CHAPTER 10

# Configuring SSL Traffic through the CSS

This chapter describes how to configure the CSS and the SSL Acceleration Module to perform a Secure Sockets Layer (SSL) connection between a client and HTTP or SSL servers. It also provides an overview on SSL as implemented by the SSL Acceleration Module in the CSS. Information in this chapter applies only to a Cisco 11500 series CSS containing an SSL Acceleration Module (hereinafter referred to as the SSL module).

**Note** The SSL feature is intended for use with the optional SSL module.

This chapter contains the following major sections:

- SSL Cryptography Overview
- Overview of the SSL Module Functions in the CSS
- SSL Configuration Quick Starts
- Configuring SSL Certificates and Keys
- Configuring an SSL Proxy List
- Adding SSL Proxy Lists to Services
- Configuring an SSL Content Rule

# SSL Cryptography Overview

Secure Sockets Layer (SSL) is an application-level protocol that provides encryption technology for the Internet, ensuring secure transactions such as the transmission of credit card numbers for e-commerce Web sites. SSL provides the secure transaction of data between a client and a server through a combination of privacy, authentication, and data integrity. SSL relies upon certificates, private-public key exchange pairs, and Diffie-Hellman key agreement parameters for this level of security.

The CSS uses the SSL module and a special set of SSL commands to perform the SSL cryptographic functions between a client and HTTP servers. The SSL functions include user authentication, private-key and public-key generation, certificate management, and data packet encryption and decryption.

The SSL module supports SSL version 3.0 and Transport Layer Security (TLS) version 1.0. The module understands and accepts an SSL version 2.0 ClientHello message to allow dual version clients to communicate with the CSS through the SSL module. In this case, the client indicates an SSL version of 3.0 in the version 2.0 ClientHello, which informs the SSL module that the client can support SSL version 3.0. The SSL module returns a version 3.0 ServerHello message.

**Note** Although there are very few clients on the market today that support only SSL version 2.0, the SSL module will be unable to pass network traffic if the client supports only version 2.0.

A typical SSL session with the SSL module requires encryption ciphers to establish and maintain the secure connection. Cipher suites provide the cryptographic algorithms required by the SSL module to perform key exchange, authentication, and Message Authentication Code (MAC). See the "Specifying Cipher Suites" section in this chapter for details about the supported cipher suites.

This section provides an overview on SSL cryptography as implemented through the SSL module in the CSS. It covers:



# SSL Public Key Infrastructure Overview

SSL provides authentication, encryption, and data integrity in a Public Key Infrastructure (PKI). PKI is a set of policies and procedures to establish a secure information exchange between devices. Three fundamental elements characterize the PKIs used in asymmetric cryptography. These three elements provide a secure system for deploying e-commerce and a reliable environment for building virtually any type of electronic transactions, from corporate intranets to Internet-based e-business applications.

These elements include:

- confidentiality
- authentication
- message integrity

## Confidentiality

*Confidentiality* means that unintended users cannot view the data. In PKIs, confidentiality is achieved by encrypting the data through a variety of methods. In SSL, specifically, large amounts of data are encrypted using one or more symmetric keys that are known only by the two endpoints. Because the symmetric key is usually generated by one of the endpoints, it must be transmitted securely to the other endpoint. Secure transmittal of a symmetric key is generally achieved by two mechanisms, *key exchange* or *key agreement*.

*Key exchange* is the most common of these two secure transmittal mechanisms. In key exchange, one device generates the symmetric key and then encrypts it using an asymmetric encryption scheme before transmitting it to the other side. Asymmetric encryption requires both devices have a public key and a private key.

The two keys are mathematically related; data that can be encrypted by the public key can be decrypted by the private key, and vice versa. The most commonly used key exchange algorithm is the Rivest Shamir Adelman (RSA) algorithm.

For SSL, the sender encrypts the symmetric keys with the public key of the receiver. This ensures that the private key of the receiver is the only key that can decrypt the transmission. The security of asymmetric encryption depends entirely on the fact that the private key is known only by the owner and not by any other party. If this key were compromised for any reason, a fraudulent Web user (or Web site) could decrypt the stream containing the symmetric key and the entire data transfer.

In *key agreement*, the two sides involved in a data exchange cooperate to generate a symmetric (shared) key. The most common key agreement algorithm is the Diffie-Hellman algorithm. Diffie-Hellman depends on certain parameters to generate the shared key that is calculated and exchanged between the client and the server.

## Authentication

*Authentication* is necessary for one or more devices in the exchange to verify that the party to whom they are talking to is really who they claim to be. For example, assume a client is connecting to an e-commerce website. Before sending sensitive information such as a credit card number, the client verifies that the server is an e-commerce website. In certain instances, it may be necessary for both the client and the server to authenticate themselves to each other before beginning the transaction. In a financial transaction between two banks, both the client and the server need to be confident that the other is who they say they are. SSL facilitates this authentication through the use of digital certificates.

Digital certificates are a form of digital identification to prove the identity of the client to the server. A Certificate Authority (CA) issues digital certificates in the context of a PKI, which uses public-key and private-key encryption to ensure security. CAs are trusted authorities who “sign” certificates to verify their authenticity. Clients or servers connected to the CSS must have trusted certificates from the same CA, or from different CAs in a hierarchy of trusted relationships (for example, “A” trusts “B,” and “B” trusts “C,” therefore “A” trusts “C”).

A certificate ensures that the identification information is correct, and that the public key actually belongs to that client or server. Digital certificates contain information such as details about the owner, details about the certificate issuer, the owner's public key, validity and expiration dates, and associated privileges.

Upon receiving a certificate, a client can connect to the certificate issuer and verify the validity of the certificate using the issuer's public key. This ensures that the certificate is actually issued and signed by an authorized entity. A certificate remains valid until it expires or is terminated.

## Message Integrity

*Message integrity* is a means of assuring the recipient of a message that the contents of the message have not been tampered with during transit. SSL achieves this by applying a message digest to the data before transmitting it. A message digest is a function that takes an arbitrary length message and outputs a fixed-length string that is characteristic of the message.

An important property of the message digest is that it is extremely difficult to reverse. Simply appending a digest of the message to itself before sending it is not enough to guarantee integrity. An attacker can change the message and then change the digest accordingly. Encoding the message digest with the sender's private key creates a Message Authentication Code (MAC), the message integrity algorithm, which the recipient can then decode using the sender's public key. SSL supports two different algorithms for a MAC: Message Digest 5 (MD5) and Secure Hash Algorithm (SHA).

This integrity scheme, however, does not work if the sender's private key is compromised. The attacker can now forge the sender's MACs. Message integrity also depends heavily on the protection of private keys. This process is known as digital signing.

RSA key pairs are effective for signing the MAC. However, it may be advantageous to separate the functions of key exchange and signing. The Digital Signature Algorithm (DSA) is an SSL algorithm that is used strictly for digital signatures but not for key exchange.

DSA was standardized as FIPS-186, which is the Digital Signature Standard (DSS). DSA and DSS can be used interchangeably. DSS uses the same crypto-math as Diffie-Hellman and requires parameters similar to Diffie-Hellman to generate keys. Additionally, DSS is restricted for use only with the Secure Hash Algorithm 1 (SHA-1) message digest.

# SSL Module Cryptography Capabilities

Table 10-1 provides information on the SSL cryptography capabilities of the SSL module.

*Table 10-1 SSL Module SSL Cryptography Capabilities*

| SSL Cryptography Function | Functions Supported by the SSL Module |
|---|---|
| SSL versions | SSL version 3.0 and Transport Layer Security (TLS) version 1.0 |
| Public key exchange and key agreement algorithms | • **RSA** - 512-bit, 768-bit, 1024-bit, and 2048-bit (key exchange and key agreement algorithm)<br>• **DSA** - 512-bit, 768-bit, 1024-bit, and 2048-bit[1] (certificate signing algorithm)<br>• **Diffie-Hellman** - 512-bit, 768-bit, 1024-bit, and 2048-bit (key agreement algorithm) |
| Encryption types | • Data Encryption Standard (DES)<br>• Triple-Strength Data Encryption Standard (3DES)<br>• RC4<br>See Table 10-14 for a list of supported cipher suites and key encryption types. |
| Hash types | • SSL MAC-MD5<br>• SSL MAC-SHA1<br>See Table 10-14 for a list of supported cipher suites and hash types. |

*Table 10-1 SSL Module SSL Cryptography Capabilities (continued)*

| SSL Cryptography Function | Functions Supported by the SSL Module |
|---|---|
| Digital certificates | The SSL module supports all major digital certificates from Certificate Authorities (CAs), including those listed below:<br>• VeriSign<br>• Entrust<br>• Netscape iPlanet<br>• Windows 2000 Certificate Server<br>• Thawte<br>• Equifax<br>• Genuity |

1. The SSL module supports the importing of 2048-bit DSA key pairs. The SSL module does not support the generation of 2048-bit DSA key pairs.

# Overview of the SSL Module Functions in the CSS

The Cisco 11500 series CSS supports multiple SSL modules. The CSS 11501 supports a single integrated SSL module. The CSS 11503 and CSS 11506 support multiple SSL modules; a maximum of two in a CSS 11503 and a maximum of four in a CSS 11506.

The SSL module is responsible for all user authentication, public/private key generation, certificate management, and packet encryption and decryption functions between the client and the server. It is dependent on the Switch Module to provide the interface for processing network traffic and the Switch Control Module (SCM) to send and receive configuration information.

The CSS stores all certificates and keys on the SCM disk. The CSS supports a maximum of 256 certificates and 256 key-pairs, which equals approximately 3 MB of storage space on the disk. The CSS stores all certificate- and key-related files in a secure location on the disk. When processing connections, the CSS loads the certificates and keys into volatile memory on the SSL module for faster access.

No network traffic is sent to an SSL module from the SCM until an SSL content rule is activated to:

- Define where the content physically resides
- Where to direct the request for content (which service)
- Which load-balancing method to use

An SSL proxy list determines the flow of information to and from an SSL module. An entry in the proxy list defines the flow from a client to an SSL module. An entry also defines a flow from an SSL module to a backend SSL server. To define how an SSL module processes SSL requests for content, add an SSL proxy list to an SSL service.

When you create an entry in a proxy list to define the flow between an SSL module and a client, the module operates as a virtual SSL server by adding security services between the web browsers (the client) and the HTTP connection. All inbound SSL flows from a client terminate at an SSL module.

Once the connection is terminated, the SSL module decrypts the data and sends the data as clear text to the CSS for a decision on load balancing. The CSS transmits the data as clear text either to an HTTP server or back to the SSL module when a backend SSL server is configured on the CSS.

A backend SSL server entry in an SSL proxy list defines the flow from the SSL module to the backend SSL server. The SSL module, acting as a virtual client preserving the originating client's IP address, encrypts the clear text data and initiates the SSL connection to the backend server.

On the outbound flow from the CSS, the SSL module responds in the reverse direction and sends the data back to the client.

For more detailed information, see the "Processing of SSL Flows by the SSL Module" section.

# SSL Configuration Quick Starts

This section provides a quick overview on how to manage SSL certificates in the CSS, create an SSL proxy list for virtual and backend SSL servers, and add an SSL proxy list to an SSL service. Each step includes the CLI command required to complete the task. RSA has been chosen for the quick start procedures in this section because it is a popular public-key algorithm for encryption and authentication.

This section includes the following quick start procedures:

- RSA Certificate and Key Generation Quick Start
- RSA Certificate and Key Import Quick Start
- SSL Proxy List Quick Start
- SSL Service and Content Rule Quick Start

## RSA Certificate and Key Generation Quick Start

Table 10-2 provides an overview of the steps required to generate and associate an RSA key pair and certificate in the CSS. Key and certificate generation may be necessary in instances when you do not have pre-existing keys or certificates for the CSS. You may want to initially generate RSA keys and temporary certificates on the CSS for internal SSL testing until the Certificate Authority responds to the certificate request and returns the authentic certificate. A generated certificate is temporary and expires in 30 days.

*Table 10-2 RSA Certificate and Key Generation Quick Start*

| Task and Command Example |
|---|
| **1.** Enter configuration mode.<br>`# config`<br>`(config) #` |
| **2.** Generate the RSA key pair used in the exchange.<br>`(config) # ssl genrsa CSSrsakey1 1024 "passwd123"`<br>`Please be patient this could take a few minutes` |
| **3.** Associate the generated RSA key pair with a file.<br>`(config) # ssl associate rsakey myrsakey1 CSSrsakey1` |

*Table 10-2 RSA Certificate and Key Generation Quick Start (continued)*

| Task and Command Example |
|---|
| **4.** After generating the RSA key pair, generate the Certificate Signing Request (CSR) file for the RSA key pair file and transfer the certificate request to the Certificate Authority (CA).<br><br>`(config) # ssl gencsr myrsakey1`<br>`You are about to be asked to enter information.` |
| **5.** While awaiting the return of the site certificate from the CA, testing can be done by generating a self-signed (temporary) certificate.<br><br>`(config) # ssl gencert certkey myrsakey1 signkey myrsakey1 CSScertfile1 "passwd123"`<br>`You are about to be asked to enter information` |
| **6.** Associate the generated certificate with a file.<br><br>`(config) # ssl associate cert myrsacert1 CSScertfile1` |
| **7.** Compare the public key in the associated certificate with the public key stored with the associated private key and verify that they are identical.<br><br>`(config) # ssl verify myrsacert1 myrsakey1`<br>`Certificate mycert1 matches key mykey1` |

# RSA Certificate and Key Import Quick Start

Table 10-3 provides an overview of the steps required to import and associate an RSA certificate and key pair to the CSS from a remote server.

*Table 10-3 RSA Certificate and Key Import Quick Start*

| Task and Command Example |
|---|
| **1.** Define a secure File Transfer Protocol (FTP) record file to import certificates and private keys into the CSS from an SFTP server.<br>`# ftp-record ssl_record 192.168.19.21 johndoe "abc123" /home/johndoe` |
| **2.** Use secure FTP to transfer the imported certificates and private keys to the CSS.<br>`# copy ssl sftp ssl_record import rsacert.pem PEM "passwd123"`<br>`Connecting`<br>`Completed successfully`<br><br>`# copy ssl sftp ssl_record import rsakey.pem PEM "passwd123"`<br>`Connecting`<br>`Completed successfully` |
| **3.** Enter configuration mode.<br>`# config`<br>`(config) #` |
| **4.** To use RSA public key exchange and authentication:<br>**a.** Associate the imported RSA certificate with a file.<br>`(config) # ssl associate cert myrsacert1 rsacert.pem`<br>**b.** Associate the imported RSA key pair with a file.<br>`(config) # ssl associate rsakey myrsakey1 rsakey.pem` |
| **5.** Compare the public key in the associated certificate with the public key stored with the associated private key and verify that they are identical.<br>`(config) # ssl verify myrsacert1 myrsakey1`<br>`Certificate mycert1 matches key mykey1` |

# SSL Proxy List Quick Start

An SSL proxy list configures the flow of data to and from an SSL module. You must define a virtual SSL server entry in an SSL proxy list for an SSL module to properly process and terminate SSL communications from the client and initiate an HTTP connection to the CSS.

When clear text data from the HTTP connection requires encryption to a backend SSL server, a backend SSL server entry in the SSL proxy list allows an SSL module to encrypt the data and initiate an SSL connection to the backend server.

Table 10-4 provides an overview of the steps required to create a virtual SSL server entry in an SSL proxy list for an RSA certificate and key pair.

*Table 10-4 CSS SSL Proxy List for a Virtual SSL Server Quick Start*

| Task and Command Example |
|---|
| **1.** Create the SSL proxy list.<br>`(config)# ssl-proxy-list ssl_list1`<br>`Create ssl-list <ssl_list1>, [y/n]: y`<br>Once you create an SSL proxy list, the CLI enters into ssl-proxy-list configuration mode for the newly created SSL proxy list.<br>`(config-ssl-proxy-list[ssl_list1])#` |
| **2.** Specify a number to identify a virtual SSL server in the SSL proxy list.<br>`(config-ssl-proxy-list[ssl_list1])# ssl-server 20` |
| **3.** Specify a virtual IP (VIP) address. Enter a VIP address that corresponds to an SSL content rule.<br>`(config-ssl-proxy-list[ssl_list1])# ssl-server 20 vip address 192.168.3.6` |
| **4.** (Optional) Specify the virtual TCP port number if you need to change it to correspond with the content rule. By default, the virtual TCP port number is 443.<br>`(config-ssl-proxy-list[ssl_list1])# ssl-server 20 port 444` |

*Table 10-4 CSS SSL Proxy List for a Virtual SSL Server Quick Start (continued)*

| Task and Command Example |
|---|
| **5.** Specify the name of an existing RSA certificate association and RSA key pair association for the SSL proxy list virtual SSL server.<br>`(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert myrsacert1`<br>`(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey myrsakey1` |
| **6.** Assign the appropriate cipher suite for the RSA certificates and keys in use, the IP address of the backend content rule used for the cipher suite, and the TCP port of the backend content rule.<br>`(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cipher rsa-export-with-rc4-40-md5 192.168.3.6 8080 5` |
| **7.** (Optional) Specify the URL rewrite option for the domain name of the URL to be redirected to avoid nonsecure HTTP 300-series redirects.<br>`(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22 www.mydomain.com` |
| **8.** Continue to Table 10-5 if the flow requires encryption to a backend SSL server. If not, continue to step 9. |
| **9.** Activate the completed SSL proxy list.<br>`(config-ssl-proxy-list[ssl_list1])# active` |

Table 10-5 provides an overview of steps required to create an SSL proxy list between the SSL module and the backend SSL server.

*Table 10-5 CSS SSL Proxy List Entry for a Backend SSL Server Quick Start*

| Task and Command Example |
|---|
| **1.** Specify a number to identify a backend SSL server in the SSL proxy list.<br>`(config-ssl-proxy-list[ssl_list1])# backend-server 1` |
| **2.** Specify an IP address. Enter an IP address that corresponds to the address of the service for the backend SSL server.<br>`(config-ssl-proxy-list[ssl_list1])# backend-server 1 ip address 192.168.4.4` |

*Table 10-5 CSS SSL Proxy List Entry for a Backend SSL Server Quick Start (continued)*

| Task and Command Example |
|---|
| 3. (Optional) By default, the virtual TCP port number for the backend server is 80. Assign the virtual TCP port number if you need to change it.<br>`(config-ssl-proxy-list[ssl_list1])# backend-server 1 port 8080` |
| 4. Specify server IP address for the backend server. Enter a valid IP address for the server.<br>`(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-ip 192.168.4.4` |
| 5. (Optional) By default, the server port number for the backend server is 443. Assign the server port number if you need to change it.<br>`(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-port 113` |
| 6. (Optional) By default, the backend server supports all available CSS cipher suites. If necessary, assign a specific cipher suite to be used by the backend SSL server, for example the RSA certificates and keys:<br>`(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher rsa-export-with-rc4-40-md5` |
| 7. Activate the completed SSL proxy list.<br>`(config-ssl-proxy-list[ssl_list1])# active` |

# SSL Service and Content Rule Quick Start

Table 10-6 provides an overview of the steps required to create an SSL service for a virtual SSL server, including adding the SSL proxy list to the service and creating an SSL content rule.

*Table 10-6 CSS SSL Server Service and Content Rule Quick Start*

| Task and Command Example |
|---|
| **1.** Create an SSL service.<br>`(config)# service ssl_serv1`<br>`Create service <ssl_serv1>, [y/n]: y` |
| **2.** Specify **ssl-accel** as the service type.<br>`(config-service[ssl_serv1])# type ssl-accel` |
| **3.** Specify the slot of the SSL module in the CSS chassis.<br>`(config-service[ssl_serv1])# slot 3` |
| **4.** Disable the CSS from sending keepalive messages to the service.<br>`(config-service[ssl_serv1])# keepalive type none` |
| **5.** Add the SSL proxy list to the SSL service.<br>`(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1` |
| **6.** Activate the SSL service.<br>`(config-service[ssl_serv1])# active` |
| **7.** Create an SSL content rule.<br>`(config)# owner ssl_owner`<br>`Create owner <ssl_owner>, [y/n]: y`<br>`(config-owner[ssl_owner])# content ssl_rule1`<br>`Create content <ssl_rule1>, [y/n]: y` |
| **8.** Configure a virtual IP (VIP) address or domain name for the content rule. Ensure the VIP address is the same as the address specified in the SSL proxy list.<br>`(config-owner-content[ssl-rule1]# vip address 192.168.3.6` |
| **9.** Specify a TCP port number for the content rule. Ensure the port number is the same as the port specified in the SSL proxy list.<br>`(config-owner-content[ssl-rule1]# port 444` |

*Table 10-6 CSS SSL Server Service and Content Rule Quick Start (continued)*

| Task and Command Example |
|---|
| **10.** If you are using two or more SSL modules and want to use stickiness based on SSL version 3 session ID for a Layer 5 content rule, specify the following parameters in the content rule to take advantage of the SSL session ID reuse:<br>• Enter the **application ssl** command to specify the SSL application type.<br>`(config-owner-content[ssl-rule1])# application ssl`<br>• Enter the **advanced-balance ssl** command to enable the content rule to be sticky based on SSL.<br>`(config-owner-content[ssl-rule1])# advanced-balance ssl` |
| **11.** Add the SSL service to the content rule.<br>`(config-owner-content[ssl_rule1])# add service ssl_serv1` |
| **12.** Activate the content rule.<br>`(config-owner-content[ssl_rule1])# active` |
| **13.** Save your configuration changes to the running configuration.<br>`# copy running-config startup-config` |
| **14.** Continue to Table 10-7 if your configuration includes a backend SSL server. |

If you configured a backend SSL server entry in an SSL proxy list, Table 10-7 provides an overview of the steps required to create an SSL service for a backend SSL server, including adding the SSL proxy list to the service and creating an SSL content rule.

*Table 10-7 CSS Backend SSL Server Service and Content Rule Quick Start*

| Task and Command Example |
|---|
| **1.** Create an SSL service.<br>`(config)# service ssl_serv2`<br>`Create service <ssl_serv2>, [y/n]: y` |
| **2.** Specify **ssl-accel-backend** as the service type.<br>`(config-service[ssl_serv2])# type ssl-accel-backend` |

*Table 10-7 CSS Backend SSL Server Service and Content Rule Quick Start (continued)*

| Task and Command Example |
|---|
| **3.** Configure a virtual IP (VIP) address for the backend server. The IP address must match the IP address configured for the backend server.<br>`(config-service[ssl_serv2])# vip address 192.168.4.4` |
| **4.** (Optional) Configure a virtual port number for the backend server. The port number must match the virtual TCP port number configured for the backend server. By default, the port number is 80. In this example, the port number is 8080.<br>`(config-service[ssl_serv2])# port 8080` |
| **5.** (Optional) By default, the service keepalive type is ICMP. You can also configure the keepalive type for a backend service to be none, TCP, or SSL. If you configure a TCP or SSL keepalive type, you must configure the keepalive port correctly for the service to work.<br>For example, to configure a keepalive type of SSL, enter.<br>`(config-service[ssl_serv2])# keepalive type ssl`<br>Then configure the port for the backend SSL server. For example, enter:<br>`(config-service[ssl_serv2])# keepalive port 443` |
| **6.** Add the SSL proxy list to the SSL service.<br>`(config-service[ssl_serv2])# add ssl-proxy-list ssl_list1` |
| **7.** Activate the SSL service.<br>`(config-service[ssl_serv2])# active` |
| **8.** Add the backend server to an SSL content rule.<br>`(config)# owner ssl_owner`<br>`(config-owner[ssl_owner])# content ssl_backend_rule1`<br>`Create content <ssl_backend_rule1>, [y/n]: y` |
| **9.** Configure a virtual IP (VIP) address or domain name for the content rule. Ensure the VIP address is the same as the address specified for the content rule for the virtual SSL server.<br>`(config-owner-content[ssl_backend_rule1]# vip address 192.168.3.6` |

*Table 10-7 CSS Backend SSL Server Service and Content Rule Quick Start (continued)*

| Task and Command Example |
|---|
| **10.** Specify a TCP port number for the content rule. Ensure the port number is the same as the virtual TCP port specified for the backend SSL entry in the SSL proxy list.<br>`(config-owner-content[ssl_backend_rule1]# port 8080` |
| **11.** Enter the **advanced-balance arrowpoint-cookie** command to enable the content rule to be sticky based on an arrowpoint cookie.<br>`(config-owner-content[ssl_backend_rule1])# advanced-balance arrowpoint-cookie` |
| **12.** Enter the **url** command set to /* to use stickiness based on the cookie.<br>`(config-owner-content[ssl_backend_rule1])# url "/*"` |
| **13.** Add the SSL service to the content rule.<br>`(config-owner-content[ssl_backend_rule1])# add service ssl_serv2` |
| **14.** Activate the content rule.<br>`(config-owner-content[ssl_backend_rule1])# active` |
| **15.** Save your configuration changes to the running configuration.<br>`# copy running-config startup-config` |

# Configuring SSL Certificates and Keys

Before creating an SSL proxy list and adding the SSL proxy list to an SSL service, you must load a set of digital certificates and public/private key pairs on the CSS disk (flash disk or hard disk). The digital certificates and key pairs are a form of digital identification for user authentication. Certificates are issued by Certificate Authorities (CAs) such as VeriSign and Thawte. Each certificate includes the name of the issuing authority, the name of the entity that the certificate was issued for, the entity's public key, and timestamps that indicate the certificate's expiration date. You can use files received from a CA, import the certificate and keys from an existing secure server, or generate your own certificate and keys on the CSS.

The CSS supports the generation of certificates and keys directly within the CSS for purposes of testing. Your requirement to use generated certificates and keys instead of certificates and keys from a trusted authority depends on your environment. For example, the use of the CSS and SSL for a company's internal Web site may not require the use of certificates from a trusted CA. A certificate and key pair generated within the CSS may be sufficient to satisfy the intranet SSL requirement.

The CSS stores digital certificates and key pairs in encrypted files in a secure area on the CSS.

**Caution** When importing or exporting certificates and keys with the CSS, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (operating the CSS in a diskless environment). The network-mounted method of CSS booting is not supported with SSL termination; the certificates and keys must be local to the CSS and SSL module.

This section covers:

- Importing or Exporting Certificates and Private Keys
- Generating Certificates and Private Keys in the CSS
- Associating Certificates and Private Keys in the CSS
- Showing Certificate and Key Pair Information
- Removing Certificates and Private Keys from the CSS

**Note** To implement good security policies when importing or generating SSL certificates and key pairs, administrators should understand the various user modes of the CSS and have strong password policies to protect those user modes. For more information, refer to the *Cisco Content Services Switch Command Reference*, Chapter 2, CLI Commands, the "(config) username-technician" section.

# Importing or Exporting Certificates and Private Keys

You can import preexisting or new certificates and private keys to the CSS disk from a file, or a series of files, that are stored on a remote secure server. To transfer these files, Cisco Systems recommends that you use a secure encrypted transport mechanism between the CSS and the remote server. The CSS supports the Secure Shell protocol (SSHv2), which provides secure encryption communications between two hosts over an insecure network. The CSS supports file transport between network devices using the Secure File Transfer Protocol (SFTP) and the File Transfer Protocol (FTP). Of the two file transport protocols, Cisco Systems recommends SFTP as the transport mechanism of choice. It is similar to FTP except that it uses a secure and encrypted connection.

Before you import certificates or keys to the CSS:

- On the CSS, ensure that SSH access to the CSS is enabled to accept connections from SSH clients and that the Secure Management license key is installed prior to transferring certificates and keys. By default, SSH access is enabled through the **no restrict ssh** global command. If SSH access is restricted, or if the license key is not installed, SSH will not accept connections from SSH clients and the **copy ssl sftp** command will fail, resulting in generation of an error message.

**Note** For details about configuring Secure Shell Daemon on the CSS, refer to the *Cisco Content Services Switch Administration Guide*.

- On the SFTP server, verify that the server is properly configured so that the user directory points to the directory where the certificates and keys reside. This path is required to ensure certificates and keys are properly copied from or to the SFTP server.

**Caution** When using SSH, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (a diskless environment). If SSH is enabled and the CSS has been booted using a network boot from a network-mounted file system, the CSS logs an error message by SSH as the protocol attempts to initialize and then exits from operation, which impacts importing and exporting certificates and keys.

## Configuring the Default SFTP or FTP Server to Import Certificates and Private Keys

Before you begin, use the **ftp-record** command to define the SFTP or FTP server that you intend to use to download imported certificates and private keys to the CSS disk. For details about using the **ftp-record** command to create an SFTP or FTP record file to use when accessing the server from the CSS, refer to the *Cisco Content Services Switch Administration Guid*e.

**Note** When defining the FTP record for the **copy ssl** command, ensure that the base directory, if used, is relative to the SSH directory where the SSH server resides. For example, if the username is *sshlogin* and the SSH server is installed in d:\Program Files\Network, the default directory for the files would be d:\Program Files\Network\ssh. This path is required to ensure certificates and keys are properly copied to or from the SFTP server.

For example, to define the *ssl_record*, enter:

```
# ftp-record ssl_record 192.168.19.21 johndoe "abc123"
/home/johndoe
```

## Transferring Certificates and Private Keys to the CSS

Use the **copy ssl** command to facilitate the import or export of certificates and private keys from or to the CSS. The CSS stores all imported files in a secure location on the CSS. This command is available only in SuperUser mode.

The syntax for this command is:

**copy ssl** [*protocol*] *ftp_record* [**import** *filename* [*format*] *"password"* {**"***passphrase***"**}|**export** *filename2 "password"*]

The variables are:

- *protocol* - The type of protocol used to transfer the certificate and private key file. The valid entries are **sftp** or **ftp**. Cisco Systems recommends the SFTP protocol for the transport mechanism because it provides the most security.
- *ftp_record* - The name of the previously-created FTP record containing the remote host information.
- **import** - Import the file from the remote server.

- *filename* - The name of the file you want to import from the server. Include the full path to the file. You can enter a maximum of 128 characters.
- *format* - The file format of the certificate to be imported. Once the certificate file is converted to PEM format and DES encoded, it is stored on the CSS SCM in a special (and secure) directory. The valid import file formats are:
  - **DER** - Binary format encoding of the certificate file in ASN.1 using the Distinguished Encoding Rules (DER-encoded X509 certificate). For example, an imported certificate from a Microsoft Windows NT IIS 4.0 server.
  - **PEM** - Privacy Enhanced Mail, a base64 encoding of the certificate file (PEM-encoded X509 certificate). For example, an imported certificate from an Apache/SSL UNIX server.
  - **PKCS12** - Standard from RSA Data Security, Inc. for storing certificates and private keys. For example, an imported certificate from a Microsoft Windows 2000 IIS 5.0 server.
- *"password"* - The password used to DES (Data Encryption Standard) encode the imported certificate or private key. Encoding the imported file prevents unauthorized access to the certificate or private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.
- *"passphrase"* - (Optional for PEM files) The passphrase used to encrypt the certificate or key being imported into the CSS. Enter the passphrase as a quoted text string.

**Note** You must enter a passphrase for a PKCS12 file (.pfx). The CSS uses the passphrase to decrypt the file.

- **export** - Export the file to the remote server.
- *filename2* - The name you want to assign to the file on the server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum length of 32 characters.

**Note** An imported file can contain certificates, RSA or DSA key pairs, or Diffie-Hellman parameters. You must distinguish whether the files contain certificates, privates keys, or Diffie-Hellman parameters by associating the specific contents to a filename. See the "Associating Certificates and Private Keys in the CSS" section.

For example, to import the *rsacert.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsacert.pem PEM "passwd123"
Connecting
Completed successfully
```

For example, to import the *rsakey.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsakey.pem PEM "passwd123"
Connecting
Completed successfully
```

To export the *rsacert.pem* certificate from the CSS to a remote server, enter:

```
# copy ssl sftp ssl_record export rsacert.pem "passwd123"
```

If the **copy ssl** command fails to import certificates or keys, verify the following areas:

- The user account and password in the ftp record are correct
- The base directory is ssh or ssh/path
- The SSH server is reachable
- The SSH server IP address is correct in the ftp-record

# Generating Certificates and Private Keys in the CSS

If you do not have preexisting keys, Diffie-Hellman parameters, and certificates for the CSS, you may want to generate them on the CSS disk for purposes of testing. The CSS includes a series of certificate and private key management utilities. These utilities simplify the process of generating an RSA private key, DSA private key, a Diffie-Hellman parameter file, a certificate signing request (CSR), and a self-signed temporary certificate.

**Note** The **ssl genrsa**, **gencsr**, **gendsa**, and **gencert** commands all produce a valid certificate or key pair. Be aware, however, that most Web browsers will flag the certificate as signed by an unrecognized signing authority.
A generated certificate is temporary and expires in 30 days.

This section covers:

- Generating an RSA Key Pair
- Generating a DSA Key Pair
- Generating Diffie-Hellman Key Parameters
- Using an RSA Key to Generate a Certificate Signing Request
- Generating a Self-Signed Certificate

## Generating an RSA Key Pair

Use the **ssl genrsa** command to generate an RSA private/public key pair for asymmetric encryption. RSA key pairs are used to sign and encrypt packet data, and they are required before another device (client or server) can exchange an SSL certificate with the CSS. The key pair refers to a public key and its corresponding private (secret) key. The CSS stores the generated RSA key pair as a file on the CSS.

The syntax for this command is:

**ssl genrsa** *filename numbits "password"*

The variables are:

- *filename* - The name of generated RSA key pair file. Enter an unquoted text string with a maximum of 31 characters. The key pair filename is used only for identification in the CSS.

- *numbits* - The key pair strength. The number of bits in the key pair file defines the size of the RSA key pair used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the RSA security policy. Available entries (in bits) are 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- *"password"* - The password used to encode the RSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the RSA key pair *myrsakeyfile1*, enter:

```
(config) # ssl genrsa myrsakeyfile1 1024 "passwd123"
 Please be patient this could take a few minutes
```

After you generate an RSA key pair, you can generate a Certificate Signing Request (CSR) file for the RSA key pair file and transfer the certificate request to the Certificate Authority (CA). This provide an added layer of security because the RSA private key originates directly within the CSS and does not have to be transported externally. You can then create a temporary certificate for internal testing until the CA responds to the certificate request and returns the authentic certificate. Each generated key pair must be accompanied by a certificate to work.

You must also associate an RSA key pair name to the generated RSA key pair, as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

## Generating a DSA Key Pair

Use the **ssl gendsa** command to generate a DSA private/public key pair for asymmetric encryption. DSA is the public key exchange cryptographic system developed by the National Institutes of Science and Technology. DSA can only be used for digital signatures (signings); it cannot be used for key private/public exchange. The CSS stores the generated DSA key pair as a file on the CSS.

The syntax for this command is:

**ssl gendsa** *filename numbits "password"*

The variables are:

- *filename* - The name of the generated DSA key pair file. Enter an unquoted text string with a maximum of 31 characters. The key pair filename is used only for identification in the CSS.
- *numbits* - The key pair strength. The number of bits in the key pair file defines the size of the DSA key pair used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the DSA security policy. Available entries (in bits) are 512 (least security), 768 (normal security), and 1024 (highest security).
- *"password"* - The password used to encode the DSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the DSA key pair *mydsakeyfile2*, enter:

```
(config) # ssl gendsa mydsakeyfile2 512 "passwd123"
Please be patient this could take a few minutes
```

You must also associate a DSA key pair name to the generated DSA key pair as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

## Generating Diffie-Hellman Key Parameters

Use the **ssl gendh** command to generate a Diffie-Hellman key agreement parameter file. Diffie-Hellman is a shared key agreement algorithm. Diffie-Hellman key exchange uses a complex algorithm and public/private keys to encrypt and then decrypt packet data. The CSS stores the generated Diffie-Hellman key parameter file.

**Note** Generation of a Diffie-Hellman key agreement parameter file can sometimes take a lengthy period of time (perhaps up to 20 minutes) and is a CPU-intensive utility. If you are running the **ssl gendh** utility, ensure that the CSS is not actively passing traffic at the same time to avoid impacting CSS performance.

The syntax for this command is:

**ssl gendh** *filename numbits "password"*

The variables are:

- *filename* - The name of the file to store the Diffie-Hellman key parameters. Enter an unquoted text string with a maximum of 31 characters. The filename is used only for identification in the CSS.
- *numbits* - The key strength. The number of bits in the file defines the size of the Diffie-Hellman key used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the Diffie-Hellman security policy. Available entries (in bits) are 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- *"password"* - The password used to encode the Diffie-Hellman key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the Diffie-Hellman key parameter list *dhparamfile2*, enter:

```
(config) # ssl gendh dhparamfile2 512 "passwd123"
Please be patient this could take a few minutes
```

You must also associate a Diffie-Hellman parameter filename to the generated Diffie-Hellman parameter file, as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

## Using an RSA Key to Generate a Certificate Signing Request

Use the **ssl gencsr** *rsakey* command to generate a Certificate Signing Request (CSR) file for an RSA key pair file and to transfer the certificate request to the Certificate Authority (CA). You must generate a CSR file if you are requesting a new certificate or renewing a certificate. When the CA signs the CSR using its RSA private key, the CSR becomes the certificate.

The *rsakey* variable specifies the key on which the RSA certificate is built. It is the public key that is embedded in the certificate.

To use the RSA key pair to generate a CSR, ensure the RSA key pair file is loaded on the CSS. Associate an RSA key pair name to the generated RSA keypair (see the "Associating Certificates and Private Keys in the CSS" section). If the appropriate key pair does not exist, the CSS logs an error message.

For example, to generate a CSR based on the RSA key pair *myrsakey1*, enter:

```
CSS11503(config)# ssl gencsr myrsakey1
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]MA
Locality Name (city) [SomeCity]Boxborough
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com]webadmin@cisco.com

-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQICAQAwgZwxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNQTETMBEGA1UE
BxMKQm94Ym9yb3VnaDEcMBoGA1UEChMTQ2lzY28gU3lzdGVtcywgSW5jLjESMBAG
A1UECxMJV2ViIEFkbWluMRYwFAYDVQQDEw13d3cuY2lzY28uY29tMSEwHwYJKoZI
hvcNAQkBFhJra3JvZWJlckBjaXNjby5jb20wXDANBgkqhkiG9w0BAQEFAANLADBI
AkEAqHXjtQUVXvmo6tAWPiMpe6oYhZbJUDgTxbW4VMCygzGZn2wUJTgLrifDB6N3
v+1tKFndE686BhKqfyOidml3wQIDAQABoAAwDQYJKoZIhvcNAQEEBQADQQA94yC3
4SUJJ4UQEnO2OqRGLOZpAElc4+IV9aTWK6NmiZsM9Gt0vPhIkLx5jjhVRLlb27Ak
H6D5omXa0SPJan5x
-----END CERTIFICATE REQUEST-----

CSS11503(config)#
```

The **ssl gencsr** command generates the CSR and outputs it to the screen. Most major Certificate Authorities have Web-based applications that require you to cut and paste the certificate request to the screen. Note that the CSR is not saved in the CSS.

To test your CSR file, you can create a temporary certificate by generating a CSR and signing it with your own private key. While this produces a valid certificate, most browsers flag the certificate as signed by an unrecognized signing authority. To generate a temporary certificate, see the "Generating a Self-Signed Certificate" section.

## Generating a Self-Signed Certificate

Use the **ssl gencert** command to generate and save a temporary certificate to a file on disk in the CSS. For purposes of SSL testing you can generate a temporary certificate by generating a CSR and signing it with your own private key. A generated certificate is temporary and expires in 30 days.

**Note** The **ssl gencert** command produces a valid certificate. However, most Web browsers flag this certificate as signed by an unrecognized signing authority.

Before you generate the certificate, consider:

- The key pair that the certificate is based on (RSA or DSA).
- The key used to sign the certificate.

The **ssl gencert** command can sign RSA or DSA certificates with either an RSA key pair or a DSA key pair.

**Note** Although the CSS allows signing an RSA certificate with a DSA key (and a DSA certificate with an RSA key) it is a more standard practice that an RSA certificate is signed with RSA keys (and DSA certificate is signed with a DSA key).

The syntax for this command is:

**ssl gencert certkey** *certkey* **signkey** *signkey certfile* **"***password***"**

The variables are:

- **certkey** *certkey* - The name of the RSA or DSA key pair on which the certificate is based. Enter an unquoted text string with a maximum of 31 characters.
- **signkey** *signkey* - The RSA or DSA key pair to be used to sign the certificate. Enter an unquoted text string with a maximum of 31 characters.
- *certfile* - The name of the file used to store the certificate as a file on the CSS. Enter an unquoted text string with a maximum of 31 characters.

- *"password"* - The password used to encode the certificate file using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to interactively generate the *mycertfile2* certificate, enter:

```
CSS11503(config)# ssl gencert certkey myrsakey signkey
myrsasignkey myrsacertfile "passwd123"
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]MA
Locality Name (city) [SomeCity]Boxborough
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com]webadm@cisco.com

CSS11503(config)#
```

You must also associate the contents of this temporary certificate to a filename, as discussed in the "Associating Certificates and Private Keys in the CSS" section of this chapter.

# Associating Certificates and Private Keys in the CSS

After you import or generate certificate and key pair files, you must distinguish to the CSS whether these files contain certificates, private keys, or Diffie-Hellman parameters. You do this by associating certificate names, private/public key pair names, or Diffie-Hellman parameter names to the particular imported files.

When you associate the entries specified in the various certificate and private key commands to files, the CSS stores the bindings in the running configuration. Before you log out or reboot the CSS, you must copy the contents of the running-config file to the startup-config file to save configuration changes and have the CSS use this configuration on subsequent reboots. When you reboot the CSS, the certificate and key associations are loaded automatically.

This section covers:



## Associating a Certificate to a File

Use the **ssl associate cert** command to associate a certificate name to an imported or generated certificate. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

**ssl associate cert** *certname filename*

The variables are:

- *certname* - The name of the certificate association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the certificate. Enter a maximum of 128 characters. To see a list of imported or generated certificates, use the **ssl associate cert** *certname* **?** command.

For example, to associate the certificate name *myrsacert*1 to the imported certificate file *rsacert.pem*, enter:

```
(config) # ssl associate cert myrsacert1 rsacert.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate ssl cert myrsacert1
```

**Note** The **no** form of the command will not function if the associated certificate is in use by an active SSL proxy list.

## Associating an RSA Key Pair to a File

Use the **ssl associate rsakey** command to associate an RSA key pair name to an imported or generated RSA key pair. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

**ssl associate rsakey** *keyname filename*

The variables are:

- *keyname* - The name of the RSA key pair association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the RSA key pair. Enter a maximum of 128 characters. To see a list of imported or generated RSA keys, use the **ssl associate rsakey** *keyname* **?** command.

For example, to associate the RSA key name *myrsakey1* to the imported *rsakey.pem*, enter:

```
(config) # ssl associate rsakey myrsakey1 rsakey.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate rsakey myrsakey1
```

**Note** The **no** form of the command will not function if the associated RSA key pair is in use by an active SSL proxy list.

## Associating a DSA Key Pair to a File

Use the **ssl associate dsakey** command to associate a DSA key pair name to an imported or generated DSA key pair. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

**ssl associate dsakey** *keyname filename*

The variables are:

- *keyname* - The name of the DSA key pair association. Enter an unquoted text string with a maximum of 31 characters.

- *filename* - The name of the file containing the DSA key pair. Enter a maximum of 128 characters. To see a list of imported or generated DSA keys, use the **ssl associate dsakey** *keyname* **?** command.

For example, to associate the DSA key name *mydsakey1* to the imported *dsakey.pem*, enter:

```
(config) # ssl associate dsakey mydsakey1 dsakey.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate dsakey mydsakey1
```

**Note** The **no** form of the command will not function if the associated DSA key pair is in use by an active SSL proxy list.

## Associating Diffie-Hellman Parameters to a File

Use the **ssl associate dhparam** command to associate a Diffie-Hellman name to an imported or generated Diffie-Hellman parameter file. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

**ssl associate dhparam** *paramname filename*

The variables are:

- *paramname* - The name of the Diffie-Hellman parameter association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the Diffie-Hellman parameters. Enter a maximum of 128 characters. To see a list of imported or generated Diffie-Hellman files, use the **ssl associate dhparam** *filename* **?** command.

For example, to associate the Diffie-Hellman filename *mydhparam1* to the imported *dhparams.pem*, enter:

```
(config) # ssl associate dhparam mydhparam1 dhparams.pem
```

To remove the association to the file, enter:

```
(config) # no ssl associate dhparam mydhparam1
```

**Note** The **no** form of the command will not function if the associated Diffie-Hellman parameter list is in use by an active SSL proxy list.

## Verifying a Certificate Against a Key Pair

A digital certificate is built around a public key, and can only be used with one key pair. Use the **ssl verify** command to compare the public key in the associated certificate with the public key stored with the associated private key, and verify that they are identical. To see a list of certificate and key pair associations, use the **ssl verify ?** command.

**Note** If the certificate does not match the public/private key pair, the CSS logs an error message.

The syntax for this command is:

**ssl verify** *certname keyname*

The variables are:

- *certname* - The association name of the certificate used to verify against the specified key pair.
- *keyname* - The association name of the key pair used to verify against the specified certificate.

For example, to verify the *myrsacert1* digital certificate against the *myrsakey1* key pair, enter:

```
(config)# ssl verify myrsacert1 myrsakey1
Certificate and key match
```

# Showing Certificate and Key Pair Information

A number of **show** commands in the CSS enable you to display information about SSL certificates and key pairs stored on the CSS. Enter the following **show** commands from any mode:

- **show ssl associate cert** - Displays certificate associations
- **show ssl associate rsakey** - Displays RSA key pair associations
- **show ssl associate dsakey** - Displays DSA key pair associations
- **show ssl associate dhparam** - Displays information about Diffie-Hellman parameter associations
- **show ssl associate** - Displays all file associations for the CSS
- **show ssl files** - Displays all certificate, key pair, and Diffie-Hellman parameter files loaded on the CSS

## Showing SSL Certificates

Use the **show ssl associate cert** *certname* command to display summary data for certificate associations in the CSS. You can optionally specify a certificate name to view detailed information about the certificate, corresponding to the certificate association. If you do not specify a certificate name, all certificate associations appear in the **show ssl associate cert** output.

To display information about all certificate associations, enter:

```
show ssl associate cert
```

Table 10-8 describes the fields in the **show ssl associate cert** output.

***Table 10-8 Field Descriptions for the show ssl associate cert Command***

| Field | Description |
|---|---|
| Certificate Name | The name of the certificate association |
| File Name | The name of the file containing the certificate |
| Used By List | Indicates if the certificate association is used by the SSL proxy list containing the VIP address of the virtual server |

To display information about a specific certificate association, enter:

```
show ssl associate cert myrsacert1
```

Table 10-9 describes the fields in the **show ssl associate cert** *certname* output.

***Table 10-9 Field Descriptions for the show ssl associate cert certname Command***

| Field | Description |
|---|---|
| Certificate | The name of the Certificate Association (CA) that issued the certificate. |
| Version | The version of the certificate. |
| Serial Number | The serial number associated with the certificate. |
| Signature Algorithm | The digital signature algorithm (such as RSA) used for the encryption of information with a public/private key pair. |
| Issuer | The organization that generated the certificate and will vouch for it. An issuer is also the Certificate Authority (CA). |
| **Validity** | |
| Not Before | The starting time period, before which the certificate is not considered valid. |
| Not After | The ending time period, after which the certificate is not considered valid. |
| Subject | The certified party that possesses the private key. |
| **Subject Public Key Info** | |
| Public Key Algorithm | The name of the key exchange algorithm used to generate the public key (for example, RSA). |
| RSA Public Key | The number of bits in the key to define the size of the RSA key pair used to secure Web transactions. |
| Modulus | The actual public key on which the certificate was built. |
| Exponent | One of the base numbers used to generate the key. |

*Table 10-9 Field Descriptions for the show ssl associate cert certname Command (continued)*

| Field | Description |
|---|---|
| X509v3 Extensions | An array of X509v3 extensions added to the certificate. |
| X509v3 Basic Constraints | Indicates if the subject may act as a CA, with the certified public key being used to verify certificate signatures. If so, a certification path length constraint may also be specified. |
| Netscape Comment | A comment that may be displayed when the certificate is viewed. |
| X509v3 Subject Key Identifier | Identifies the public key being certified. It enables distinct keys used by the same subject to be differentiated (for example, as key updating occurs). |
| X509v3 Authority Key Identifier | Identifies the public key to be used to verify the signature on this certificate or CRL. It enables distinct keys used by the same CA to be distinguished (for example, as key updating occurs). |
| Signature Algorithm | The name of the algorithm used for digital signatures (but not for key exchanges). |
| Hex Numbers | The actual signature of the certificate. The client can regenerate this signature using the specified algorithm to make sure that the certificate data has not been changed. |

## Showing SSL RSA Private Keys

Use the **show ssl associate rsakey** *keyname* command to obtain information about RSA private key associations in the CSS. You can optionally specify an RSA key name to view information about a specific RSA key association (key size and type). If you do not specify an RSA keyname, you see a list of all RSA key associations.

**Note** When you view the contents of a specific key only, specifics on the key size and key type appears. This restriction occurs because the key contents are secure and should not be viewed.

To display information about all RSA private key associations:

```
(config) # show ssl associate rsakey
```

Table 10-10 describes the fields in the **show ssl associate rsakey** output.

***Table 10-10 Field Descriptions for the show ssl associate rsakey Command***

| Field | Description |
|---|---|
| Key Name | The name of the RSA key association |
| File Name | The name of the file containing the RSA key pair |
| Used By List | Indicates if the RSA key association is used by the SSL proxy list containing the VIP address of the virtual server |

To display information about a specific RSA key pair association, enter:

```
(config) # show ssl associate rsakey myrsakey1
1024-bit RSA keypair
```

## Showing SSL DSA Private Keys

Use the **show ssl associate dsakey** *keyname* command to obtain information about DSA private key associations in the CSS. You can optionally specify a DSA key name to view information about a specific DSA key association (key size and type). If you do not specify a DSA keyname, you see a list of all DSA key associations.

**Note** When you view the contents of a specific key only, specifics on the key size and key type appears. This restriction occurs because the key contents are secure and should not be viewed.

To display information about all DSA key associations, enter:

```
(config) # show ssl associate dsakey
```

Table 10-11 describes the fields in the **show ssl associate dsakey** output.

***Table 10-11 Field Descriptions for the show ssl associate dsakey Command***

| Field | Description |
|---|---|
| Key Name | The name of the DSA key association |
| File Name | The name of the file containing the DSA key pair |
| Used By List | Indicates if the DSA key association is used by the SSL proxy list containing the VIP address of the virtual server |

To display information about a specific DSA key pair association, enter:

```
(config) # show ssl associate dsakey mydsakey1
1024-bit DSA keypair
```

## Showing SSL Diffie-Hellman Parameters

Use the **show ssl associate dhparam** *paramname* to obtain information about Diffie-Hellman parameters. You can optionally specify a parameter filename to view information about a specific Diffie-Hellman parameter file association. If you do not specify a Diffie-Hellman parameter filename, you see a list of all Diffie-Hellman parameter file associations.

To display information about all Diffie-Hellman associations:

```
(config) # show ssl associate dhparam
```

Table 10-12 describes the fields in the **show ssl associate dhparam** output.

***Table 10-12 Field Descriptions for the show ssl associate dhparam Command***

| Field | Description |
|---|---|
| Parameter Name | The name of the Diffie-Hellman parameter association |
| File Name | The name of the file containing the Diffie-Hellman parameters |
| Used By List | Indicates if the Diffie-Hellman file association is used by the SSL proxy list containing the VIP address of the virtual server |

To display information about a specific Diffie-Hellman parameter file association, enter:

```
(config) # show ssl associate dhparam mydhparam1
512-bit DH parameters
```

## Showing SSL Associations

Use the **show ssl associate** to display a summary of all certificate and key associations stored on the CSS.

To display a summary of SSL associations for the CSS, enter:

```
CSS11506(config)# show ssl associate

 Certificate Name        File Name            Used by List
 ----------------        ---------            ------------
 rsacert                 rsacert.pem          yes

 RSA Key Name            File Name            Used by List
 ------------            ---------            ------------
 rsakey                  rsakey.pem           yes

 DH Param Name           File Name            Used by List
 -------------           ---------            ------------
 dhparams                dhparams.pem         no

 DSA Key Name            File Name            Used by List
 ------------            ---------            ------------
 dsakey                  dsakey.pem           no
```

## Showing SSL Certificates, Key Pairs, and Diffie-Hellman Parameter Files

Use the **show ssl files** to display a list of certificates, key pairs, and Diffie-Hellman parameter files loaded on the CSS.

For example, enter:

```
(config) # show ssl files
```

Table 10-13 describes the fields in the **show ssl files** output.

*Table 10-13 Field Descriptions for the show ssl files Command*

| Field | Description |
|---|---|
| File Name | The name of the imported or manually-generated certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file. |
| File Type | The format of the imported or manually-generated certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file. File types can include DES-encoded, PEM-encoded, or PKCS#12-encoded. |
| File Size | The total size (in Kbytes) of the certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file. |

# Removing Certificates and Private Keys from the CSS

Use the **clear ssl file** command to remove certificates and private keys from the CSS that are no longer valid. Note that the **clear ssl file** command does not function if the file currently has an association with it. First remove the association to the file by specifying the **no ssl associate** command (see the "Associating Certificates and Private Keys in the CSS" section).

The syntax for this global configuration mode command is:

**clear ssl file** *filename password*

The variables are:

- *filename* - The name of the certificate, key pair, or Diffie-Hellman parameter file that you want to remove from the CSS.
- *password* - The password used to encode the file using DES when it was originally imported or generated by the CSS. This password must be an exact match or the file cannot be cleared.

For example, to remove *dsacert.pem* from the CSS, enter:

```
# clear ssl file dsacert.pem "passwd123"
```

# Configuring an SSL Proxy List

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

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

This section describes the steps required to configure an SSL proxy list for a virtual or backend server, and to activate it. It covers:

- Creating an SSL Proxy List
- Adding a Description to an SSL Proxy List
- Configuring Virtual SSL Servers for an SSL Proxy List
- Configuring Backend SSL Servers for an SSL Proxy List
- Activating and Suspending an SSL Proxy List

# Creating an SSL Proxy List

An SSL proxy list is a group of related virtual or backend SSL servers that are associated with an SSL service.

Use the **ssl-proxy-list** command to create an SSL proxy list. You can access the ssl-proxy-list configuration mode from most configuration modes except for ACL, boot, group, rmon, or owner configuration modes. You can also use this command from the ssl-proxy-list configuration mode to access another SSL proxy list.

Enter the SSL proxy list name as an unquoted text string from 1 to 31 characters in length.

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

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

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

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

To delete an existing SSL proxy list, enter:

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

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

# Adding a Description to an SSL Proxy List

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

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

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

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

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

# Configuring Virtual SSL Servers for an SSL Proxy List

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

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

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

**Note** You cannot modify the virtual SSL servers in an active SSL proxy list. You must first suspend the SSL proxy list to make modifications to any of the virtual SSL servers in a specific SSL proxy list. Once you have modified the SSL proxy list, suspend the SSL service, activate the SSL proxy list, and then activate the SSL service.

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

- Creating an SSL Proxy Configuration ssl-server Index
- Specifying a Virtual IP Address
- Specifying a Virtual Port
- Specifying the RSA Certificate Name
- Specifying the RSA Key Pair Name
- Specifying the DSA Certificate Name
- Specifying the DSA Key Pair Name
- Specifying the Diffie-Hellman Parameter Filename
- Specifying Cipher Suites
- Specifying SSL or TLS Version
- Specifying Secure URL Rewrite
- Specifying SSL Session Cache Timeout
- Specifying SSL Session Handshake Renegotiation
- Specifying SSL TCP Client-Side Connection Timeout Values
- Specifying SSL TCP Server-Side Connection Timeout Values
- Specifying the Nagle Algorithm for SSL TCP Connections

## Creating an SSL Proxy Configuration ssl-server Index

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

Enter a value from 1 to 256 that corresponds to the SSL proxy list virtual SSL server number.

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

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

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

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

## Specifying a Virtual IP Address

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

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

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

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

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

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

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

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

## Specifying a Virtual Port

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

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

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

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

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

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

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

## Specifying the RSA Certificate Name

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

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

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

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

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

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

## Specifying the RSA Key Pair Name

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

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

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

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

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

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

## Specifying the DSA Certificate Name

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

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

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

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

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

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

## Specifying the DSA Key Pair Name

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

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

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

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

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

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

## Specifying the Diffie-Hellman Parameter Filename

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

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

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

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

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

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

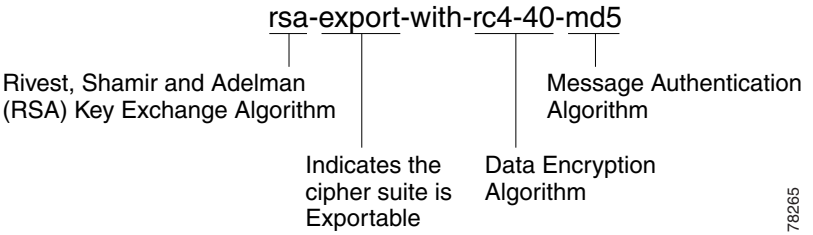
## Specifying Cipher Suites

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

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

Each cipher suite specifies a set of key exchange algorithms. Figure 10-1 summarizes the algorithms associated with the rsa-export-with-rc4-40-md5 cipher suite.

***Figure 10-1 Cipher Suite Algorithms***




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

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

The syntax for this command is:

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

The options and variables are:

- **ssl-server** *number* - The number used to identify the virtual SSL server in the SSL proxy list.
- **cipher** *name* - The name of a specific cipher suite (as listed in Table 10-14).
- *ip_address* or *hostname* - The IP address to assign to the backend content rule used with the cipher suite. Specify the IP address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com).

- *port* - The TCP port of the backend content rule through which the backend HTTP connections are sent.
- **weight** *number* - Optional parameter. Assigns a priority to the cipher suite, with 10 being the highest weight. By default, all configured cipher suites have a weight of 1. When negotiating which cipher suite to use, the SSL module selects from the client list based on the cipher suite configured with the highest weight. A higher weight will bias towards the specified cipher suite. To set the weight for a cipher suite, enter a number from 1 to 10. The default is 1.

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

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

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

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

Table 10-14 lists all supported cipher suites and values for the specific SSL server (and corresponding SSL proxy list). Table 10-14 also lists whether those cipher suites are exportable from the CSS, along with the authentication certificate and encryption key required by the cipher suite.

If you use the default setting or select the **all-cipher-suite** option, the CSS sends the suites in the same order as they appear in Table 10-14, starting with rsa-with-rc4-128-md5.

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

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

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

*Table 10-14 SSL Cipher Suites Supported by the CSS*

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

*Table 10-14 SSL Cipher Suites Supported by the CSS (continued)*

| Cipher Suite | Exportable | Authentication Certificate Used | Key Exchange Algorithm Used |
|---|---|---|---|
| dhe-dss-export-with-des40-cbc-sha | Yes | DSA (DSS) certificate | Ephemeral Diffie-Hellman key exchange |
| dhe-rsa-export-with-des40-cbc-sha | Yes | RSA certificate | Ephemeral Diffie-Hellman |
| dh-anon-export-with-rc4-40-md5 | Yes | Neither party is authenticated | Diffie-Hellman |
| dh-anon-export-with-des40-cbc-sha | Yes | Neither party is authenticated | Diffie-Hellman |
| rsa-export1024-with-des-cbc-sha | Yes | RSA certificate | RSA key exchange |
| dhe-dss-export1024-with-des-cbc-sha | Yes | DSA (DSS) certificate | Ephemeral Diffie-Hellman |
| rsa-export1024-with-rc4-56-sha | Yes | RSA certificate | RSA key exchange |
| dhe-dss-export1024-with-rc4-56-sha | Yes | DSA (DSS) certificate | Ephemeral Diffie-Hellman |

## Specifying SSL or TLS Version

Use the **ssl-server** *number* **version** *protocol* command to specify the SSL or Transport Layer Security (TLS) protocol version. By default, the SSL version is SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

The options include:

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

For example, to specify SSL version 3.0, enter:

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

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

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

## Specifying Secure URL Rewrite

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

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

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

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

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

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

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

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

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

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

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

The options and variables are:

- **ssl-server** *number* - The number used to identify the virtual SSL server in the SSL proxy list.
- **urlrewrite** *number* - The number of the URL rewrite rule to be added to the virtual SSL server. Enter a value from 1 to 32 corresponding to the URL rewrite rule. You can add a maximum of 32 URL rewrite rules to each SSL server for handling HTTP to HTTPS redirects.

- *hostname* - The domain name of the URL to be redirected (for example, www.mydomain.com). Enter an unquoted text string with a maximum length of 240 characters that corresponds to the domain name of the URL rewrite host. Do not include the directory path as part of the hostname. If you intend to use wildcards in domain names to identify and match on more than one host in a single domain, insert an asterisk (*) wild card character in the domain name.
- **sslport** *port* - (Optional) Specifies the port used for SSL network traffic. Enter a TCP port number that corresponds with an SSL content rule, which uses the specified TCP port number. The SSL module rewrites an HTTP redirect matching the URL redirect rule with the specified SSL port (or default port 443 if no port number is specified). Enter a port value from 1 to 65535. The default value is 443.
- **clearport** *port* - (Optional) Specifies the port used for clear text network traffic. The SSL module matches redirects in the Location Response-Header field with the specified clear text port (or default port 80 if no port number is specified). Enter a port value from 1 to 65535. The default value is 80.

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

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

To remove URL rewrite rule 22, enter:

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

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

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

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

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

## Specifying SSL Session Cache Timeout

Use the **ssl-server** *number* **session-cache** *seconds* command to configure the SSL module to resume connection with a client using a previously established secret key. In SSL, a new session ID is created every time the client and the CSS SSL module go through a full key exchange and establish a new master secret key. Specifying an SSL session cache timeout allows the SSL module to reuse the master key on subsequent connections with the client, which can speed up the SSL negotiation process. You can specify a timeout value to set the total amount of time an SSL session ID remains valid before the SSL module requires the full SSL handshake to establish a new SSL connection.

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

Enter an SSL session cache timeout value in seconds, from 0 (SSL session ID reuse disabled) to 72000 (20 hours). The default is 300 seconds (5 minutes). By disabling this option (entering a value of 0), the full SSL handshake occurs for each new connection between the client and the SSL module.

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

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

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

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

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

## Specifying SSL Session Handshake Renegotiation

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

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

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

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

To disable the rehandshake data option, enter:

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

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

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

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

To disable the rehandshake timeout option, enter:

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

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

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

## Specifying SSL TCP Client-Side Connection Timeout Values

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

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

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

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

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

Enter a TCP SYN inactivity timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

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

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

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

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

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

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

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

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

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

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

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

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

# Specifying SSL TCP Server-Side Connection Timeout Values

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

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

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

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

Use the **ssl-server** *number* **tcp server syn-timeout** *seconds* command to specify a timeout value that the CSS uses to end a TCP connection with a server that has not successfully completed the TCP three-way handshake prior to transferring data. The SYN timer counts the delta between the CSS initiating the backend TCP connection by transmitting a SYN and the server replying with a SYN/ACK.

Enter a TCP SYN timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

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

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

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

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

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

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

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

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

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

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

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

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

## Specifying the Nagle Algorithm for SSL TCP Connections

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

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

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

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

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

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

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

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

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

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

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

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

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

# Configuring Backend SSL Servers for an SSL Proxy List

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

**Note** You cannot modify the backend SSL servers in an active SSL proxy list. You must first suspend the SSL proxy list to make modifications to any of the backend-servers in a specific SSL proxy list. Once you have modified the SSL proxy list, suspend the SSL service, activate the SSL proxy list, and then activate the SSL service.

To configure a backend server for use by the SSL module, you must create and configure a backend server entry in an SSL proxy list. Configure an IP address that corresponds to the address of the service and the server IP address. Then activate the SSL proxy list.

After you configure and activate the SSL proxy list, add the list to a backend-SSL service; assign a service type of ssl-accel-backend. When you activate the service, the CSS sends the configuration data to the SSL module.

The following sections describe:

- Creating the Backend-Server Entries for an SSL Proxy List
- Configuring the IP Address for an SSL Backend Server
- Configuring the Virtual Port
- Configuring the Server IP Address
- Configuring the Server Port
- Configuring SSL Version
- Configuring the Available Cipher Suites
- Configuring SSL Session Cache Timeout
- Configuring SSL Session Handshake Renegotiation
- Configuring TCP Virtual Client Connections Timeout Values
- Configuring TCP Server-Side Connection Timeout Values on the SSL Module
- Specifying the Nagle Algorithm for SSL TCP Connections

## Creating the Backend-Server Entries for an SSL Proxy List

You must create a backend server entry before you can configure any of the backend server parameters. Use the **backend-server** *number* command to identify backend SSL server parameters for the SSL proxy list. Enter a value from 1 to 256 for the entry.

For example, to create an entry for a backend server 1, enter:

```
(config-ssl-proxy-list[sslist3])# backend-server 1
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1
```

## Configuring the IP Address for an SSL Backend Server

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

For example, to configure the IP address 192.168.2.3 for backend server 1, enter:

```
(config-ssl-proxy-list[sslist3])# backend-server 1 ip address
192.168.2.3
```

To remove the IP address from the backend server, enter:

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 ip address
```

**Note** If you do not configure the IP address when you issue the **active** command, the following error message appears and the CSS does not activate the list.

```
SSL-server/Backend-server must have valid IP Address
```

## Configuring the Virtual Port

By default, the virtual port for the backend server is port 80. The virtual port directs the clear text data traffic from the SSL module to the CSS. To configure a different virtual port for the SSL backend server, use the **backend-server** *number* **port** command. Enter a port number from 1 to 65535.

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 port 1200
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 port
```

## Configuring the Server IP Address

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 server-ip
192.168.2.3
```

To remove the server IP address from the backend server, enter:

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 server-ip
```

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

```
SSL-server/Backend-server must have valid IP address
```

## Configuring the Server Port

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 server-port 155
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 server-port
```

## Configuring SSL Version

For a backend server, the SSL module initiates the SSL connection. The version in the ClientHello message sent to the server indicates the highest supported version.

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

Use the **backend-server** *number* **version** command to specify which version of SSL the backend server supports:

- **ssl3** - SSL version 3.
- **tls1**- TLS version 1.
- **ssl-tls** - SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 version ssl3
```

To reset the default SSL version, enter:

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 version
```

## Configuring the Available Cipher Suites

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

Table 10-14 earlier in this chapter lists all supported cipher suites for the SSL module and the corresponding cipher suite value. These values match those defined for SSL version 3.0 and TLS version 1.0. The table also lists those Cipher suites that are exportable in any version of the software.

If you use the default setting or select the **all-cipher-suite** option, the CSS sends the suites in the same order as they appear in Table 10-14, starting with rsa-with-rc4-128-md5.

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 cipher
rsa-with-rc4-128-md5
```

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

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 cipher
rsa-with-rc4-128-md5 weight 10
```

To remove one or more of the configured cipher suites for the backend server, enter:

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 cipher
rsa-with-rc4-128-md5
```

## Configuring SSL Session Cache Timeout

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

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 session-cache 500
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 session-cache
```

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 session-cache 0
```

## Configuring SSL Session Handshake Renegotiation

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

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

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 handshake data 500
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 handshake data
```

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

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 handshake timeout
30
```

To reset the timeout to 0, disable the rehandshake timeout period for the backend server by entering:

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 handshake
timeout
```

## Configuring TCP Virtual Client Connections Timeout Values

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

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

- Specifying a TCP SYN Timeout Value for the Virtual Client Connection
- Specifying a TCP Inactivity Timeout for a Virtual Client Connection

### Specifying a TCP SYN Timeout Value for the Virtual Client Connection

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

Enter a TCP SYN inactivity timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp virtual
syn-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp virtual
syn-timeout 0
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 tcp virtual
syn-timeout
```

## Specifying a TCP Inactivity Timeout for a Virtual Client Connection

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

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

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp virtual
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp virtual
inactivity-timeout 0
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 tcp virtual
inactivity-timeout
```

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

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

To configure the timeout values of a TCP connection with the server, see the following sections:

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

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

Use the **backend-server** *number* **tcp server syn-timeout** *seconds* command to specify a timeout value that the CSS uses to end a TCP connection with a server that has not successfully completed the TCP three-way handshake prior to transferring data. The SYN timer counts the delta between the CSS initiating the backend TCP connection by transmitting a SYN and the server replying with a SYN/ACK.

Enter a TCP SYN timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp server
syn-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp server
syn-timeout 0
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 tcp server
syn-timeout
```

## Specifying a TCP Inactivity Timeout for a Server-Side Connection

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

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

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

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp server
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[sslist3])# backend-server 1 tcp server
inactivity-timeout 0
```

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

```
(config-ssl-proxy-list[sslist3])# no backend-server 1 tcp server
inactivity-timeout
```

## Specifying the Nagle Algorithm for SSL TCP Connections

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

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

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

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

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual
nagle disable
```

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

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual
nagle enable
```

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

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

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

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

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

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

# Activating and Suspending an SSL Proxy List

Use the **active** command to activate the new or modified SSL proxy list. Before you can activate an SSL proxy list, ensure that you have created at least one virtual or backend SSL server in the list (see the "Configuring Virtual SSL Servers for an SSL Proxy List" section or the "Specifying the Nagle Algorithm for SSL TCP Connections" section earlier in this chapter).

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

To activate an SSL proxy list, enter:

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

After you activate an SSL proxy list, you can add it to a service. See the "Adding SSL Proxy Lists to Services" section later in this chapter.

**Note** No modifications to an SSL proxy list are permitted on an active list. Suspend the list prior to making changes, and then reactivate the SSL proxy list once the changes are complete. Once you have modified the SSL proxy list, suspend the SSL service, reactivate the SSL proxy list, and then reactivate the SSL service.

To view the virtual or backend SSL servers in a list, use the **show ssl-proxy-list** (see the "Showing SSL Proxy Configuration Information" section in this chapter).

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

To suspend an active SSL proxy list, enter:

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

# Adding SSL Proxy Lists to Services

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

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

This section covers:

- Configuring a Virtual SSL Server to a Service
- Configuring a Backend SSL Server to a Service
- Activating the SSL Service
- Suspending the SSL Service

## Configuring a Virtual SSL Server to a Service

After you configure a virtual SSL server on an SSL proxy list for an SSL module, add the active list to an SSL service. The active list explains how the CSS processes SSL requests for content through the specific SSL module. The following sections cover:

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

## Creating an SSL Service for an SSL Module

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

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

To create service *ssl_serv1*, enter:

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

The CSS transitions into the newly created service mode.

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

## Specifying the SSL Acceleration Service Type

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

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

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

To specify the SSL acceleration service type, enter:

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

## Adding an SSL Proxy List to a Service

Use the **add ssl-proxy-list** command in service mode to include an SSL proxy list as part of an SSL service. Enter the name of the previously created SSL proxy list (see the "Creating an SSL Proxy List" section in this chapter) that you want to add to the service.

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

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

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

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

## Specifying SSL Module Slot

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

The valid slot entries are:

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

Slot 1 is reserved for the SCM.

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

For example, to identify an SSL module in slot 3 of the Cisco 11503 chassis, enter:

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

## Disabling Keepalive Messages for the SSL Module

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

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

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

## Specifying the SSL Session ID Cache Size

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

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

The backend session ID cache is 4096 entries and is not configurable.

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

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

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

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

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

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

# Configuring a Backend SSL Server to a Service

After you configure an SSL proxy list for a backend SSL server, add the active list to an SSL service to define how the CSS processes SSL requests for content from a backend SSL server. Configuring the backend SSL service is similar to configuring a local service except you must set the service type to **ssl-accel-backend**. Also, this type of service requires an SSL proxy list with a backend server entry.

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

- The service must have a configured IP address
- The keepalive type for a backend service can be none, ICMP, TCP, or SSL. If you configure a TCP or SSL keepalive type, you must configure the keepalive port correctly for the service to work.
- You must configure an SSL proxy list that contains backend-server configuration for this type of service.

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

The following sections cover:

- Configuring the Backend SSL Service Type
- Adding an SSL Proxy List for a Backend SSL Server
- Configuring an IP Address for a Backend SSL Service
- Configuring the Port Number for a Backend SSL Service

## Configuring the Backend SSL Service Type

You must configure the **ssl-accel-backend** service type for a backend SSL service. To configure a service type for a backend SSL service, enter:

```
(config-service[server1]# type ssl-accel-backend
```

## Adding an SSL Proxy List for a Backend SSL Server

An SSL proxy list contains the parameters for the backend SSL service. To add the proxy list to the service, use the **add ssl-proxy-list** command. For more information on configuring an SSL proxy list for a backend server, see the "Configuring a Backend SSL Server to a Service" section earlier in this chapter.

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

For example, to add the SSL proxy list *ssl list3* for a backend SSL service, enter:

```
(config-service[server1])# add ssl-proxy-list ssllist3
```

To remove an SSL proxy list for the backend service, enter:

```
(config-service[server1])# remove ssl-proxy-list ssllist3
```

## Configuring an IP Address for a Backend SSL Service

The IP address for a backend SSL service must match the IP address configured in the SSL proxy list for the backend server.

For example, to configure the IP address 10.11.21.13 for the backend SSL service, enter:

```
(config-service[server1])# ip address 10.11.21.13
```

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

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

## Configuring the Port Number for a Backend SSL Service

The CSS uses the port number to send clear text data back to the SSL module for reencryption. By default, the CSS uses the port number of the backend content rule associated with the service, port 80. If the port number is different from the the backend HTTP-SSL content rule, use the **port** command to configure it.

Enter the port number as a integer from 1 to 65535. If you configure a port number, it must match the virtual port number configured in the SSL proxy list for the backend server.

For example, to configure a port number of 55, enter:

```
(config-service[server1])# port 55
```

To reset the port number of the backend content rule, enter:

```
(config-service[server1])# no port
```

# Activating the SSL Service

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

Before activating an SSL service:

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

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

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

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

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

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

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

No modifications may be made to an active SSL proxy list. If modifications are necessary, first suspend the ssl service to make changes to the SSL proxy list entries.

To activate service *ssl_serv1*, enter:

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

## Suspending the SSL Service

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

You must suspend a service prior to modifying an SSL proxy list.

To suspend service ssl_*serv1*, enter:

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

# Configuring an SSL Content Rule

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

This section covers:

- Configuring a Virtual SSL Service Content Rule
- Configuring a Backend Service Content Rule

## Configuring a Virtual SSL Service Content Rule

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

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

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

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

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

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

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

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

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

# Configuring a Backend Service Content Rule

For an HTTP server or backend SSL server content rule, ensure that each VIP address and port configured in the rule matches a VIP address and port configured in the cipher suite parameter for a virtual SSL server entry in the SSL proxy list (see the "Specifying Cipher Suites" section).

For a backend server, you can specify a Layer 5 cookie or URL rule. The information in the rule finds a sticky server to use or load balances a new server for a new client request.

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

# Showing SSL Proxy Configuration Information

Use the **show ssl-proxy-list** command to display information about SSL proxy lists. You can display general information about all SSL proxy lists or detailed information about a specific SSL proxy list.

Enter the **show ssl-proxy-list** commands from the specified command modes to display configuration information for an SSL proxy list:

- **show ssl-proxy-list**:
  - In ssl-proxy-list mode, this command displays detailed configuration information for the specified SSL proxy list.
  - In global, content, owner, service, SuperUser, and User modes, this command displays general configuration information for all existing SSL proxy lists.
- **show ssl-proxy-list** [**ssl-server**|**backend-server**] {*number*} - Displays detailed configuration information for the SSL proxy list and the virtual SSL servers or backend servers in the list. Optionally, you can specify an SSL or backend server number to display its configuration information. This command is available in ssl-proxy-list mode.
- **show ssl-proxy-list** *list_name* - Displays detailed configuration information for the specified SSL proxy list and all virtual SSL servers associated with the list. This command is available in global, content, owner, service, SuperUser, and User modes.
- **show ssl-proxy-list** *list_name* [**ssl-server**|**backend-server**] {*number*} - Displays detailed configuration information for the SSL proxy list and all virtual SSL servers or backend servers in the list. Optionally, you can specify an SSL or backend server number to display its configuration information. This command is available in global, content, owner, service, SuperUser, and User modes.

To view general information about all configured SSL proxy lists, enter:

```
# show ssl-proxy-list
```

Table 10-15 describes the fields in the **show ssl-proxy-list** output.

***Table 10-15 Field Descriptions for the show ssl-proxy-list Command***

| Field | Description |
|---|---|
| Name | The name of the SSL proxy list |
| Description | The description for the SSL proxy list |
| State | The state of the SSL proxy list (active or suspended) |
| Services Associated | The number of services associated with the SSL proxy list |
| Rules Associated | The number of content rules associated with the SSL proxy list |

For example, to display detailed configuration information about *ssl_list1* from the ssl-proxy-list mode, enter:

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

To display detailed configuration information about *ssl_list1* from global configuration mode, enter:

```
(config)# show ssl-proxy-list ssl_list1
```

Table 10-16 describes the fields in the **show ssl-proxy-list** *list_name* output.

***Table 10-16 Field Descriptions for the show ssl-proxy-list list name Command***

| Field | Description |
|---|---|
| Description | The description for the SSL proxy list |
| SSL-Server | |
| Number of SSL-Servers | The total number of virtual SSL servers specified for the SSL proxy list |
| SSL-Server | A unique number for the virtual SSL server |
| Number of Backend-Servers | The total number of backend servers specified for the SSL proxy list |
| Backend-server | A unique number for the backend server |
| VIP Address | The VIP address for the virtual SSL or backend server (corresponding to an SSL proxy list) |

*Table 10-16 Field Descriptions for the show ssl-proxy-list list name Command*

| Field | Description |
|---|---|
| VIP Port | The virtual TCP port for the virtual SSL or backend server (corresponding to an SSL proxy list) |
| RSA Certificate | The name of the RSA certificate |
| RSA Keypair | The name of the RSA key |
| DSA Certificate | The name of the DSA certificate |
| DSA Keypair | The name of the DSA key pair |
| DH Param | The name of the Diffie-Hellman parameter association |
| Session Cache Timeout | The period of time an SSL session ID remains valid before the CSS requires the full SSL handshake to establish a new SSL connection |
| SSL Version | The specified SSL (version 3.0), TLS (version 1.0), or SSL and TLS protocol in use |
| Re-handshake Timeout | The period of time the CSS waits before initiating an SSL rehandshake message |
| Re-handshake Data | The maximum amount of data to be exchanged between the CSS and the client, after which the CSS transmits the SSL handshake message and reestablishes the SSL session |
| Virtual TCP Inactivity Timeout | The time period that the CSS waits before terminating a TCP connection with a client when there is little or no activity occurring on the connection |
| Virtual TCP Syn Timeout | The time period that the CSS waits before terminating a TCP connection with a client that has not successfully completed the TCP three-way handshake with the CSS prior to transferring data |
| Server TCP Inactivity Timeout | The time period that the CSS waits before terminating a TCP connection with a server when there is little or no activity occurring on the connection |

*Table 10-16 Field Descriptions for the show ssl-proxy-list list name Command*

| Field | Description |
|---|---|
| Server TCP Syn Timeout | The time period that the CSS waits before terminating a TCP connection with a server that has not successfully completed the TCP three-way handshake with the CSS prior to transferring data |
| Cipher Suite(s) | The name of the cipher suite(s) assigned to the SSL content rule (see Table 10-14 for a list of all supported cipher suites and values for the specific SSL server) |
| Weight | The priority assigned to the cipher suite |
| Port | The TCP port of the backend content rule through which the backend HTTP connections are sent |
| URL Rewrite Rule(s) | |
| Number | The number of the URL rewrite rule in the SSL server |
| Rule | The domain name of the URL to be redirected |
| SSL Port | The port used for rewriting the HTTP Header Location field to contain an HTTPS location when the URL rewrite rule matches |
| Clear Port | The port used for performing the URL rewrite rule match |
| Server | The IP address assigned to the backend content rule used with the cipher suite |

# Showing SSL URL Rewrite Statistics

Use the **show ssl urlrewrite** command to view the URL rewrite rule statistics for one or more SSL modules. This command displays statistics related to the number of flows received and evaluated by the SSL module, and the number of HTTP 300-series redirects found and then rewritten.

The syntax for this command is:

**show ssl urlrewrite** {**slot** *number*}

The **slot** *number* option displays URL rewrite statistics for a specific SSL module in the CSS 11503 or CSS 11506 chassis (assuming more than one module is installed). The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl urlrewrite** command displays URL rewrite statistics for all SSL modules in the chassis.

For example, to view URL rewrite statistics for all SSL modules, enter:

```
# show ssl urlrewrite
```

For example, to view URL rewrite statistics for the SSL module in slot 5 of the CSS 11506, enter:

```
# show ssl urlrewrite slot 5
```

Table 10-17 describes the fields in the **show ssl urlrewrite** output.

***Table 10-17 Field Descriptions for the show ssl urlrewrite Command***

| Field | Description |
|---|---|
| Virtual | The VIP address for the virtual SSL server. |
| Port | The virtual TCP port for the virtual SSL server. |
| Searches | The total number of flows received from the backend server and evaluated by the SSL module to search for the presence of HTTP 300-series redirects. |
| Redirects Found | The total number of flows examined by the SSL module for which an HTTP 300-series redirect was detected. |
| Redirects Rewritten | The total number of flows examined by the SSL module for which an HTTP 300-series redirect was found matching one of the configured URL rewrite rules. This number represents the total number of redirects that have been rewritten for this VIP address. |

# Showing SSL Module Statistics

Use the **show ssl statistics** command to view the statistics for the cryptography components on one or more SSL modules. If you do not specify any options for this command, SSL statistics appear for all SSL modules in the CSS chassis.

The syntax for this command is:

**show ssl statistics** {*component*} {**slot** *number*}

The options and variables are:

- *component* - Selects a specific component in the SSL module to display statistics. The components include:
  - **ssl-proxy-server** - Displays counter statistics for the SSL proxy list component that provides SSL termination in the SSL module
  - **crypto** - Displays counter statistics for the cryptography chip
  - **ssl** - Displays counter statistics for the SSL server counter
- **slot** *number* - Displays statistics for a component in a specific SSL module in the CSS chassis (assuming more than one module is installed). Specify **slot** *number* after each **show ssl statistics** command. The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl statistics** command displays statistics for all installed SSL modules.

For example, to view all SSL statistics for the SSL module in slot 5 of the CSS chassis, enter:

```
# show ssl statistics slot 5
```

Table 10-18 describes the fields in the **show ssl statistics** output.

*Table 10-18 Field Descriptions for the show ssl statistics Command*

| Field | Description |
|---|---|
| Component | Indicates the specific component in the SSL module for which statistics are displayed. The SSL statistic functions include:<br>• **ssl-proxy-server** - Displays counter statistics for the SSL proxy list component that provides SSL termination in the SSL module.<br>• **crypto** - Displays counter statistics for the cryptography chip in the SSL module.<br>• **ssl** - Displays counter statistics for the SSL server counter. |
| Slot | Indicates the slot number of the SSL module for which statistics are displayed. Valid slots: 2 (CSS 11501), 2 and 3 (CSS 11503), or 2 to 6 (CSS 11506). |
| **SSL Proxy List Statistics** | |
| Handshake started for incoming SSL connections | Number of times the handshake process was initiated for incoming SSL connections from a client to the SSL module. |
| Handshake completed for incoming SSL connections | Number of times the handshake process was completed for incoming SSL connections from a client to the SSL module. |
| Handshake started for outgoing SSL connections | Number of times the handshake process was initiated for outgoing SSL connections from the SSL module to a client. |
| Handshake completed for outgoing SSL connections | Number of times the handshake process was completed for outgoing SSL connections from a client to the SSL module. |
| **Crypto Statistics** | |
| RSA Private | Number of RSA private key calculations requested. |
| RSA Public | Number of RSA public key calculations requested. |

*Table 10-18 Field Descriptions for the show ssl statistics Command (continued)*

| Field | Description |
|---|---|
| DH Shared | Number of Diffie-Hellman shared secret key calculations requested. |
| DH Public | Number of Diffie-Hellman public key calculations requested. |
| DSA Sign | Number of DSA signings requested. |
| DSA Verify | Number of DSA verifications requested. |
| SSL MAC | Number of SSL MAC calculations requested. |
| TLS HMAC | Number of TLS HMAC calculations requested. |
| 3DES | Number of 3 DES calculations requested. |
| ARC4 | Number of ARC4 calculations requested. |
| HASH | Number of pure hash calculations requested. |
| RSA Private Failed | Number of RSA private key calculations that failed. |
| RSA Public Failed | Number of RSA public key calculations that failed. |
| DH Shared Failed | Number of Diffie-Hellman shared secret key calculations that failed. |
| DH Public Failed | Number of Diffie-Hellman public key calculations that failed. |
| DSA Sign Failed | Number of DSA signings that failed. |
| DSA Verify Failed | Number of DSA verifications that failed. |
| SSL MAC Failed | Number of SSL MAC calculations that failed. |
| TLS HMAC Failed | Number of TLS HMAC calculations that failed. |
| 3DES Failed | Number of 3 DES calculations that failed. |
| ARC4 Failed | Number of ARC4 calculations that failed. |
| HASH Failed | Number of pure hash calculations that failed. |
| Hardware Device Not Found | Number of times that a call was made to the cryptography hardware and no hardware acceleration device was available. |

*Table 10-18 Field Descriptions for the show ssl statistics Command (continued)*

| Field | Description |
|---|---|
| Hardware Device Timed Out | Number of times the cryptography hardware did not complete an acceleration request within the specified time. This function is not currently implemented. This counter should always be 0. |
| Invalid Crypto Parameter | Number of times a hardware acceleration function was requested with an invalid parameter from the CSS. Invalid parameters include an invalid bit length for the operation, a buffer that is not a multiple of 4 bytes in length, a buffer that does not begin on an even 4-byte boundary, requesting an operation on a buffer with too many fragments or too few fragments (such as with no input), or requesting an illegal (nonsense) function. |
| Hardware Device Failed | Number of times the hardware acceleration device failed. This counter only increments on a DMA error. |
| Hardware Device Busy | Number of times he hardware acceleration device was busy and could not accept an acceleration request. |
| Out Of Resources | Number of times no hardware buffers were available and the cryptography hardware could not accept an acceleration request. |
| Cancelled -- Device Reset | Number of cancelled status returns due to a CSS reboot. |
| **SSL Statistics** | |
| RSA Private Decrypt calls | Number of RSA private decryption calls. |
| RSA Public Decrypt calls | Number of RSA public encryption calls. |
| DH Compute key calls | Number of Diffie-Hellman Compute key calls. |
| DH Generate key calls | Number of Diffie-Hellman Generate key calls. |
| DSA Verify calls | Number of DSA Verifications calls. |
| DSA Sign calls | Number of DSA Signing calls. |

*Table 10-18 Field Descriptions for the show ssl statistics Command (continued)*

| Field | Description |
|---|---|
| MD5 raw hash calls | Number of MD5 pure hash calls. |
| SHA1 raw hash calls | Number of SHA1 pure hash calls. |
| 3-DES calls | Number of 3-DES calls. |
| RC4 calls | Number of RC4 calls. |
| SSL MAC (MD5) calls | Number of SSL Message Authentication Code (MAC) computations using MD5 algorithm. |
| SSL MAC (SHA1) calls | Number of SSL MAC computations using SHA algorithm. |
| TLS MAC (MD5) calls | Number of TLS MAC computations using MD5 algorithm. |
| TLS MAC (SHA1) calls | Number of TLS MAC computations using SHA algorithm. |
| Level 1 Alerts Received | Number of Level 1 alerts received. |
| Level 2 Alerts Received | Number of Level 2 alerts received. |
| Level 1 Alerts Sent | Number of Level 1 alerts transmitted. |
| Level 2 Alerts Sent | Number of Level 2 alerts transmitted. |
| SSL received bytes from TCP | Number of bytes SSL received from TCP. |
| SSL transmitted bytes to TCP | Number of bytes SSL transmitted to TCP. |
| SSL received Application Data bytes | Number of Application Data bytes received by the SSL module. |
| SSL transmitted Application Data bytes | Number of Application Data bytes transmitted by the SSL module. |
| SSL received non-application data bytes | Number of non-application data (handshake, alert, and change cipher) bytes received by the SSL module. |
| SSL transmitted non-application data bytes | Number of non-application data (handshake, alert, and change cipher) bytes transmitted by the SSL module. |

*Table 10-18 Field Descriptions for the show ssl statistics Command (continued)*

| Field | Description |
|---|---|
| RSA Private Decrypt failures | Number of RSA Private Decrypt calls that failed. |
| MAC failures for packets received | Number of times the MAC could not be verified for the incoming SSL messages. |
| Rehandshake TimerAlloc failed | Number of times the SSL module was unable to allocate the Rehandshake Timer. |

## Clearing SSL Statistics

Use the **clear ssl statistics** command to clear the SSL statistics counters for all SSL modules in the CSS chassis. The reset statistics appear as 0 in the **show ssl statistics** display.

To clear SSL statistics counters for a specific module in either the CSS 11503 or CSS 11506, use the **clear ssl statistics** command and specify the **slot** *number* following the command. The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506).

To clear the SSL statistics counter, enter:

```
# clear ssl statistics
```

# Showing SSL Flows

Use the **show ssl flows** command to display information about the active flows for each VIP address, port, and SSL module. The output displays TCP proxy flows, active SSL flows (a subset of TCP proxy flows), and SSL flows occurring during the handshake phase of the protocol (a subset of active SSL flows).

The syntax for this command is:

**show ssl flows** {**slot** *number*}

The **slot** *number* option displays information about the active flows for a specific SSL module in the CSS chassis (assuming more than one module is installed). The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl flows** command displays statistics for all installed SSL modules.

To view SSL flows for all SSL modules in the CSS, enter:

```
# show ssl flows
```

To view SSL flows for a specific SSL module in the CSS chassis (for example, installed in slot 5), enter:

```
# show ssl flows slot 5
```

Table 10-19 describes the fields in the **show ssl flows** output.

***Table 10-19 Field Descriptions for the show ssl flows Command***

| Field | Description |
|---|---|
| SSL Acceleration Flows for Slot | The slot number of the SSL module for which flows are displayed. Valid slots: 2 (CSS 11501), 2 and 3 (CSS 11503), or 2 to 6 (CSS 11506). |
| Virtual | Virtual address of the ssl-server. |
| Port | Virtual TCP port of the ssl-server. |
| TCP Proxy Flows | Number of TCP connections that are currently being proxied through the SSL virtual IP address. These connections could either be in:<br>• The TCP handshake or teardown phase and, therefore, not carrying any SSL traffic<br>• The Established TCP phase and carrying SSL traffic |

*Table 10-19 Field Descriptions for the show ssl flows Command (continued)*

| Field | Description |
|---|---|
| Active SSL Flows | Current number of TCP Proxy Flows that are carrying active SSL connections. These flows are the Established TCP connections in which an SSL Client Hello message has been received by the CSS. The SSL flows remain in this active state until the teardown process is initiated, either by sending or receiving an SSL Alert message. The Active SSL Flows number is a subset of the TCP Proxy Flows column. |
| SSL Flows in Handshake | The current number of Active SSL Flows that are in the handshake phase of the SSL protocol but are not yet sending data. This means that an SSL Client Hello message has been received by the CSS but the final finished message still has not been sent. The SSL Flows in Handshake number is a subset of the Active SSL Flows column. |

# Examples of SSL Proxy Configurations

This section 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 Backend SSL Servers
- SSL Full Proxy Configuration — One SSL Module

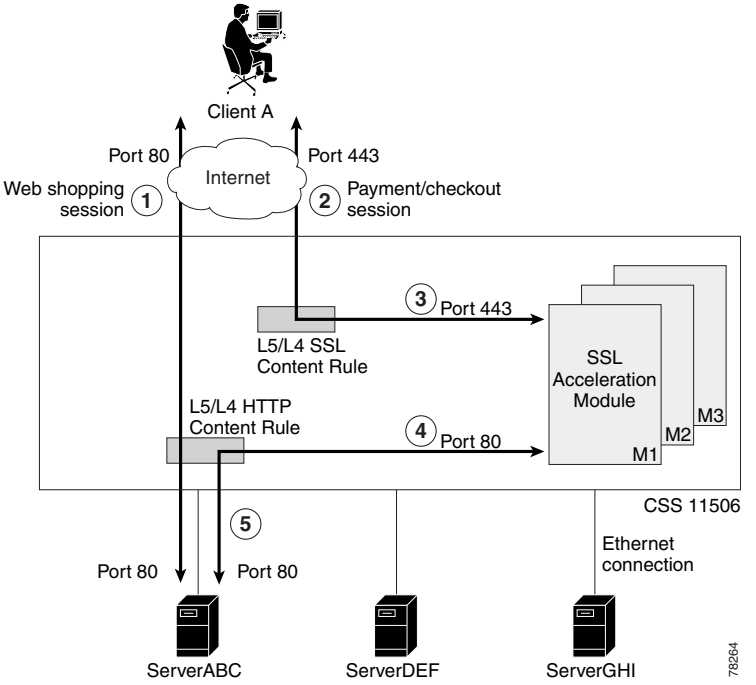
# Processing of SSL Flows by the SSL Module

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 10-2 illustrates a CSS with three SSL modules (M1, M2, and M3) configured to off load the SSL traffic from the backend servers (ServerABC, ServerDEF, and ServerGHI). Figure 10-2 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 10-2 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.

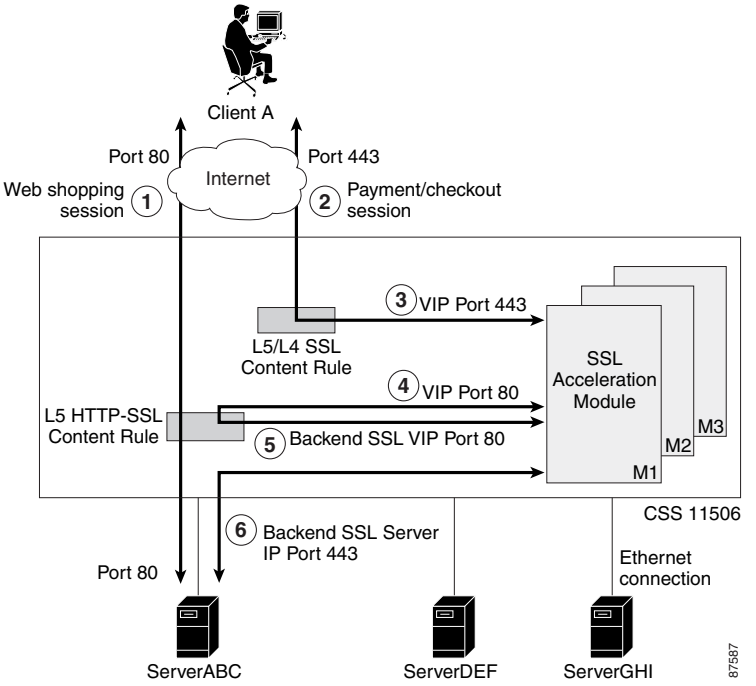
Multiple TCP connections can comprise an entire SSL session. For each of those connections, the same process takes place among the client, SSL module, and server. The SSL Session ID maintains the stickiness between the client and the SSL module and the cookie maintains the stickiness between the SSL 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 backend session ID cache is 4096 entries and is not configurable.

When you configure a backend SSL server on the CSS (Figure 10-3), 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 10-3, the SSL module initiates an HTTP connection to an HTTP-SSL content rule with services to a backend SSL server.

*Figure 10-3 CSS Configuration with a Backend SSL Server*




In step 5 shown in Figure 10-3, the CSS directs the clear text traffic back to the SSL module through an IP address that maps directly to a backend SSL server. The SSL module terminates the clear text connection.

In step 6 of Figure 10-3, the SSL module re-encrypts the traffic and establishes an SSL connection to the backend SSL server. The SSL module sends the traffic through the CSS to the selected backend SSL server.

# SSL Transparent Proxy Configuration — One SSL Module

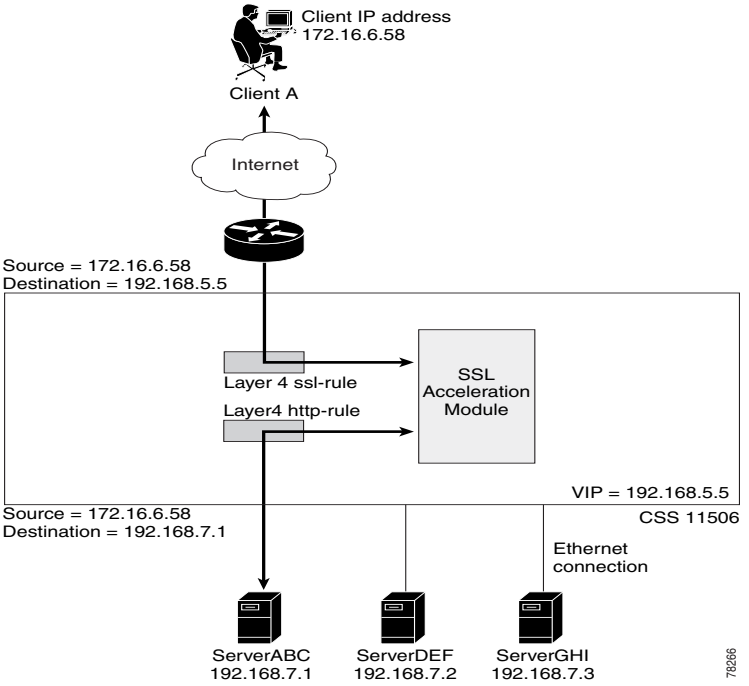
An SSL transparent proxy server is a proxy server that preserves the client's IP address as the source IP address for the backend connection to the server. When you configure an SSL transparent proxy on the CSS, the CSS intercepts and redirects outbound client requests to an HTTP server on the network 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 10-4 illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in Figure 10-4 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 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 10-4 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
```

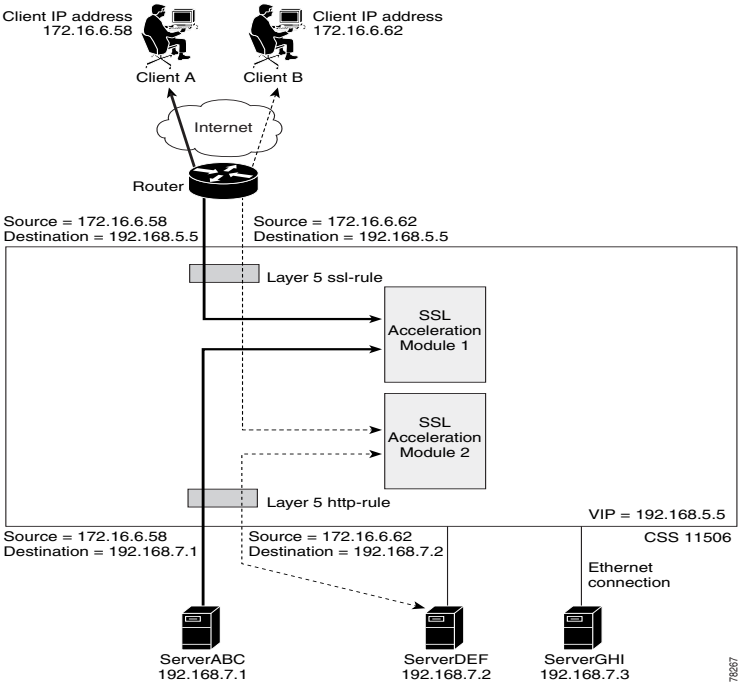
# SSL Transparent Proxy Configuration — Two SSL Modules

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 10-6 illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in Figure 10-6 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 10-5 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
```

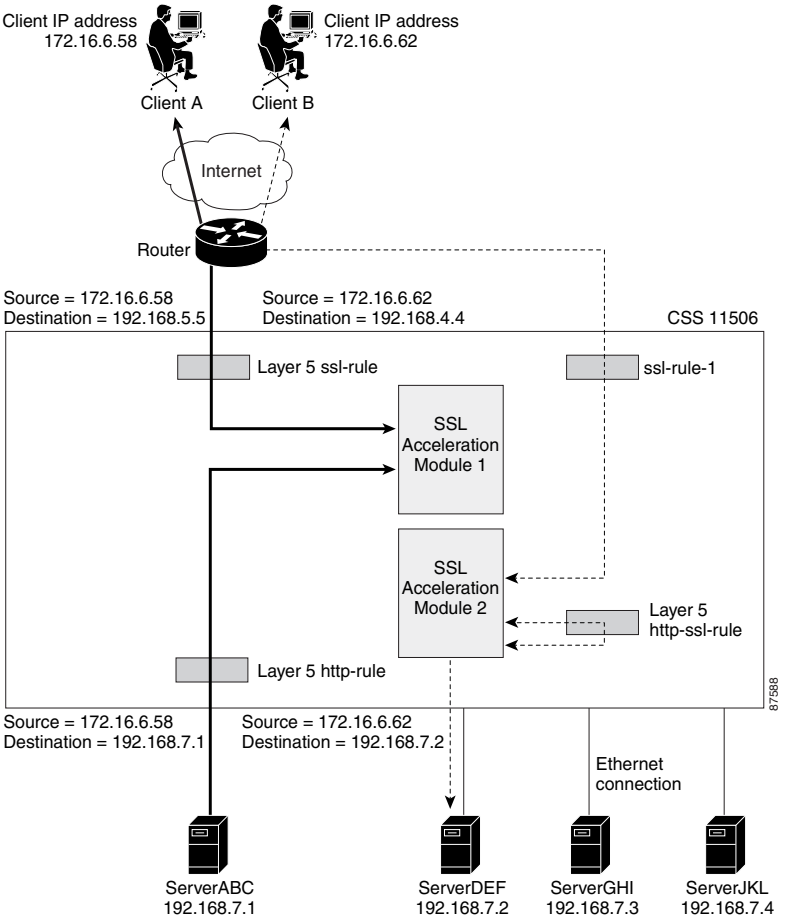
# SSL Transparent Proxy Configuration — HTTP and Backend SSL Servers

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 backend 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 backend SSL server.

In Figure 10-6, 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 10-6 SSL Transparent Proxy Configuration - HTTP and Backend SSL Servers*



```
!************************** 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
  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
  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 backend connection to the HTTP server using a different source IP address than of the client. This configuration does not preserve the client's IP address for the backend connection to the HTTP server.

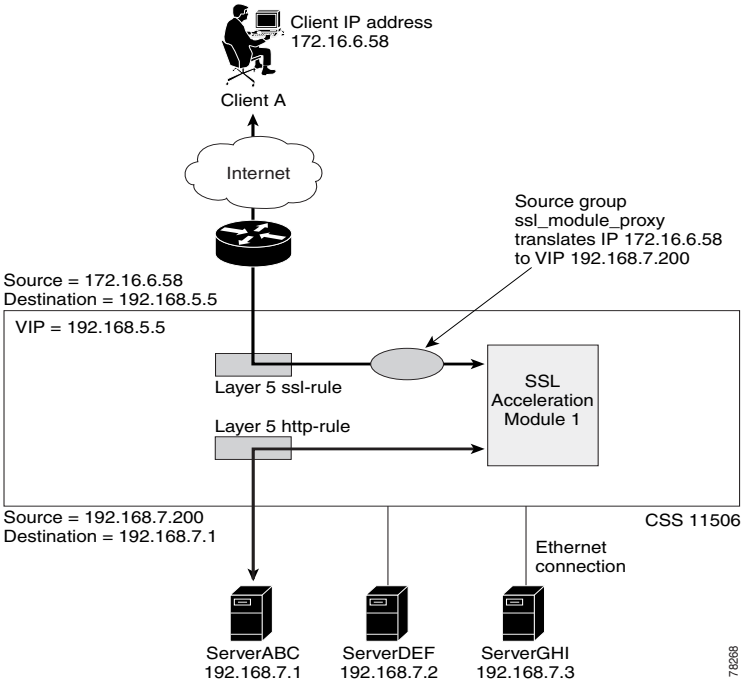
This section provides an example configuration for an SSL full proxy between a client, a CSS with a single SSL module, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). A Layer 5 sticky content rule is used in the configuration. For the CSS to implement a full proxy configuration with an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

Figure 10-7 illustrates this full proxy configuration.

For purposes of illustration, the configuration example in Figure 10-7 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 10-7 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 192.168.5.254 255.255.255.0

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
```