Examples of CSS SSL Configurations

This chapter describes the SSL flow process with the SSL module and includes example proxy configurations. Each configuration section includes a running-configuration example and an accompanying illustration.

This section covers:

- Processing of SSL Flows by the SSL Module
- SSL Transparent Proxy Configuration — One SSL Module
- SSL Transparent Proxy Configuration — Two SSL Modules
- SSL Transparent Proxy Configuration — HTTP and Back-End SSL Servers
- SSL Full Proxy Configuration — One SSL Module
- SSL Initiation Configurations

Processing of SSL Flows by the SSL Module

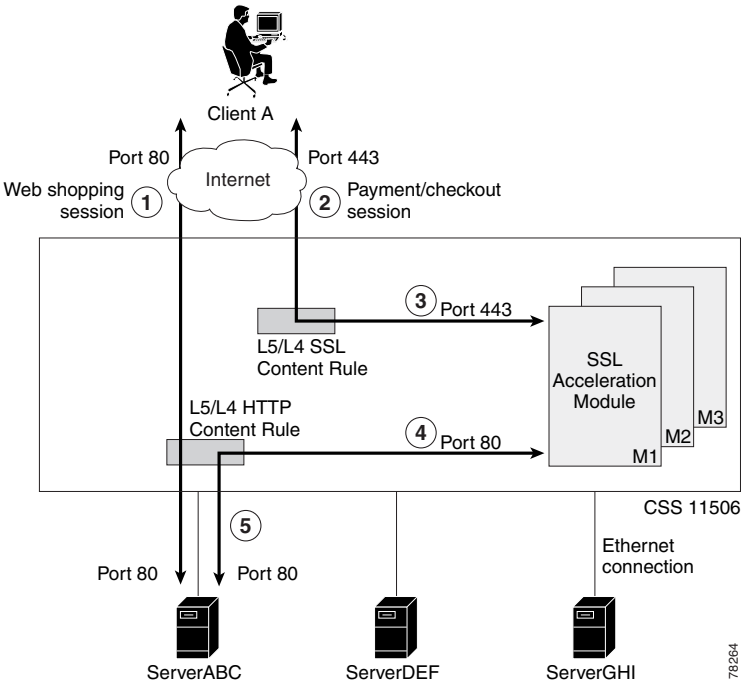
To terminate SSL flows, the SSL module functions as a proxy server, which means that it is the TCP endpoint for inbound SSL traffic. The SSL module maintains a separate TCP connection for each side of the communications, the client side and the server side. The proxy server can perform both TCP and SSL handshakes.

The following example is intended as an overview on the flow process; how the CSS and SSL module translate flows from HTTPS-to-HTTP for inbound packets and from HTTP-to-HTTPS for outbound packets.

Figure 8-1 illustrates a CSS with three SSL modules (M1, M2, and M3) configured to off load the SSL traffic from the back-end servers (ServerABC, ServerDEF, and ServerGHI). Figure 8-1 also shows the CSS maintaining a consistent stickiness between HTTP and SSL connections from the same client.

1. In a normal Web shopping-cart application, a transaction consists of multiple HTTP connections for shopping or browsing, and a few SSL connections for the final order placement and payment checkout sequence. The client must remain stuck to the same server that holds the customer's database information during the entire transaction. During the initial HTTP connections from a client to a server, the client is stuck to a server by using Layer 5 HTTP cookies or a URL content rule. At checkout, the client transitions to SSL connections.

Figure 8-1 CSS Configuration with Multiple SSL Modules



2. The client transmits the encrypted payment or order information through an SSL connection (TCP SYN received through destination port 443). In this example, when the client connection reaches the CSS, the CSS uses a Layer 5 SSL Session ID sticky content rule to load balance the SSL connection among the three SSL modules (M1, M2, and M3). When the inbound TCP SYN connection reaches the SSL module (the SSL server), it terminates the TCP connections from the client.
3. Once an SSL module is selected (for example, M1), the CSS forwards the SSL packet to that module. The Session ID is saved in the sticky table for subsequent SSL connections from the same client. Once this SSL flow is mapped, the CSS forwards all subsequent packets for this connection to SSL module M1. If there are additional SSL connections associated with this transaction (as determined by the SSL Session ID), the CSS also forwards and maps the packets to SSL module M1.
4. The SSL module terminates the SSL connection and decrypts the packet data. The SSL module then initiates an HTTP connection to a content rule configured on the CSS. The data in this HTTP connection is clear text.
5. The HTTP content rule uses the Layer 5 HTTP cookies or URL sticky content rule on this HTTP request. The cookie or URL string in this clear text HTTP request is used to locate the same server (ServerABC) as the one initially used by the non-SSL HTTP connection in the transactions (for example, online shopping). The CSS forwards the request to ServerABC and maps this flow. Once the flow is mapped, the return HTTP response from the server is sent to the same SSL module (M1) that sent the original request. The SSL module encrypts the response as an SSL packet (it translates flows from HTTP-to-HTTPS for outbound packets) and sends the packets back to the client through the correct SSL connection.

When the TCP connection is finished, the four flows (the two flows between the client and SSL module, and the two flows between the SSL module and the Server) are torn down.

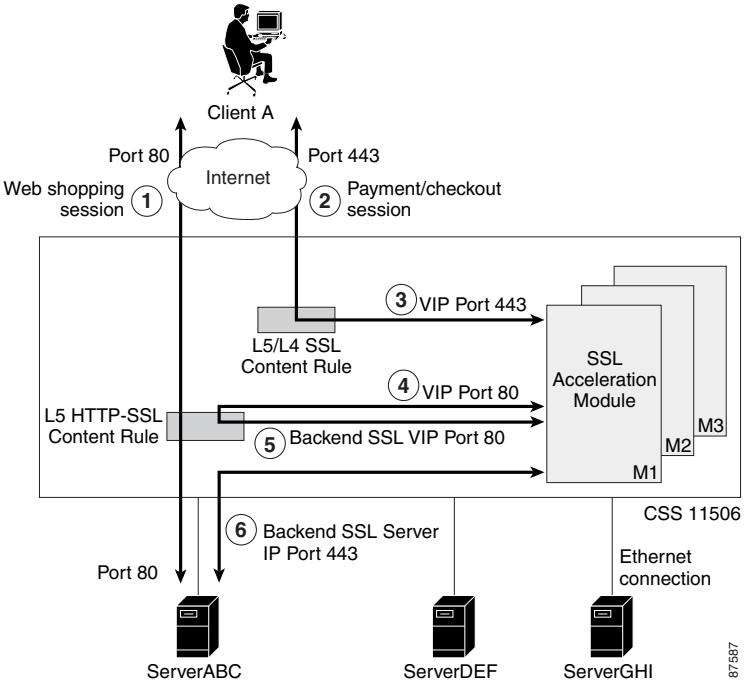
An entire SSL session can comprise Multiple TCP connections. For each of those connections, the same process takes place among the client, SSL module, and server. The SSL Session ID maintains the stickiness between the client and the SSL module and the cookie maintains the stickiness between the SSL module and the servers. In this way, stickiness can be maintained consistently through the entire web transaction.

Note By default, the SSL session cache for the SSL module can hold 10000 sessions. The cache size is the maximum number of SSL session IDs that can be stored in a dedicated session cache on an SSL module. If necessary for your SSL service, use the **session-cache-size** command to reconfigure the size of the SSL session ID cache for the SSL service.

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

When you configure a back-end SSL server on the CSS (Figure 8-2), flow processing from the client to CSS is the same as steps 1 through 4 in the previous example. However in step 4 shown in Figure 8-2, the SSL module initiates an HTTP connection to an HTTP-SSL content rule with services to a back-end SSL server.

Figure 8-2 CSS Configuration with a Back-End SSL Server



In step 5 shown in Figure 8-2, the CSS directs the clear text traffic back to the SSL module through an IP address that maps directly to a back-end SSL server. The SSL module terminates the clear text connection.

In step 6 of Figure 8-2, the SSL module re-encrypts the traffic and establishes an SSL connection to the back-end SSL server. The SSL module sends the traffic through the CSS to the selected back-end SSL server.

SSL Transparent Proxy Configuration — One SSL Module

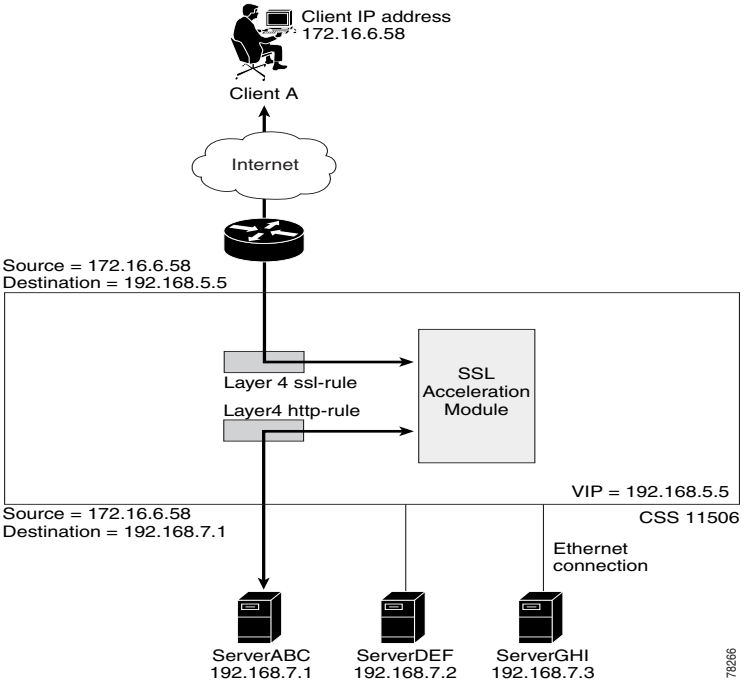
An SSL transparent proxy server is a proxy server that preserves the client's IP address as the source IP address for the back-end connection to the server. When you configure an SSL transparent proxy on the CSS, the CSS intercepts and redirects outbound client requests to an HTTP server on the network without changing the source IP address.

This section provides a simple configuration of an SSL transparent proxy between a client, a CSS with a single SSL module, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). Two content rules are used in this configuration, an SSL content rule and a HTTP content rule. The SSL content rule is for Layer 4 because there is only a single SSL module and there is no need to maintain client-to-server (SSL) stickiness. The use of a Layer 4 content rule in this configuration may improve CSS performance.

Figure 8-3 illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in Figure 8-3 shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rule (http-rule). These two VIP addresses do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP address for the clear-text content rule to place it in non-routable address space. In this example, instead of specifying a VIP address of 192.168.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 8-3 Transparent Proxy Configuration with a Single SSL Module




```
!************************** GLOBAL ***************************
  logging commands enable

  ssl associate dsakey dsakey dsakey.pem
  ssl associate dhparam dhparams dhparams.pem
  ssl associate rsakey rsakey rsakey.pem
  ssl associate cert rsacert rsacert.pem

  ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeeebabfcfagyezc5f /
```

```
!************************* CIRCUIT **************************
circuit VLAN1

  ip address 192.168.8.254 255.255.255.0

circuit VLAN2

  ip address 192.168.7.254 255.255.255.0

!********************** SSL PROXY LIST **********************
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.168.5.5
  ssl-server 111 port 443
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
  active

!************************* SERVICE **************************
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.168.7.1
  protocol tcp
  port 80
  keepalive type http
  active

service serverDEF
  ip address 192.168.7.2
  protocol tcp
  port 80
  keepalive type http
  active

service serverGHI
  ip address 192.168.7.3
  protocol tcp
  port 80
  keepalive type http
  active
```

```
!************************** OWNER ***************************
owner ap.com
content ssl-rule
    vip address 192.168.5.5
    protocol tcp
    port 443
    add service ssl_module1
    active

content http-rule
    vip address 192.168.5.5
    protocol tcp
    port 80
    add service serverABC
    add service serverDEF
    add service serverGHI
    advanced-balance cookies
    active
```

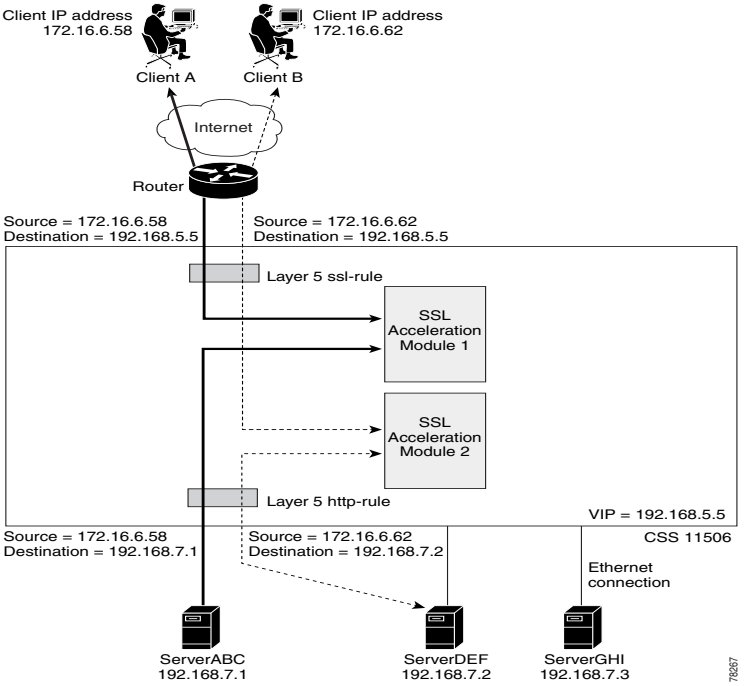
SSL Transparent Proxy Configuration — Two SSL Modules

This section provides an example configuration for an SSL transparent proxy between a client, a CSS with two SSL modules, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). A Layer 5 SSL sticky content rule is used in the configuration to maintain stickiness of the client to a particular SSL module. The Layer 5 SSL sticky content rule ensures SSL session ID reuse to eliminate the rehandshake process (which speeds up the SSL negotiation process) and to increase overall performance.

Figure 8-4 illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in Figure 8-4 shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rule (http-rule). These two VIP addresses do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP address for the clear-text content rule to place it in nonroutable address space. In this example, instead of specifying a VIP address of 192.168.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 8-4 Transparent Proxy Configuration with Two SSL Modules




```
!**************************** GLOBAL ****************************
  logging commands enable

  ssl associate dsakey dsakey dsakey.pem
  ssl associate rsakey rsakey rsakey.pem
  ssl associate cert rsacert rsacert.pem
  ssl associate dhparam dhparams dhparams.pem

  ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeeebabfcfagyezc5f /
```

```
!************************* CIRCUIT **************************
circuit VLAN1

  ip address 192.168.8.254 255.255.255.0

circuit VLAN2

  ip address 192.168.7.254 255.255.255.0

!********************** SSL PROXY LIST **********************
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.168.5.5
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
  ssl-server 111 port 443
  active

!************************* SERVICE **************************
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service ssl_module2
  type ssl-accel
  keepalive type none
  slot 6
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.168.7.1
  protocol tcp
  port 80
  keepalive type http
  active

service serverDEF
  ip address 192.168.7.2
  protocol tcp
  port 80
  keepalive type http
  active
```

```
service serverGHI
  ip address 192.14.7.3
  protocol tcp
  port 80
  keepalive type http
  active

!************************** OWNER ***************************
owner ap.com

content ssl-rule
    vip address 192.168.5.5
    protocol tcp
    port 443
    add service ssl_module1
    add service ssl_module2
    application ssl
    advanced-balance ssl
    active

content http-rule
    vip address 192.168.5.5
    protocol tcp
    port 80
    url "/*"
    add service serverABC
    add service serverDEF
    add service serverGHI
    advanced-balance cookies
    active
```

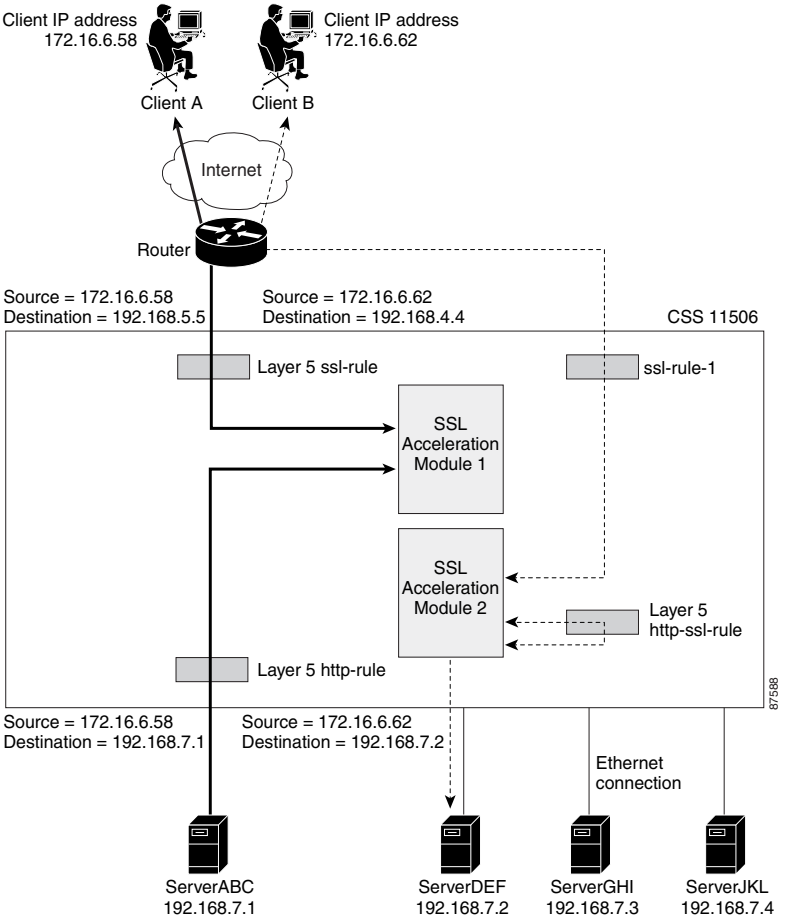
SSL Transparent Proxy Configuration — HTTP and Back-End SSL Servers

This section provides an example configuration for an SSL transparent proxy for two clients, a CSS with two SSL modules, two HTTP servers (ServerABC and ServerGHI), and two back-end SSL servers (ServerDEF and ServerJKL). This configuration is similar to the previous configuration. (See the "SSL Transparent Proxy Configuration — Two SSL Modules" section.) However, this example includes the configuration for a back-end SSL server.

In Figure 8-5, Client A's SSL connection has a destination address 192.168.5.5 that matches content rule ssl-rule. The CSS load balances the SSL connection to SSL module 1. The module terminates the connection, decrypts the data to clear text and initiates an HTTP connection to content rule http-rule. The CSS forwards the request to HTTP server ServerABC.

Client B's SSL connection has a destination address 192.28.4.4 that matches content rule ssl-rule-1. The CSS load balances the SSL connection to SSL module 2. The module terminates the connection, decrypts the data to clear text and initiates an HTTP connection to content rule http-ssl-rule. The CSS directs the clear text data back to SSL module 2. The module terminates the connection, re-encrypts the traffic, and establishes an SSL connection to SSL server ServerDEF.

Figure 8-5 SSL Transparent Proxy Configuration - HTTP and Back-End SSL Servers



The following configuration includes commands containing default values that do not appear in the running configuration. To identify these commands, they appear in italic.

```
!************************** GLOBAL ***************************
  logging commands enable

  ssl associate dsakey dsakey dsakey.pem
  ssl associate rsakey rsakey rsakey.pem
  ssl associate cert rsacert rsacert.pem
  ssl associate dhparam dhparams dhparams.pem

  ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeeebabfcfagyezc5f /
!************************** CIRCUIT **************************
circuit VLAN1

  ip address 192.168.8.254 255.255.255.0

circuit VLAN2

  ip address 192.168.7.254 255.255.255.0

!********************** SSL PROXY LIST ***********************
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.168.5.5
  ssl-server 111 port 443
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
  active

  ssl-server 2
  ssl-server 2 vip address 192.28.4.4
  ssl-server 2 port 443
  ssl-server 2 rsacert rsacert
  ssl-server 2 rsakey rsakey
  ssl-server 2 cipher rsa-with-rc4-128-md5 192.28.4.4 8080
  active

  backend-server 3
  backend-server 3 ip address 192.168.7.2
  backend-server 3 port 8080
  backend-server 3 server-ip 192.168.7.2
  backend-server 3 rsacert rsacert
  active
```

```
  backend-server 4
  backend-server 4 ip address 192.168.7.4
  backend-server 4 port 8080
  backend-server 4 server-ip 192.168.7.4
  backend-server 4 rsacert rsacert
  active

!************************* SERVICE ************************
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service ssl_module2
  type ssl-accel
  keepalive type none
  slot 6
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.168.7.1
  protocol tcp
  port 80
  keepalive type http
  active

service serverDEF
  type ssl-accel-backend
  ip address 192.168.7.2
  protocol tcp
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list test
  active

service serverGHI
  ip address 192.14.7.3
  protocol tcp
  port 80
  keepalive type http
  active
```

```
service serverJKL
  type ssl-accel-backend
  ip address 192.168.7.4
  protocol tcp
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list test
  active

!************************** OWNER ***************************
owner ap.com

content ssl-rule
    vip address 192.168.5.5
    protocol tcp
    port 443
    add service ssl_module1
    add service ssl_module2
    application ssl
    advanced-balance ssl
    active

content http-rule
    vip address 192.168.5.5
    protocol tcp
    port 80
    url "/*"
    add service serverABC
    add service serverGHI
    advanced-balance cookies
    active

content ssl-rule-1
    vip address 192.28.4.4
    protocol tcp
    port 443
    add service ssl_module1
    add service ssl_module2
    application ssl
    advanced-balance ssl
    active
```

```
content http-ssl-rule
    vip address 192.28.4.4
    protocol tcp
    port 8080
    url "/*"
    add service serverDEF
    add service serverJKL
    advanced-balance arrowpoint-cookie
    active
```

SSL Full Proxy Configuration — One SSL Module

An SSL full proxy server is a proxy server that terminates the client's SSL connections and initiates the back-end connection to the HTTP server using a different source IP address than that of the client. This configuration does not preserve the client's IP address for the back-end connection to the HTTP server.

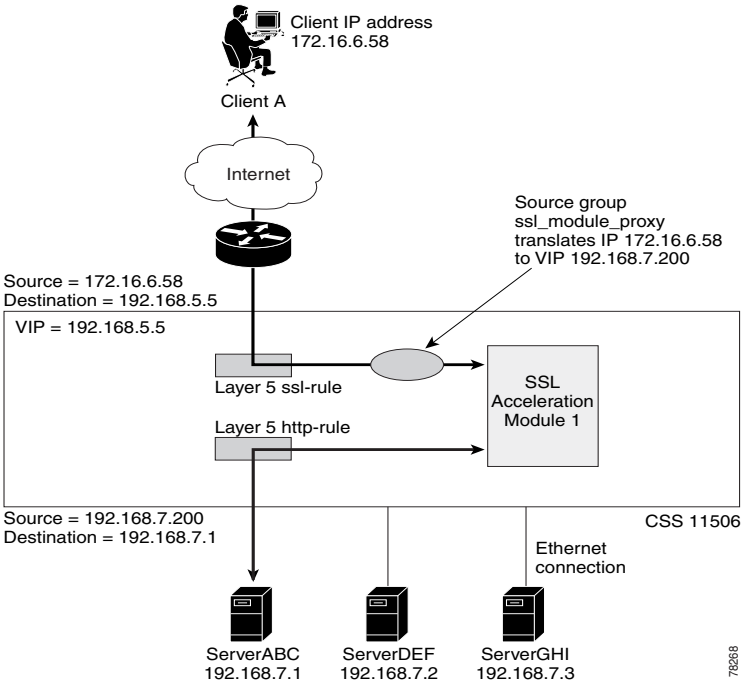
This section provides an example configuration for an SSL full proxy between a client, a CSS with a single SSL module, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). A Layer 5 sticky content rule is used in the configuration. For the CSS to implement a full proxy configuration with an SSL module, the configuration includes a source group that is used to isolate the SSL module traffic and to NAT its source address.

Figure 8-6 illustrates this full proxy configuration.

For purposes of illustration, the configuration example in Figure 8-6 shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rule (http-rule). These two VIP addresses do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP address for the clear-text content rule to place it in nonroutable address space.

In this example, instead of specifying a VIP address of 192.168.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 8-6 Full Proxy Configuration Using a Single SSL Module




```
!*************************** GLOBAL ****************************
  logging commands enable

  ssl associate dsakey dsakey dsakey.pem
  ssl associate dhparams dhparams dhparams.pem
  ssl associate rsakey rsakey rsakey.pem
  ssl associate cert rsacert rsacert.pem

  ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeeebabfcfagyezc5f /
```

```
!************************* CIRCUIT **************************
circuit VLAN1

  ip address 192.168.8.254 255.255.255.0

circuit VLAN2

  ip address 192.168.7.254 255.255.255.0

!********************** SSL PROXY LIST **********************
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.168.5.5
  ssl-server 111 port 443
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
  active

!************************* SERVICE **************************
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service ssl_module2
  type ssl-accel
  keepalive type none
  slot 6
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.168.7.1
  protocol tcp
  port 80
  keepalive type http
  active

service serverDEF
  ip address 192.168.7.2
  protocol tcp
  port 80
  keepalive type http
  active
```

```
service serverGHI
  ip address 192.168.7.3
  protocol tcp
  port 80
  keepalive type http
  active

!************************** OWNER ***************************
owner ap.com

content ssl-rule
    vip address 192.168.5.5
    protocol tcp
    port 443
    add service ssl_module1
    add service ssl_module2
    application ssl
    advanced-balance ssl
    active

content http-rule
    vip address 192.168.5.5
    protocol tcp
    port 80
    url "/*"
    add service serverABC
    add service serverDEF
    add service serverGHI
    advanced-balance cookies
    active

!************************** GROUP ***************************
group ssl_module_proxy
  add destination service serverABC
  add destination service serverDEF
  add destination service serverGHI
  vip address 192.168.7.200
  active
```

SSL Initiation Configurations

SSL initiation is the process whereby a properly configured CSS with an SSL module receives clear text from a client and connects that flow with an SSL flow that is originated by a back-end server configured on the SSL module. Use this configuration for secure site-to-site data tranfers.

This section provides two SSL initiation example:

- SSL Tunnel to Four Data Centers
- SSL Tunnel to One Data Center with Server Authentication

SSL Tunnel to Four Data Centers

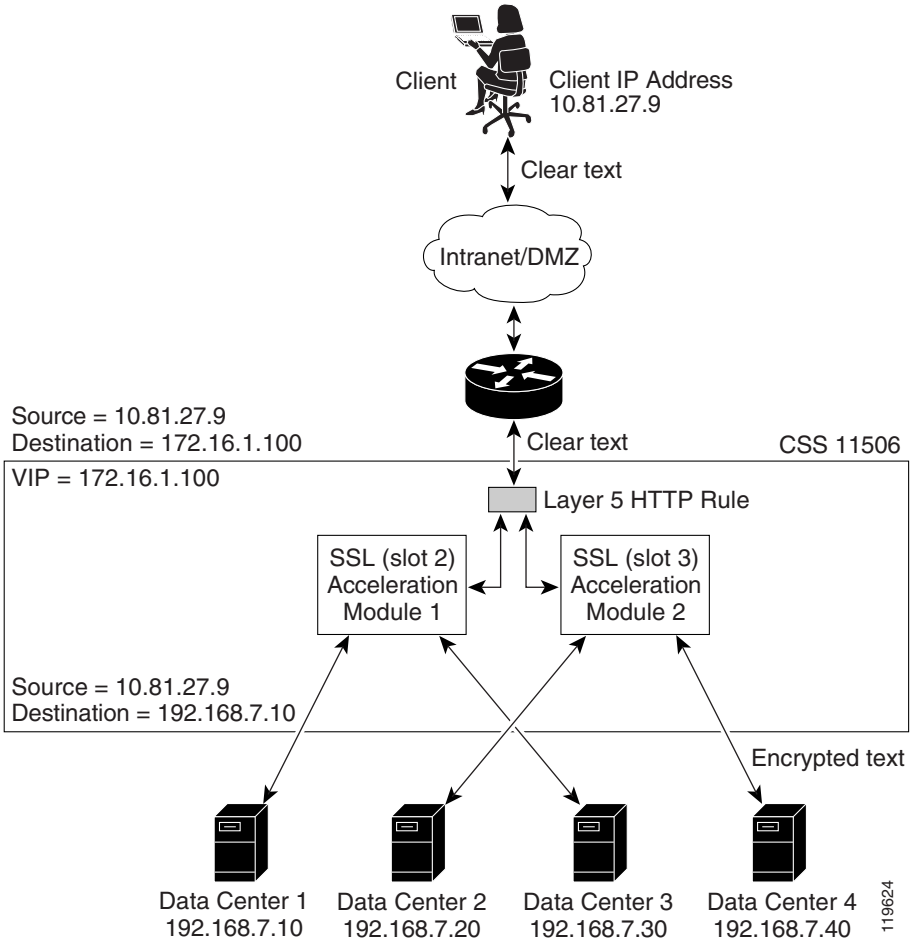
In Figure 8-7, an office contains a CSS 11506 with two SSL modules. Clients connect to a CSS VIP using clear text. The CSS load balances (by applying one of the **advanced-balance** sticky commands), NATs, and sends the connection to an SSL initiation service.

The service of type **ssl-init** tells the CSS to send the connection to the SSL module defined by the **slot** command. The service also defines the IP address of the destination (remote site).

When the connection leaves the service and hits the appropriate SSL module, the SSL proxy list must contain the destination IP address (the **ssl-init** service IP address). The SSL module encrypts the traffic and sends it to the configured destination.

- To optimally load balance flows, you must balance the SSL initiation VIPs and the SSL modules when multiple SSL modules exist (as in this example).
- The SSL initiation feature requires that the proxy list be applied to the SSL module via a service of type **ssl-init**.

Figure 8-7 SSL Initiation Between a CSS and Four Data Centers




```
!!************************** GLOBAL ***************************
  ssl associate rsakey rsakey_association rsakey.pem
  ssl associate cert rsacert_association rsacert.pem

  ftp-record acct-ftp 192.168.7.241 root des-password
ig5haaufqbnfuarb/tmp

!************************ INTERFACE ************************
interface 1/1
  bridge vlan 10
```

```
interface 1/2
  bridge vlan 20

!************************* CIRCUIT **************************
circuit VLAN10

  ip address 172.16.1.1 255.255.255.0

circuit VLAN20

  ip address 192.168.7.1 255.255.255.0

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

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

service DC1
  type ssl-init
  ip address 192.168.7.10
  protocol tcp
  port 80
  slot 2
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list SSLInit_list
active

service DC2
  type ssl-init
  ip address 192.168.7.20
```

```
  protocol tcp
  port 80
  slot 3
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list SSLInit_list
  active

service DC3
  type ssl-init
  ip address 192.168.7.30
  protocol tcp
  port 80
  slot 2
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list SSLInit_list
  active

service DC4
  type ssl-init
  ip address 192.168.7.40
  protocol tcp
  port 80
  slot 3
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list SSLInit_list
  active

!************************** OWNER ***************************
owner Example

  content ssl-init
    protocol tcp
    vip address 172.16.1.100
    port 80
    add service DC1
    add service DC2
    add service DC3
    add service DC4
    advanced-balance arrowpoint-cookie
    active
```

SSL Tunnel to One Data Center with Server Authentication

In Figure 8-8, an office contains a CSS 11506 with two SSL modules. Clients connect to the CSS VIP 192.168.7.101 using clear text. The CSS load balances (by applying the **advanced-balance arrowpoint-cookie** sticky commands), NATs, and sends the connection to an SSL initiation service.

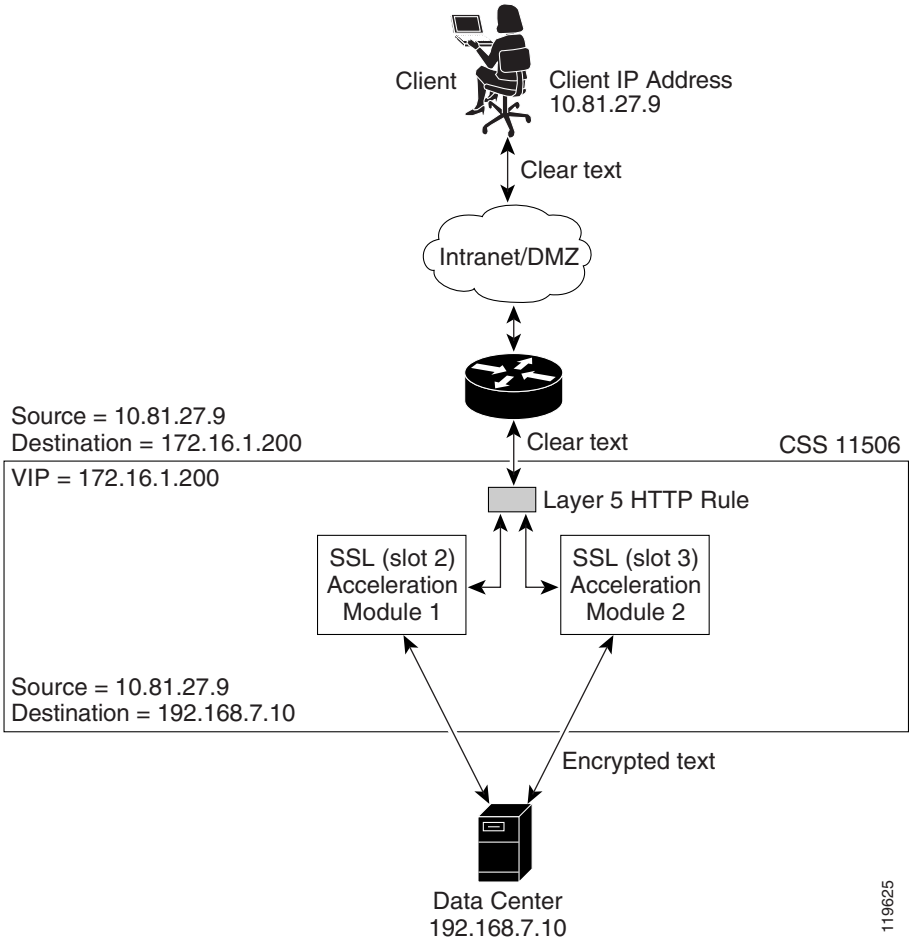
The service of type **ssl-init** tells the CSS to send the connection to the SSL module defined by the **slot** command. The service also defines the IP address of the destination (data center).

When the traffic leaves the service and enters the appropriate SSL module (in this case, slot 2), the SSL proxy list must contain the destination IP address (the **ssl-init** service IP address). The SSL module encrypts the traffic and sends it to the configured destination. By adding the certificate of the CA that signed the SSL server certificate, the CSS can authenticate the server during the SSL handshake.

Be aware of the following configuration requirements:

- To optimally utilize multiple SSL modules, you must balance the SSL initiation VIPs and the SSL modules in your configuration.
- You must apply the SSL initiation proxy list to the SSL module using a service of type **ssl-init**.
- You must obtain the certificate of the CA that issued the SSL server certificate. After you import it and associate it, define the CA certificate as a cacert within the SSL proxy list.

Figure 8-8 SSL Initiation Between a CSS and One Data Center




```
!************************** GLOBAL ***************************
  ssl associate rsakey rsakey_association rsakey.pem
  ssl associate cert rsacert_association rsacert.pem

  ftp-record acct-ftp 192.168.7.241 root des-password
ig5haaufqbnfuarb/tmp
  ftp-record config 192.168.1.241 root des-password 4f1bxangrgehjgka
    /users/rclement/ssl-init
```

```
!************************ INTERFACE ************************
interface 1/1
  bridge vlan 10

interface 1/2
  bridge vlan 20

!************************ CIRCUIT ************************
circuit VLAN10

  ip address 172.16.1.1 255.255.255.0

circuit VLAN20

  ip address 192.168.7.1 255.255.255.0

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

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

service DC-SSL1
  type ssl-init
  ip address 192.168.7.10
  protocol tcp
  port 80
  slot 2
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list SSLInit_list
  active

service DC-SSL2
  type ssl-init
  ip address 192.168.7.10
  protocol tcp
  port 80
  slot 3
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list SSLInit_list
  active
```

```
!************************** OWNER ***************************
owner Example

  content ssl-init
    protocol tcp
    vip address 192.168.7.200
    port 80
    add service DC-SSL1
    add service DC-SSL2
    advanced-balance arrowpoint-cookie
    active
```