIONAL, v_index = 23
virtual = 10.90.14.182/32:80 bidir, TCP, service = NONE, advertise = FALSE
idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
max parse len = 2000, persist rebalance = TRUE
ssl sticky offset = 0, length = 32
conns = 0, total conns = 12
Default policy:
    server farm = WEB, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
Policy          Tot matches  Client pkts  Server pkts
-----
(default)      12           75         67

Cat6k# show mod csm 5 sticky

```

group	sticky-data	real	timeout
10	ssl ADB70000:000DBCAF	172.16.14.182	1680
10	ssl A03F0000:00602F30	172.16.14.10	1596

These examples show the output of the various **show** commands on the SSLM:

```

SSLM# show ssl-proxy service sslterm
Service id: 1, bound_service_id: 257
Virtual IP: 172.16.14.10, port: 443
Server IP: 10.90.14.182, port: 80
rsa-general-purpose certificate trustpoint: certs-key
Certificate chain for new connections:
Certificate:
    Key Label: RSAKEY, 1024-bit, exportable
    Key Timestamp: 13:12:48 UTC Aug 23 2004
    Serial Number: 1C2B
Root CA Certificate:
    Serial Number: 00
Certificate chain complete
Admin Status: up
Operation Status: up

SSLM# show ssl-proxy stats
TCP Statistics:
    Conns initiated      : 14           Conns accepted      : 14
    Conns established    : 28           Conns dropped       : 10
    Conns Allocated     : 14           Conns Deallocated  : 14
    Conns closed        : 28           SYN timeouts       : 0
    Idle timeouts       : 0           Total pkts sent    : 181
    Data packets sent   : 90           Data bytes sent    : 47214
    Total Pkts rcvd    : 196           Pkts rcvd in seq  : 85
    Bytes rcvd in seq  : 32480

SSL Statistics:
    conns attempted     : 14           conns completed    : 14
    full handshakes     : 11           resumed handshakes : 3
    active conns        : 0           active sessions    : 0
    renegs attempted    : 0           conns in reneg     : 0
    handshake failures  : 0           data failures      : 0
    fatal alerts rcvd   : 0           fatal alerts sent  : 0
    no-cipher alerts    : 0           ver mismatch alerts: 0
    no-compress alerts  : 0           bad macs received  : 0
    pad errors          : 0           session fails      : 0

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	: 1
Hash Full Drops	: 0	Hash Alloc Fails	: 0
Flow Creates	: 28	Flow Deletes	: 28
Conn Id allocs	: 14	Conn Id deallocs	: 14
Tagged Pkts Drops	: 0	Non-Tagg Pkts Drops	: 0
Add ipcs	: 2	Delete ipcs	: 0
Disable ipcs	: 1	Enable ipcs	: 0
Unsolicited ipcs	: 0	Duplicate Add ipcs	: 0
IOS Broadcast Pkts	: 68857	IOS Unicast Pkts	: 293
IOS Multicast Pkts	: 0	IOS Total Pkts	: 69150
IOS Congest Drops	: 0	SYN Discards	: 0

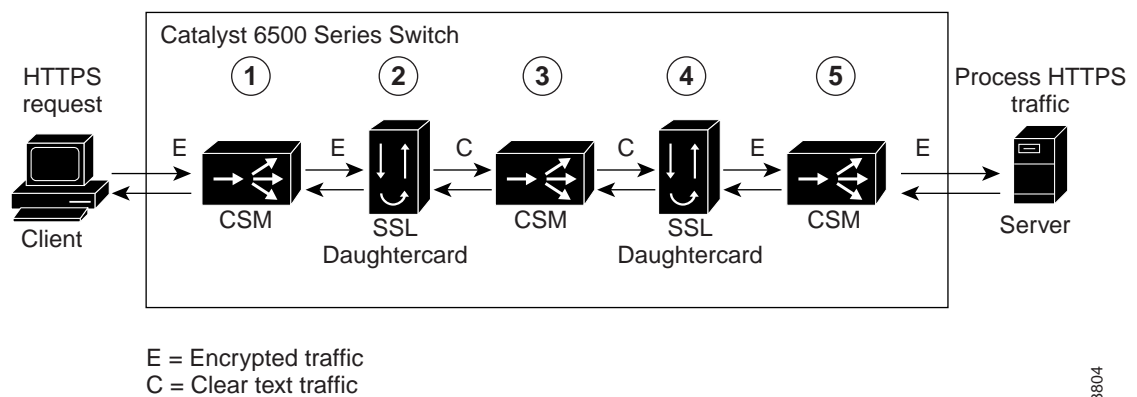
Integrated Secure Content-Switching Service Example

Configuring an integrated secure content-switching service (using a content switching module [CSM] as a server load balancer) with backend encryption has all the benefits of load-balancing and content switching, while securing data with full SSL coverage as it traverses paths of vulnerability.

As shown in [Figure B-4](#), an integrated secure content-switching service configuration involves five processing steps:

1. The CSM load balances the SSL traffic, based on either load-balancing rules or using the SSL sticky feature. See the [“Configuring Session Persistence \(Stickiness\)”](#) section on page 10-1 for information on configuring sticky connections, to an SSL daughter card.
2. The SSL daughter card terminates the SSL session, decrypts the SSL traffic into clear text traffic, and forwards the traffic back to the CSM.
3. The CSM content-switches the clear text traffic to the SSL daughter card again for encryption to SSL traffic.
4. The SSL daughter card forwards the encrypted SSL traffic to the CSM.
5. The CSM forwards the SSL traffic to the HTTPS server.

Figure B-4 Backend Encryption Example—Integrated Secure Content-Switching Service



113804

Configuring the CSM

This example shows how to configure the VLANs on the CSM. VLAN 24 is the VLAN through which client traffic arrives. VLAN 35 is the VLAN between the SSL daughter card and the CSM.

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# module ContentSwitchingModule 6
Router(config-module-csm)# vlan 24 client
Route(config-slb-vlan-client)# ip address 24.24.24.24 255.0.0.0
Route(config-slb-vlan-client)# vlan 35 server
Route(config-slb-vlan-server)# ip address 35.35.35.35 255.0.0.0
Route(config-slb-vlan-server)# route 36.0.0.0 255.0.0.0 gateway 35.200.200.3
```

This example shows how to configure the URL policy for Layer 7 parsing:

```
Route(config-slb-vlan-server)# map URL url
Router(config-slb-map-url)# match protocol http method GET url /*
```

This example shows how to create server farms:

```
Router(config-slb-map-url)# serverfarm SSLCARDS
Router(config-slb-sfarm)# real 35.200.200.101 local
Router(config-slb-real)# inservice
```

```
Router(config-slb-real)# serverfarm VLAN36REALS
Router(config-slb-sfarm)# real 36.200.200.14
Router(config-slb-real)# inservice
Router(config-slb-real)# real 36.200.200.5
Router(config-slb-real)# inservice
```

This example shows how to create the virtual servers:

```
Router(config-slb-real)# vserver LB-HTTP-SSLMODS
Router(config-slb-vserver)# virtual 35.35.35.25 tcp 81
Router(config-slb-vserver)# vlan 35
Router(config-slb-vserver)# slb-policy URL
Router(config-slb-vserver)# inservice

Router(config-slb-vserver)# vserver LB-SSL-SSLMODS
Router(config-slb-vserver)# virtual 24.24.24.25 tcp https
Router(config-slb-vserver)# serverfarm SSLCARDS
Router(config-slb-vserver)# inservice
```

This example shows how to display the status of the real servers and virtual servers:

```
Router# sh module contentSwitchingModule all reals
----- CSM in slot 6 -----
real                server farm      weight  state          conns/hits
-----
35.200.200.101     SSLCARDS         8       OPERATIONAL    0
36.200.200.14     VLAN36REALS     8       OPERATIONAL    0
36.200.200.5      VLAN36REALS     8       OPERATIONAL    0

Router# sh module contentSwitchingModule all vservers
----- CSM in slot 6 -----
vserver            type  prot  virtual                vlan state  conns
-----
LB-HTTP-SSLMODS  SLB   TCP   35.35.35.25/32:81      35  OPERATIONAL  0
LB-SSL-SSLMODS   SLB   TCP   24.24.24.25/32:443    ALL  OPERATIONAL  0
```

Configuring the SSL Daughter Card

This example shows how to create the VLAN between the SSL daughter card and the CSM:

```
ssl-proxy(config)# ssl-proxy vlan 35
ssl-proxy(config-vlan)# ipaddr 35.200.200.3 255.0.0.0
ssl-proxy(config-vlan)# gateway 35.200.200.100
ssl-proxy(config-vlan)# admin
```

This example shows how to configure a trusted certificate authority pool on the SSL daughter card:

```
ssl-proxy(config-vlan)# ssl-proxy pool ca net
ssl-proxy(config-ca-pool)# ca trustpoint keon-root
ssl-proxy(config-ca-pool)# ca trustpoint net-root
ssl-proxy(config-ca-pool)# ca trustpoint TP-1024-pcks12-root
```

This example shows how to configure a URL rewrite policy on the SSL daughter card:

```
ssl-proxy(config)# ssl-proxy policy url-rewrite frontend
ss(config-url-rewrite-policy)# url www.cisco.com clearport 80 sslport 443
ss(config-url-rewrite-policy)# url wwwin.cisco.com clearport 80 sslport 443
ss(config-url-rewrite-policy)# url wwwin.cisco.com clearport 81 sslport 443
```

This example shows how to configure the SSL server proxy that accepts client traffic coming through the CSM. This example also shows how to configure client authentication, SSL v2.0 forwarding, and URL rewrite policy.



Note

For SSL V2.0 connections, the SSL daughter card directly opens a connection to the configured server.

```
ssl-proxy(config-ca-pool)# ssl-proxy service frontend
ssl-proxy(config-ssl-proxy)# virtual ipaddr 35.200.200.101 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 35.35.35.25 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 35.200.200.14 protocol tcp port 443 sslv2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint TP-1024-pkcs12
ssl-proxy(config-ssl-proxy)# policy url-rewrite frontend
ssl-proxy(config-ssl-proxy)# trusted-ca net
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
```

This example shows how to configure the SSL client proxy that accepts clear text traffic from the CSM after the traffic completes Layer 7 parsing and decides the real server. This example also shows how to configure client certificates and a wildcard proxy.



Note

The gateway address (35.200.200.125) is the address through which the real servers (36.200.200.14 and 36.200.200.5) are reached.

```
ssl-proxy(config-ssl-proxy)# ssl-proxy service wildcard client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 0.0.0.0 0.0.0.0 protocol tcp port 81 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 35.200.200.125 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint client-cert
ssl-proxy(config-ssl-proxy)# no nat server
ssl-proxy(config-ssl-proxy)# trusted-ca net
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# ^Z
```

This example shows how to display the status of the SSL server proxy service:

```
ssl-proxy# show ssl-proxy service frontend
Service id: 2, bound_service_id: 258
Virtual IP: 35.200.200.101, port: 443
Server IP: 35.35.35.25, port: 81
SSLv2 IP: 35.200.200.14, port: 443
URL Rewrite Policy: frontend
Certificate authority pool: net
  CA pool complete
rsa-general-purpose certificate trustpoint: TP-1024-pkcs12
Certificate chain for new connections:
Certificate:
  Key Label: TP-1024-pkcs12, 1024-bit, not exportable
  Key Timestamp: 22:53:16 UTC Mar 14 2003
  Serial Number: 3C2CD2330001000000DB
Root CA Certificate:
  Serial Number: 313AD6510D25ABAE4626E96305511AC4
Certificate chain complete
Certificate authentication type: All attributes (like CRL) are verified
Admin Status: up
Operation Status: up
ssl-proxy#
```

This example shows how to display the status of the SSL client proxy service:

```
ssl-proxy# show ssl-proxy service wildcard
Service id: 267, bound_service_id: 11
Virtual IP: 0.0.0.0, port: 81 (secondary configured)
Virtual IP mask: 0.0.0.0
Server IP: 35.200.200.125, port: 443
Certificate authority pool: net
  CA pool complete
rsa-general-purpose certificate trustpoint: client-cert
Certificate chain for new connections:
Certificate:
  Key Label: client-cert, 1024-bit, not exportable
  Key Timestamp: 18:42:01 UTC Jul 14 2003
  Serial Number: 04
Root CA Certificate:
  Serial Number: 01
Certificate chain complete
Certificate authentication type: All attributes (like CRL) are verified
Admin Status: up
Operation Status: up
ssl-proxy#
```

Certificate Security Attribute-Based Access Control Examples

Certificate security attribute-based access control adds fields to the certificate that allow specifying an access control list (ACL) to create a certificate-based ACL.

For information on configuring certificate security attribute-based access control, refer to *Certificate Security Attribute-Based Access Control* at this URL:

<http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftcrtacl.htm>

This example shows that the SSL connections for the SSL proxy service “ssl-offload” are successful only if the subject-name of the client certificate contains the domain name **.cisco.com**:

```
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
```

```

ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca root-ca
ssl-proxy(config-ca-pool)# ca trustpoint root
ssl-proxy(config-ca-pool)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca trustpoint root
ssl-proxy(ca-trustpoint)# enrollment mode ra
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca certificate map acl 10
ssl-proxy(ca-certificate-map)# subject-name co .cisco.com
ssl-proxy(ca-certificate-map)# exit

```

This example shows that the certificate ACLs are configured so that SSL connections for the proxy service “ssl-offload” are successful for the following conditions:

- The subject-name of the client certificate contains **ste3-server.cisco.com** or **ste2-server.cisco.com**.
- The valid-start of the client certificate is greater than or equal to 30th Jul 2003.
- The expiration date of the client certificate is less than 1st Jan 2007.
- The issuer-name of the client certificate contains the string “certificate manager.”

```

ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca root-ca
ssl-proxy(config-ca-pool)# ca trustpoint root
ssl-proxy(config-ca-pool)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca trustpoint root
ssl-proxy(ca-trustpoint)# enrollment mode ra
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca certificate map acl 10
ssl-proxy(ca-certificate-map)# subject-name co ste3-server.cisco.com
ssl-proxy(ca-certificate-map)# valid-start ge Jul 30 2003 00:00:00 UTC
ssl-proxy(ca-certificate-map)# expires-on lt Jan 01 2007 00:00:00 UTC
ssl-proxy(ca-certificate-map)# issuer-name co certificate manager
ssl-proxy(ca-certificate-map)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca certificate map acl 20
ssl-proxy(ca-certificate-map)# subject-name co ste2-server.cisco.com
ssl-proxy(ca-certificate-map)# expires-on lt Jan 01 2007 00:00:00 UTC

```

```

ssl-proxy(ca-certificate-map)# issuer-name co certificate manager
ssl-proxy(ca-certificate-map)# valid-start ge Jul 30 2003 00:00:00 UTC
ssl-proxy(ca-certificate-map)# exit

```

This example shows that the server certificate is checked for the domain name in the certificate field. SSL initiation is successful only if the subject-name of the server certificate contains the domain name **.cisco.com**.

```

ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy service ssl-initiation client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# trusted-ca root
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy pool ca root-ca
ssl-proxy(config-ca-pool)# ca trustpoint root
ssl-proxy(config-ca-pool)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca trustpoint root
ssl-proxy(ca-trustpoint)# enrollment mode ra
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca certificate map acl 10
ssl-proxy(ca-certificate-map)# subject-name co .cisco.com
ssl-proxy(ca-certificate-map)# exit
ssl-proxy(config)#

```

HTTP Header Insertion Examples

The following examples show how to insert various HTTP headers and how to display header insertion statistics.

Example 1

This example shows how to insert custom headers, client IP address and TCP port number information, and a prefix string in HTTP requests sent to the server:

```

ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy policy http-header ssl-offload
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_with_1024_bit_key_size
ssl-proxy(config-http-header-policy)# client-ip-port
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80

```

```

ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)# exit

```

Custom headers and client IP address and TCP port number information are added to every HTTP request and are prefixed by the prefix string, as shown below:

```

SSL-OFFLOAD-Client-IP:7.100.100.1
SSL-OFFLOAD-Client-Port:59008
SSL-OFFLOAD-SOFTWARE VERSION :2.1(1)
SSL-OFFLOAD-module :SSL MODULE - CATALYST 6500
SSL-OFFLOAD-type-of-proxy:server_proxy_with_1024_bit_key_size

```

This example shows how to display header insertion information:

```

ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
  Session Headers Inserted :0          Custom Headers Inserted :2
  Session Id's Inserted    :0          Client Cert. Inserted   :0
  Client IP/Port Inserted  :2
  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 SSL statistics:

```

ssl-proxy# show ssl-proxy stats ssl
SSL Statistics:
  conns attempted          :2          conns completed         :2
  conns in handshake       :0          conns in data           :0
  renegs attempted        :0          conns in reneg         :0
  active sessions         :0          max handshake conns    :1
  rand bufs allocated      :0          cached rand buf miss   :0
  current device q len:0   :0          max device q len       :2
  sslv2 forwards          :0          cert reqs processed    :0
  fatal alerts rcvd       :0          fatal alerts sent      :0
  stale packet drops      :0          service_id discards    :0
  session reuses          :0

```

```

SSL3 Statistics:
  full handshakes         :0          resumed handshakes     :0
  handshake failures      :0          data failures          :0
  bad macs received       :0          pad errors             :0
  conns established with cipher rsa-with-rc4-128-md5 :0
  conns established with cipher rsa-with-rc4-128-sha :0
  conns established with cipher rsa-with-des-cbc-sha :0
  conns established with cipher rsa-with-3des-edc-cbc-sha :0

```

```

TLS1 Statistics:
  full handshakes         :1          resumed handshakes     :1
  handshake failures      :0          data failures          :0
  bad macs received       :0          pad errors             :0
  conns established with cipher rsa-with-rc4-128-md5 :0
  conns established with cipher rsa-with-rc4-128-sha :2
  conns established with cipher rsa-with-des-cbc-sha :0
  conns established with cipher rsa-with-3des-edc-cbc-sha :0

```

Example 2

This example shows how to insert session headers and a prefix string. The full session headers are added to the HTTP request when the full SSL handshake occurs. However, only the session ID is inserted when the session resumes.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy policy http-header ssl-offload
ssl-proxy(config-http-header-policy)# session
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)# exit
```

For the full SSL handshake, the session headers, prefixed by the prefix string, are added to the HTTP request as shown below:

```
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
```

When the session resumes, only the session ID is inserted:

```
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
```

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 SSL statistics:

```
ssl-proxy# show ssl-proxy stats ssl
SSL Statistics:
  conns attempted      :2          conns completed        :2
  conns in handshake  :0          conns in data          :0
  renegs attempted    :0          conns in reneg         :0
  active sessions     :0          max handshake conns    :1
  rand bufs allocated :0          cached rand buf miss   :0
  current device q len:0          max device q len       :2
  sslv2 forwards      :0          cert reqs processed    :0
  fatal alerts rcvd   :0          fatal alerts sent      :0
  stale packet drops  :0          service_id discards    :0
  session reuses      :0
```

```

SSL3 Statistics:
  full handshakes      :0          resumed handshakes :0
  handshake failures  :0          data failures      :0
  bad macs received   :0          pad errors         :0
  conns established with cipher rsa-with-rc4-128-md5      :0
  conns established with cipher rsa-with-rc4-128-sha     :0
  conns established with cipher rsa-with-des-cbc-sha     :0
  conns established with cipher rsa-with-3des-ede-cbc-sha :0

TLS1 Statistics:
  full handshakes      :1          resumed handshakes :1
  handshake failures  :0          data failures      :0
  bad macs received   :0          pad errors         :0
  conns established with cipher rsa-with-rc4-128-md5     :0
  conns established with cipher rsa-with-rc4-128-sha     :2
  conns established with cipher rsa-with-des-cbc-sha     :0
  conns established with cipher rsa-with-3des-ede-cbc-sha :0

```

Example 3

This example shows how to insert the custom headers, the decoded client certificate fields, and the IP address and destination TCP port number of the client-side connection, prefixed by the prefix string. The complete decoded client certificate fields are inserted for the full SSL handshake. However, only the session ID is inserted when the SSL session resumes.

```

ssl-proxy# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
ssl-proxy(config)# ssl-proxy policy http-header ssl-offload
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_with_1024_bit_key_size
ssl-proxy(config-http-header-policy)# client-cert
ssl-proxy(config-http-header-policy)# client-ip-port
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)# exit

```

For the full SSL handshake, the custom headers, the decoded client certificate fields, the IP address and destination TCP port number of the client-side connection, prefixed by the prefix string, are added to the HTTP request, as shown below:

```

SSL-OFFLOAD-Client-IP:7.100.100.1
SSL-OFFLOAD-Client-Port:59011
SSL-OFFLOAD-Session-Id:0F:61:9C:F2:E5:98:70:9D:1B:C1:EA:1D:38:F5:A1:2B:00:00:0E:03:00:60:
2F:30:9C:2F:1D:7D:5A:82:30:F6
SSL-OFFLOAD-SOFTWARE VERSION :2.1(1)
SSL-OFFLOAD-module :SSL MODULE - CATALYST 6500
SSL-OFFLOAD-type-of-proxy:server_proxy_with_1024_bit_key_size
SSL-OFFLOAD-ClientCert-Valid:1
SSL-OFFLOAD-ClientCert-Error:none

```

```

SSL-OFFLOAD-ClientCert-Fingerprint:1B:11:0F:E8:20:3F:6C:23:12:9C:76:C0:C1:C2:CC:85
SSL-OFFLOAD-ClientCert-Subject-CN:a
SSL-OFFLOAD-ClientCert-Issuer-CN:Certificate Manager
SSL-OFFLOAD-ClientCert-Certificate-Version:3
SSL-OFFLOAD-ClientCert-Serial-Number:0F:E5
SSL-OFFLOAD-ClientCert-Data-Signature-Algorithm:sha1WithRSAEncryption
SSL-OFFLOAD-ClientCert-Subject:OID.1.2.840.113549.1.9.2 = ste2-server.cisco.com +
OID.2.5.4.5 = B0FFF22E, CN = a, O = Cisco
SSL-OFFLOAD-ClientCert-Issuer:CN = Certificate Manager, OU = HSS, O = Cisco, L = San Jose,
ST = California, C = US
SSL-OFFLOAD-ClientCert-Not-Before:22:29:26 UTC Jul 30 2003
SSL-OFFLOAD-ClientCert-Not-After:07:00:00 UTC Apr 27 2006
SSL-OFFLOAD-ClientCert-Public-Key-Algorithm:rsaEncryption
SSL-OFFLOAD-ClientCert-RSA-Public-Key-Size:1024 bit
SSL-OFFLOAD-ClientCert-RSA-Modulus-Size:1024 bit
SSL-OFFLOAD-ClientCert-RSA-Modulus:B3:32:3C:5E:C9:D1:CC:76:FF:81:F6:F7:97:58:91:4D:B2:0E:
C1:3A:7B:62:63:BD:5D:F6:5F:68:F0:7D:AC:C6:72:F5:72:46:7E:FD:38:D3:A2:E1:03:8B:EC:F7:C9:9A:
80:C7:37:DA:F3:BE:1F:F4:5B:59:BD:52:72:94:EE:46:F5:29:A4:B3:9B:2E:4C:69:D0:11:59:F7:68:3A:
D9:6E:ED:6D:54:4E:B5:A7:89:B9:45:9E:66:0B:90:0B:B1:BD:F4:C8:15:12:CD:85:13:B2:0B:FE:7E:8D:
F0:D7:4A:98:BB:08:88:6E:CC:49:60:37:22:74:4D:73:1E:96:58:91
SSL-OFFLOAD-ClientCert-RSA-Exponent:00:01:00:01
SSL-OFFLOAD-ClientCert-X509v3-Authority-Key-Identifier:keyid=EE:EF:5B:BD:4D:CD:F5:6B:60:
9D:CF:46:C2:EA:25:7B:22:A5:08:00
SSL-OFFLOAD-ClientCert-X509v3-Basic-Constraints:
SSL-OFFLOAD-ClientCert-Signature-Algorithm:sha1WithRSAEncryption
SSL-OFFLOAD-ClientCert-Signature:87:09:C1:F8:86:C1:15:C5:57:18:8E:B3:0D:62:E1:0F:6F:D4:9D:
75:DA:5D:53:E2:C6:0B:73:99:61:BE:B0:F6:19:83:F2:E5:48:1B:D2:6C:92:83:66:B3:63:A6:58:B4:5C:
0E:5D:1B:60:F9:86:AF:B3:93:07:77:16:74:4B:C5

```

This example shows how to display header insertion information:

```

ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
  Session Headers Inserted :0          Custom Headers Inserted :1
  Session Id's Inserted    :1          Client Cert. Inserted   :1
  Client IP/Port Inserted  :1
  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 SSL statistics:

```

ssl-proxy# show ssl-proxy stats ssl
SSL Statistics:
  conns attempted          :1          conns completed         :1
  conns in handshake       :0          conns in data           :0
  renegs attempted        :0          conns in renege         :0
  active sessions         :0          max handshake conns    :1
  rand bufs allocated      :0          cached rand buf miss   :0
  current device q len:0   max device q len       :2
  sslv2 forwards          :0          cert reqs processed    :1
  fatal alerts rcvd       :0          fatal alerts sent      :0
  stale packet drops      :0          service_id discards    :0
  session reuses          :0

SSL3 Statistics:
  full handshakes         :0          resumed handshakes     :0
  handshake failures      :0          data failures          :0
  bad macs received       :0          pad errors              :0
  conns established with cipher rsa-with-rc4-128-md5 :0
  conns established with cipher rsa-with-rc4-128-sha :0
  conns established with cipher rsa-with-des-cbc-sha :0
  conns established with cipher rsa-with-3des-edc-cbc-sha :0

```

```

TLS1 Statistics:
  full handshakes      :1          resumed handshakes :0
  handshake failures  :0          data failures       :0
  bad macs received   :0          pad errors           :0
  conns established with cipher rsa-with-rc4-128-md5      :0
  conns established with cipher rsa-with-rc4-128-sha      :0
  conns established with cipher rsa-with-des-cbc-sha      :0
  conns established with cipher rsa-with-3des-ede-cbc-sha :1

```

URL Rewrite Examples

These examples show how to configure URL rewrite depending on the desired outcome and assume the following proxy configuration:

```

ssl-proxy service frontend
virtual ipaddr 35.200.200.101 protocol tcp port 443 secondary
server ipaddr 35.200.200.14 protocol tcp port 80
certificate rsa general-purpose trustpoint TP-1024-pkcs12
policy url-rewrite test-url-rewrite
inservice
!

```

Example 1

This example shows how to configure a protocol rewrite (for example, HTTP to HTTPS) when the clear text port is a standard HTTP port 80. In this example, when the server sends the relocation string as `http://ssl-136.cisco.com/index2.html`, the SSL daughter card rewrites the string as `https://ssl-136.cisco.com/index2.html`.

To configure a protocol rewrite (HTTP to HTTPS), specify any of the following URL rewrite rules:

- ```
ssl-proxy policy url-rewrite test-url-rewrite
 url ssl-136.cisco.com
 !
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
  url ssl*
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
 url *com
 !
```

### Example 2

This example shows how to configure a protocol rewrite (for example, HTTP to HTTPS) when the clear text port is a nonstandard HTTP port. In this example, when the server sends the relocation string as `http://ssl-136.cisco.com:100/index2.html`, the SSL daughter card rewrites the string as `https://ssl-136.cisco.com/index2.html`.

To configure a protocol rewrite (HTTP to HTTPS) with a nonstandard clear text port, specify any of the following URL rewrite rules:

- ```
ssl-proxy policy url-rewrite test-url-rewrite
  url ssl-136.cisco.com clearport 100
!
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
 url ssl* clearport 100
!
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
  url *com clearport 100
!
```

Example 3

This example shows how to configure a protocol rewrite and SSL port rewrite when the clear text port is a standard HTTP port 80. In this example, when the server sends the relocation string as `http://ssl-136.cisco.com/index2.html`, the SSL daughter card rewrites the string as `https://ssl-136.cisco.com:445/index2.html`.

To configure a protocol rewrite (HTTP to HTTPS) with a nonstandard SSL text port, specify any of the following URL rewrite rules:

- ```
ssl-proxy policy url-rewrite test-url-rewrite
 url ssl-136.cisco.com sslport 445
!
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
  url ssl* sslport 445
!
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
 url *com sslport 445
!
```

## Example 4

This example shows how to configure a protocol rewrite and SSL port rewrite when the clear text port is nonstandard. In this example, when the server sends the relocation string as `http://ssl-136.cisco.com:100/index2.html`, the SSL daughter card rewrites the string as `https://ssl-136.cisco.com:445/index2.html`.

To configure a protocol rewrite and SSL port rewrite with a nonstandard clear text port, specify any of the following URL rewrite rules:

- ```
ssl-proxy policy url-rewrite test-url-rewrite
  url ssl-136.cisco.com clearport 100 sslport 445
!
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
 url ssl* clearport 100 sslport 445
!
```
- ```
ssl-proxy policy url-rewrite test-url-rewrite
  url *com clearport 100 sslport 445
!
```

This example displays the above URL rewrite policy:

```
ssl-proxy# show ssl-proxy policy url-rewrite test-url-rewrite
Rule URL                               Clearport SSLport
 1 *com                                 100           445
SSL proxy services using this policy:
    frontend
Usage count of this policy:1

ssl-proxy#
```