CHAPTER 3

Configuring SSL Termination

Note The information in this chapter applies to both the ACE module and the ACE appliance unless otherwise noted. The features in this chapter apply to IPv4 and IPv6 unless otherwise noted.

This chapter describes the steps required to configure a context on the Cisco ACE Application Control Engine as a virtual SSL server for SSL termination. It contains the following major sections:

- Example of an SSL Termination Configuration

Note To verify that the SSL connection from a client to the ACE was properly initiated, you can monitor the handshake counters in the **show stats crypto server** command output (see Chapter 6, Displaying SSL Information and Statistics). The handshake counters increment for successful connections. For example, the SSLv3 Full Handshakes counter indicates that the handshake completed successfully and the SSLv3 Resumed Handshakes counter indicates that the handshake resumed successfully by using a session ID. When traffic is flowing, those numbers should increment. If there are failures, then the alerts sent and received counters should also increment.

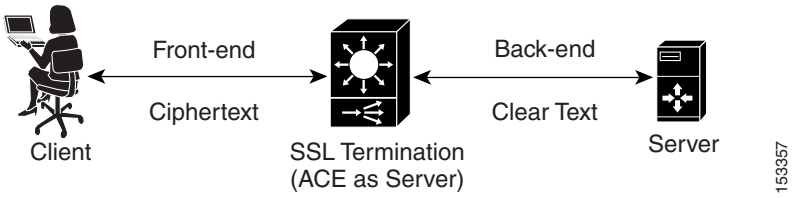
SSL Termination Overview

SSL termination occurs when the ACE, acting as an SSL proxy server, terminates an SSL connection from a client and then establishes a TCP connection to an HTTP server. When the ACE terminates the SSL connection, it decrypts the ciphertext from the client and transmits the data as clear text to an HTTP server.

Figure 3-1 shows the following network connections in which the ACE terminates the SSL connection with the client:

- Client to ACE—SSL connection between a client and the ACE acting as an SSL proxy server
- ACE to Server—TCP connection between the ACE and the HTTP server

Figure 3-1 SSL Termination with a Client



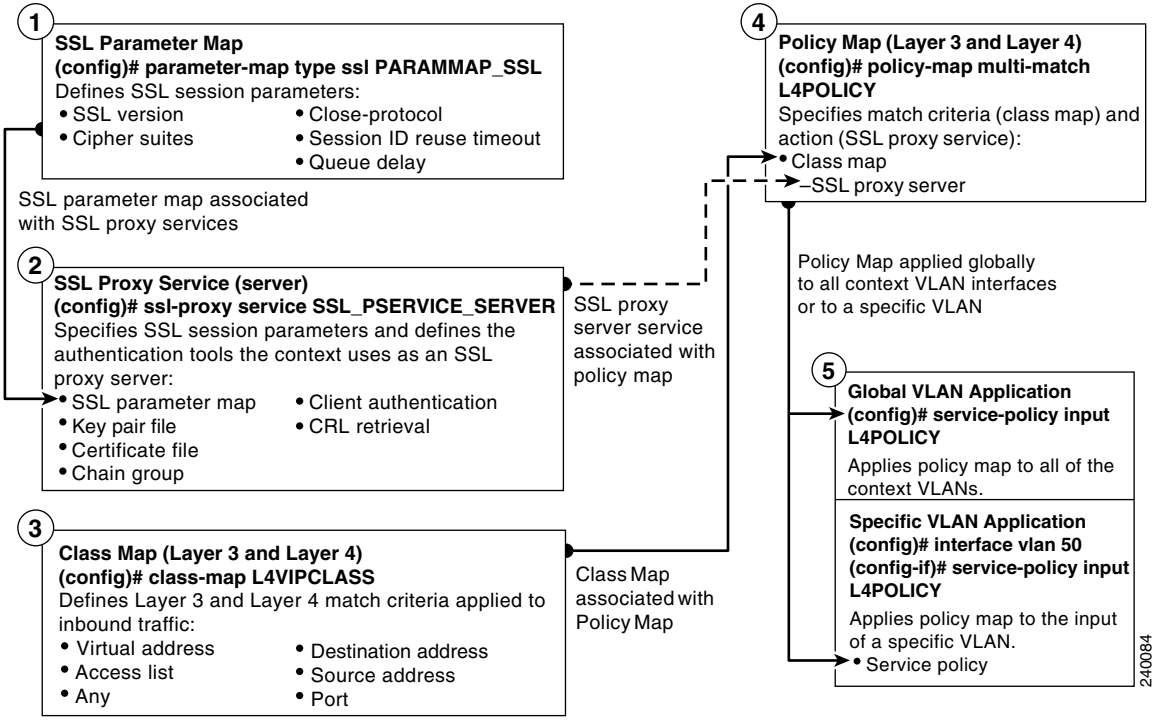
The ACE uses parameter maps, SSL proxy services, and class maps to build the policy maps that determine the flow of information between the client, the ACE, and the server. SSL termination is a Layer 3 and Layer 4 application because it is

based on the destination IP addresses of the inbound traffic flow from the client. For this type of application, you create a Layer 3 and Layer 4 policy map that the ACE applies to the inbound traffic.

When configuring a policy map for SSL termination, you associate a parameter map and SSL proxy server service with the policy map to define the SSL session parameters and client/server authentication tools, such as the certificate and RSA key pair. You also associate a class map with the policy map to define the virtual SSL server IP addresses that the destination IP address of the inbound traffic must match. When a match occurs, the ACE negotiates with the client to establish an SSL connection. You can define a maximum of 250 virtual SSL servers for a single class map.

Figure 3-2 provides a basic overview of the process required to build and apply the Layer 3 and Layer 4 policy map that the ACE uses for SSL termination. The figure also shows how you associate the various components of the policy map configuration with each other.

Figure 3-2 Basic SSL Termination Configuration Flow Diagram



ACE SSL Termination Configuration Prerequisites

Before configuring your ACE for SSL operation, you must first configure it for server load balancing (SLB). During the real server and server farm configuration process, when you associate a real server with a server farm, ensure that you assign an appropriate port number for the real server. The default behavior by the ACE is to automatically assign the same destination port that was used by the inbound connection to the outbound server connection if you do not specify a port.

For example, if the incoming connection to the ACE is a secure client HTTPS connection, the connection is typically made on port 443. If you do not assign a port number to the real server, the ACE will automatically use port 443 to connect

to the server, which results in the ACE making a clear-text HTTP connection over port 443. In this case, you would typically define an outbound destination port of 80, 81, or 8080 for the back-end server connection.

During the SLB traffic policy configuration process, you create the following configuration objects:

- Layer 7 class map
- Layer 3 and Layer 4 class map
- Layer 7 policy map
- Layer 3 and Layer 4 policy map

After configuring SLB, you modify the existing SLB class maps and policy maps with the SSL configuration requirements described in this guide for SSL termination.

To configure your ACE for SLB, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

SSL Termination Configuration Quick Start

Table 3-1 provides a quick overview of the steps required to configure the ACE for SSL termination. Each step includes the CLI command or a reference to the procedure required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 3-1.

Note The following quick start does not include a procedure for creating a parameter map as shown in Figure 3-2. The ACE uses the default parameter map settings as described in Table 3-2.

Table 3-1 SSL Termination Configuration Quick Start

Task and Command Example

1. If you are operating in multiple contexts, observe the CLI prompt to verify that you are operating in the desired context. If necessary, log directly in to, or change to, the correct context.
 	
`host1/Admin# changeto C1`
`host1/C1#`
 	
The rest of the examples in this table use the Admin context. For details on creating contexts, see the *Virtualization Guide, Cisco ACE Application Control Engine*.

2. Enter configuration mode.
 	
`host1/Admin# config`
`host1/Admin(config)#`

3. Create an SSL proxy server service to define the handshake parameters that the ACE, acting as an SSL server, applies to a policy map.
 	
`host1/Admin(config)# ssl-proxy service SSL_PSERVICE_SERVER`
`host1/Admin(config-ssl-proxy)#`

4. Configure the SSL proxy server service by defining the certificate and corresponding RSA key pair.
 	
`host1/Admin(config-ssl-proxy)# key MYRSAKEY_SERVER`
`host1/Admin(config-ssl-proxy)# cert MYCERT_SERVER`
`host1/Admin(config-ssl-proxy)# exit`
`host1/Admin(config)#`

5. Create a Layer 3 and Layer 4 class map and configure it with the input traffic match criteria as required.
 	
`host1/Admin(config)# class-map L4VIPCLASS`
`host1/Admin(config-cmap)# match virtual-address 2001:DB8:1::24 tcp any`
or
`host1/Admin(config-cmap)# match virtual-address 192.168.10.24 tcp any`
`host1/Admin(config-cmap)# exit`
`host1/Admin(config)#`

Table 3-1 SSL Termination Configuration Quick Start (continued)

Task and Command Example

6. Create a policy map and associate the class map created in Step 5 with it.
`host1/Admin(config)# policy-map multi-match L4POLICY`
`host1/Admin(config-pmap)# class L4VIPCLASS`
`host1/Admin(config-pmap-c)#`

7. Associate the SSL proxy server service created in Step 3 with the policy map.
`host1/Admin(config-pmap-c)# ssl-proxy server SSL_PSERVICE_SERVER`
`host1/Admin(config-pmap-c)# exit`
`host1/Admin(config-pmap)# exit`
`host1/Admin(config)#`

8. Apply the policy map to the input traffic of the desired interface as follows:
Apply the policy map globally to all context VLANs.
`host1/Admin(config)# service-policy input L4POLICY`
Apply the policy map to a specific VLAN.
`host1/Admin(config)# interface vlan 50`
`host1/Admin(config-if)# service-policy input L4POLICY`

9. Display the running configuration to verify the information that you just added is configured properly. `host1/Admin(config-if)# do show running-config`
10. (Optional) Save the configuration changes to flash memory by copying the running configuration to the startup configuration. `host1/Admin(config-if)# do copy running-config startup-config`

Creating and Defining an SSL Parameter Map

An SSL parameter map defines the SSL session parameters that the ACE applies to an SSL proxy service. Creating an SSL parameter map allows you to apply the same SSL session parameters to different proxy services. Table 3-2 describes each SSL session parameter with its default value.

Table 3-2 SSL Session Parameters of an SSL Parameter Map

SSL Session Parameter	Description	Default Value/Behavior
Cipher suites	Defines the cipher suites that the ACE supports during the SSL handshake (see Table 3-3 for a list of available cipher suites that the ACE supports)	The ACE supports all of the available cipher suites
Authentication-failure	When client authentication is enabled on the ACE, this parameter allows the ACE to continue setting up the front-end connection in an SSL termination configuration when it encounters a client certificate failure.	The ACE terminates the SSL handshake when a client certificate failure is encountered.
CDP-errors ignore	When the **crl best-effort** command is configured on the ACE, this parameter allows the ACE to ignore authentication failures due to CDP errors.	Disabled
Close-protocol	Defines how the ACE executes close-notify messages	**none**—The ACE sends a close-notify alert message to its peer when closing a session but has no expectation of receiving one back from the peer
Purpose-check disabled	When this command is configured, this parameter disables ACE from performing purpose checking on certificates during authentication.	Enabled

Table 3-2 SSL Session Parameters of an SSL Parameter Map (continued)

SSL Session Parameter	Description	Default Value/Behavior
Rehandshake	Enables rehandshake, allowing the ACE to send an SSL HelloRequest message to its peer to restart SSL handshake negotiation	Disabled
Version	Defines the SSL and TLS versions that the ACE supports during the SSL handshake	The ACE supports versions SSL3 and TLS1
Queue delay time	Defines the amount of time that the ACE keeps packet data from the server before encrypting it for the client	Disabled
Session cache timeout	Defines the amount of time that the SSL session ID remains valid before the ACE requires a new SSL handshake to establish a new SSL session.	Disabled
Expired CRL	Defines whether the ACE rejects all incoming client certificates if the CRL is expired.	Disabled

Note If you want an SSL proxy service to use the default values for the SSL session parameters, you do not need to create an SSL parameter map or associate one with the proxy service. When you do not associate a parameter map with the SSL proxy service, the ACE automatically applies the default values for the session parameters listed in Table 3-2 to the proxy service.

You can create an SSL parameter map by using the **parameter-map type ssl** command in configuration mode.

The syntax of this command is as follows:

parameter-map type ssl *parammap_name*

The *parammap_name* argument is the name of the SSL parameter map. Enter an unquoted alphanumeric string with a maximum of 64 characters.

For example, to create the SSL parameter map PARAMMAP_SSL, enter:

```
host1/Admin(config)# parameter-map type ssl PARAMMAP_SSL
```

After you create an SSL proxy parameter map, the CLI enters parameter map SSL configuration mode.

```
host1/Admin(config-parammap-ssl)#
```

If you exit out of parameter map SSL configuration mode without defining any of its SSL session parameters, the ACE configures the parameter map with the default values listed in Table 3-2.

To delete an existing SSL parameter map, enter:

```
host1/Admin(config)# no parameter-map type ssl PARAMMAP_SSL
```

This section contains the following topics:

- Defining a Description of the SSL Parameter Map
- Adding a Cipher Suite
- Continuing SSL Session Setup with Client Certificate Failures
- Configuring the ACE to Ignore Authentication Failures Due to CDP Errors
- Defining the Close-Protocol Behavior
- Disabling Purpose Checking on the Certificates
- Enabling SSL Session Rehandshake
- Defining the SSL and TLS Version
- Configuring the SSL Queue Delay
- Configuring the SSL Session Cache Timeout
- Rejecting Expired CRL Client Certificates

Defining a Description of the SSL Parameter Map

You can provide a brief summary of the SSL parameter map by using the **description** command in SSL parameter map configuration mode. The syntax of this command is as follows:

description *text*

For the *text* argument, enter an unquoted text string with a maximum of 240 alphanumeric characters including spaces.

For example, to specify a description of an SSL parameter map, enter the following command:

```
host1/Admin(config)# parameter-map type ssl PARAMMAP_SSL
host1/Admin(config-parammap-conn)# description SSL parameter map
```

To remove the description from the SSL parameter map, enter:

```
host1/Admin(config-parammap-conn)# no description
```

Adding a Cipher Suite

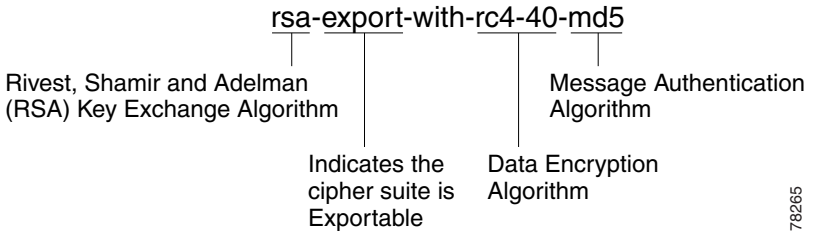
The SSL protocol supports a variety of different cryptographic algorithms, or ciphers, for use in operations such as the following:

- Authenticating the server and client to each other
- Transmitting certificates
- 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 that 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 suite they will use to authenticate each other, transmit certificates, and establish session keys.

As shown in Figure 3-3, a cipher suite consists of the following three algorithms: key exchange algorithm, data encryption algorithm, and message authentication (hash) algorithm.

Figure 3-3 Cipher Suite Algorithms



Note Exportable cipher suites are those cipher suites that are not 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.

To define each of the cipher suites that you want the ACE to support during a secure session, use the **cipher** command in ssl parameter-map configuration mode. The cipher suite that you choose depends on your environment and security requirements and must correlate to the certificates and keys that you have loaded on the ACE.

Note By default, the ACE supports all of the cipher suites listed in Table 3-3. The default setting works only when you do not configure the SSL parameter map with any specific ciphers. To return to using the all cipher suites setting, you must delete each specifically defined cipher from the parameter map by using the **no** form of the command.

The syntax of this command is as follows:

cipher *cipher_name* [**priority** *cipher_priority*]

The keywords and arguments are as follows:

- *cipher_name*—Name of the cipher suite that you want the ACE to support. Table 3-3 lists the cipher suites that the ACE supports. Enter one of the supported cipher suites from the table.
- **priority**—(Optional) Assigns a priority level to the cipher suite. The priority level represents the preference ranking of the cipher suite, with 10 being the most preferred and 1 being the least preferred. By default, all configured cipher suites have a priority level of 1. When negotiating which cipher suite to use, the ACE selects from the client list based on the cipher suite configured with the highest priority level. A higher priority level will bias towards the specified cipher suite. For SSL termination applications, the ACE uses the priority level to match cipher suites in the client's ClientHello handshake message. For SSL initiation applications, the priority level represents the order in which the ACE places the cipher suites in its ClientHello handshake message to the server.
- *cipher_priority*—Priority level of the cipher suite. Enter an integer from 1 to 10. The default is 1.

For example, to add the cipher suite rsa_with _aes_128_cbc_sha with a priority 2 level, enter:

```
host1/Admin(config)# parameter-map type ssl PARAMMAP_SSL
host1/Admin(config-parammap-ssl)# cipher rsa_with_aes_128_cbc_sha
priority 2
```

Repeat the **cipher** command for each cipher suite that you want to include in the SSL parameter map.

To delete a cipher suite from the SSL parameter map, enter:

```
host1/Admin(config-parammap-ssl)# no cipher rsa_with_aes_128_cbc_sha
```

Table 3-3 lists the available cipher suites that the ACE supports and indicates which of the supported cipher suites are exportable from the ACE. The table also lists the authentication certificate and encryption key required by each cipher suite.

If you use the default setting in which the ACE implicitly supports all of the cipher suites listed in Table 3-3 or you explicitly define each cipher suite with equal priority and the client connection uses multiple ciphers, the ACE sends the cipher suites to its peer in the same order as they appear in the table, starting with RSA_WITH_RC4_128_MD5.

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

Table 3-3 SSL Cipher Suites Supported by the ACE

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
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
RSA_WITH_AES_128_CBC_SHA	No	RSA certificate	RSA key exchange
RSA_WITH_AES_256_CBC_SHA	No	RSA certificate	RSA key exchange
RSA_EXPORT_WITH_RC4_40_MD5	Yes	RSA certificate	RSA key exchange
RSA_EXPORT1024_WITH_RC4_56_MD5	Yes	RSA certificate	RSA key exchange
RSA_EXPORT_WITH_DES40_CBC_SHA	Yes	RSA certificate	RSA key exchange
RSA_EXPORT1024_WITH_DES_CBC_SHA	Yes	RSA certificate	RSA key exchange
RSA_EXPORT1024_WITH_RC4_56_SHA	Yes	RSA certificate	RSA key exchange

Continuing SSL Session Setup with Client Certificate Failures

By default, with client authentication enabled, when the ACE encounters one of the following client certificate failures during the setup of the front-end connection in an SSL termination configuration, it terminates the SSL handshake:

- Certificate is not yet valid
- Certificate has expired
- Unable to get issuer certificate
- Certificate is revoked
- No client certificate is sent

- Certificate signature failure
- CRL is not available during the revocation check
- CRL is expired during revocation check
- All other certificate errors

You can configure the ACE to either ignore these errors and continue the SSL handshake or perform an HTTP redirect after the handshake is complete by using the **authentication-failure** command in parameter map SSL configuration mode.

The syntax of this command is as follows:

authentication-failure {**ignore** | **redirect** *reason* {**serverfarm** *serverfarm_name* | **url** *URL_string* {**301**|**302**}}}

The keywords, arguments, and options are as follows:

- **ignore**—Ignores any certificate failure during the SSL handshake, allowing the ACE to establish the SSL connection even if a certificate failure exists. If you combine the **authentication-failure ignore** command with one or more **authentication-failure redirect** commands, the ACE redirects the individual errors that you specify and ignores the remaining errors.

  You cannot configure the **authentication-failure redirect any** command with the **authentication-failure ignore** command.

- **redirect** *reason*—Performs a redirect to the specified server farm or URL when the ACE encounters the specified *reason* argument for the certificate failure after the handshake is completed.

  For more than one failure reason, you configure an **authentication-failure redirect** command for each reason.

  If multiple failures cause a redirect, the ACE performs a redirect on the first failure that it encounters. If that failure is corrected, the ACE performs a redirect on the next failure that it encounters.

  For the *reason* argument, enter one of the following keywords:

  – **cert-not-yet-valid**—Associates a certificate that is not yet valid failure with the redirect.

  – **cert-expired**—Associates an expired certificate failure with a redirect.

  – **unknown-issuer**—Associates an unknown issuer certificate failure with a redirect.

  – **cert-revoked**—Associates a revoked certificate failure with a redirect.

  - **no-client-cert**—Associates no client certificate failure with a redirect.
  - **crl-not-available**—Associates a CRL that is not available failure with a redirect.
  - **crl-has-expired**—Associates an expired CRL failure with a redirect.
  - **cert-has-signature-failure**—Associates a certificate signature failure with a redirect.
  - **cert-other-error**—Associates a all other certificate failures with a redirect.
  - **any**—Associates any of the certificate failures with the redirect. You can configure the **authentication-failure redirect any** command with individual reasons for redirection. When you do, the ACE attempts to match one of the individual reasons before using the **any** reason. You cannot configure the **authentication-failure redirect any** command with the **authentication-failure ignore** command.

- **serverfarm** *serverfarm_name*—Specifies the name of the configured server farm for load balancing. You can configure a host or redirect serverfarm.

  When you configure a host server farm, include a real server as the sorry server. The real server must be an HTTP server and you must specify a port number. When a redirect failure occurs, the ACE forwards the client connection directly to the real server without involving another VIP.

  If you exhaust the server farm IDs while adding SSL redirects, the ACE displays the following message:

  ```
  "Number of sfarms in the config have reached the maximum limit!"
  ```

- **url** *URL_string*—Specifies the static URL path for the redirect. Enter a string with a maximum of 255 characters and no spaces.
- **301|302**—Specifies the redirect code that is sent back to the client. Enter one of the following:
  - **301**, the status code for a resource permanently moving to a new location.
  - **302**, the status code for a resource temporarily moving to a new location.

For example, to ignore client certificate failures, enter:

```
host1/Admin(config)# parameter-map type ssl SSL_PARAMMAP_SSL
host1/Admin(config-parammap-ssl)# authentication-failure ignore
```

To perform a redirect to a server farm when any client certificate failure occurs, enter:

```
host1/Admin(config-parammap-ssl)# authentication-failure redirect any
serverfarm SFARM2
```

To perform a redirect to a static URL with a 302 status code when an unknown-issuer failure occurs, enter:

```
host1/Admin(config-parammap-ssl)# authentication-failure redirect
unknown-issuer url http://www.eng.com 302
```

To reset the default behavior of terminating an SSL handshake when a client certificate failure occurs, use the **no** form of the command:

```
host1/Admin(config-parammap-ssl)# no authentication-failure redirect
unknown-issuer
```

The following configuration examples instruct the ACE to perform an HTTP redirect to either a server farm or directly to a URL depending on the type of client certificate failure:

IPv6 Example

```
crypto authgroup AUTH-GROUP1

access-list EVERYONE line 8 extended permit ip anyv6 anyv6

rserver redirect EXPIRED
  webhost-redirection https://%h/support/expiredclientcert.html 302
  inservice
rserver redirect INVALID
  webhost-redirection https://%h/support/invalidclientcert.html 302
  inservice
rserver host SERVER1
  ip address 2001:DB8:1::10
  inservice
rserver host SERVER2
  ip address 2001:DB8:1::20
  inservice

serverfarm redirect EXPIRED-CERT
  rserver EXPIRED
    inservice
serverfarm redirect INVALID-CERT
  rserver INVALID
    inservice
serverfarm host WEB
```

```
  rserver SERVER1 80
    inservice
  rserver SERVER2 80
    inservice

parameter-map type ssl SSLPARAM
  authentication-failure redirect cert-not-yet-valid serverfarm
INVALID-CERT
  authentication-failure redirect cert-expired serverfarm EXPIRED-CERT
  authentication-failure redirect unknown-issuer url
https://www.example.com/NewCertRequest.html 302

ssl-proxy service SSLTERM-CLIENTAUTH
  ssl advanced-options SSLPARAM

class-map match-all CMAP-HTTPS
  2 match virtual-address 2001:DB:2::100 tcp eq https

policy-map type management first-match MGMTPOLICY
  class class-default
    permit

policy-map type loadbalance first-match SLB
  class class-default
    serverfarm WEB

policy-map multi-match VIPS
  class CMAP-HTTPS
    loadbalance vip inservice
    loadbalance policy SLB
    loadbalance vip icmp-reply

interface vlan 20
  ip address 2001:DB:2::1/64
  access-group input EVERYONE
  service-policy input VIPS
  no shutdown
interface vlan 40
  ip address 2001:DB:1::1/64
  service-policy input MGMTPOLICY
  no shutdown
```

IPv4 Example

```
crypto authgroup AUTH-GROUP1

access-list EVERYONE line 8 extended permit ip any any
```

```
rserver redirect EXPIRED
  webhost-redirection https://%h/support/expiredclientcert.html 302
  inservice
rserver redirect INVALID
  webhost-redirection https://%h/support/invalidclientcert.html 302
  inservice
rserver host SERVER1
  ip address 192.168.1.10
  inservice
rserver host SERVER2
  ip address 192.168.1.20
  inservice

serverfarm redirect EXPIRED-CERT
  rserver EXPIRED
    inservice
serverfarm redirect INVALID-CERT
  rserver INVALID
    inservice
serverfarm host WEB
  rserver SERVER1 80
    inservice
  rserver SERVER2 80
    inservice

parameter-map type ssl SSLPARAM
  authentication-failure redirect cert-not-yet-valid serverfarm
INVALID-CERT
  authentication-failure redirect cert-expired serverfarm EXPIRED-CERT
  authentication-failure redirect unknown-issuer url
https://www.example.com/NewCertRequest.html 302

ssl-proxy service SSLTERM-CLIENTAUTH
  ssl advanced-options SSLPARAM

class-map match-all CMAP-HTTPS
  2 match virtual-address 172.16.1.100 tcp eq https

policy-map type management first-match MGMTPOLICY
  class class-default
    permit

policy-map type loadbalance first-match SLB
  class class-default
    serverfarm WEB

policy-map multi-match VIPS
  class CMAP-HTTPS
```

```
    loadbalance vip inservice
    loadbalance policy SLB
    loadbalance vip icmp-reply

interface vlan 20
  ip address 172.16.1.1 255.255.255.0
  access-group input EVERYONE
  service-policy input VIPS
  no shutdown
interface vlan 40
  ip address 192.168.1.1 255.255.255.0
  service-policy input MGMTPOLICY
  no shutdown
```

Configuring the ACE to Ignore Authentication Failures Due to CDP Errors

By default, when you configure the **crl best-effort** command for client certificate revocation, if the ACE detects CRL distribution point (CDP) errors in the presented certificates or errors that occur during a CRL download, the ACE rejects the SSL connection.

The **cdp-errors ignore** command allows you to configure an SSL parameter map to ignore CDP or download errors when the **crl best-effort** command is configured. When you configure the **cdp-errors ignore** command, the ACE allows SSL connections if it detects CDP errors in the presented certificates or it could not download a valid certificate revocation list (CRL) from valid CDPs on the certificates. The syntax for this command in parameter map SSL configuration mode is as follows:

cdp-errors ignore

For example, to configure the ACE to ignore CDP errors, enter:

```
host1/Admin(config)# parameter-map type ssl PARAMMAP_SSL
host1/Admin(config-parammap-ssl)# cdp-errors ignore
```

To reset the default behavior where the ACE rejects an SSL connection when CDP errors occur, use the **no** form of the **cdp-errors ignore** command. For example, enter:

```
host1/Admin(config-parammap-ssl)# no cdp-errors ignore
```

To display the number of times that the ACE ignored CDP errors in the presented SSL certificate and allowed the SSL connection, use the **show crypto cdp-errors** command. This command displays the output of the Best Effort CDP Errors Ignored field.

Defining the Close-Protocol Behavior

You can configure how the ACE handles the sending of close-notify messages by using the **close-protocol** command in the parameter map SSL configuration mode. The syntax of this command is as follows:

close-protocol {**disabled** | **none**}

The keywords are as follows:

- **disabled**—Specifies that the ACE does not send a close-notify alert message to its peer when closing a session with no expectation of receiving one back from the peer.
- **none**—Specifies that the ACE sends a close-notify alert message to its peer when closing a session but has no expectation of receiving one back from the peer.

For example, to set **close-protocol** to disabled, enter:

```
host1/Admin(config)# parameter-map type ssl SSL_PARAMMAP_SSL
host1/Admin(config-parammap-ssl)# close-protocol disabled
```

To configure the **close-protocol** command with the default setting of none, use the **no** form of the command:

```
host1/Admin(config-parammap-ssl)# no close-protocol
```

Disabling Purpose Checking on the Certificates

By default, during client authentication of a chain of certificates, the ACE performs a purpose check on the basicContraint field for the following:

- The client certificate has a CA FALSE setting.
- The intermediate certificates have the CA TRUE setting.

If the field does not have these settings, the certificate fails authentication.

If you decide that it is unnecessary for the ACE to perform purpose checking during the authentication of the certificates, you can disable it by using the **purpose-check disabled** command in the parameter map SSL configuration mode.

The syntax of this command is as follows:

purpose-check disabled

For example, to disable purpose checking, enter:

```
host1/Admin(config)# parameter-map type ssl SSL_PARAMMAP_SSL
host1/Admin(config-parammap-ssl)# purpose-check disabled
```

To reenable the default setting of performing a purpose checking, use the **no** form of the command:

```
host1/Admin(config-parammap-ssl)# no purpose-check disabled
```

Enabling SSL Session Rehandshake

By default, SSL session rehandshake is disabled. To enable the SSL session rehandshake function, use the **rehandshake enabled** command in the parameter map SSL configuration mode.

The syntax of this command is:

rehandshake enabled

Note To enable SSL rehandshake for all VIPs in a context, see the "Enabling SSL Rehandshake for All VIPs in a Context" section.

For example, to enable the SSL rehandshake function, enter:

```
host1/Admin(config)# parameter-map type ssl PARAMMAP_SSL
host1/Admin(config-parammap-ssl)# rehandshake enabled
```

To disable the rehandshake function, enter:

```
host1/Admin(config-parammap-ssl)# no rehandshake enabled
```

To display the status of the **rehandshake enabled** command, use the **show parameter-map** command.

Defining the SSL and TLS Versions

You can specify the version of the security protocol that the ACE supports during the SSL handshake with its peer by using the **version** command in parameter map SSL configuration mode.

The syntax of this command is as follows:

version {**all** | **ssl3** | **tls1**}

The keywords are as follows:

- **all**—(Default) The ACE supports both SSL Version 3.0 and Transport Layer Security (TLS) Version 1.0.
- **ssl3**—The ACE supports only SSL Version 3.0.
- **tls1**—The ACE supports only TLS Version 1.0.

For example, to specify SSL Version 3.0 for the parameter map, enter:

```
host1/Admin(config)# parameter-map type ssl PARAMMAP_SSL
host1/Admin(config-parammap-ssl)# version ssl3
```

To remove a security protocol version from the SSL proxy parameter map, enter:

```
host1/Admin(config-parammap-ssl)# no version tls1
```

Configuring the SSL Queue Delay

The ACE queues packet data from the server before encrypting it for transmission to the client. The ACE empties the data from the queue for encryption when one of the following events occurs:

- The queue reaches 4096 bytes.
- The server sends a TCP-FIN segment.
- The queue delay time on the ACE has passed even though the queue had not reached 4096 bytes.

The queue delay time is the amount of time that the ACE waits before emptying the queued data for encryption. By default, the queue delay timer is disabled. You can set the delay time by using the **queue-delay timeout** command in parameter map SSL configuration mode. The syntax of this command is as follows:

queue-delay timeout *milliseconds*

The *milliseconds* argument is the time in milliseconds before the data is emptied from the queue. Enter an integer from 0 to 10000. A value of 0 disables the delay timer, causing the ACE to encrypt data from the server as it arrives and then send the encrypted data to the client.

Note The queue delay applies only to encrypted data that the ACE sends to the client.

For example, to set the queue delay time to 500 milliseconds, enter:

```
host1/Admin(config-parammap-ssl)# queue-delay timeout 500
```

To disable the queue delay timer, enter:

```
host1/Admin(config-parammap-ssl)# no queue-delay timeout
```

Configuring the SSL Session Cache Timeout

An SSL session ID is created every time that the client and the ACE perform a full SSL key exchange and establish a new master secret key. To quicken the SSL negotiation process between the client and the ACE, the SSL session ID reuse feature allows the ACE to reuse the secret key information in the session cache. On subsequent connections with the client, the ACE reuses the key stored in the cache from the last negotiated session. The ACE can store a maximum of 250,000 (ACE module) or 100,000 (ACE appliance) SSL session IDs in the session cache.

By default, SSL session ID reuse is disabled on the ACE. You can enable session ID reuse by setting a session cache timeout value for the total amount of time that the SSL session ID remains valid before the ACE requires a full SSL handshake to establish a new session.

You can set the session cache timeout by using the **session-cache timeout** command in parameter map SSL configuration mode. The syntax of this command is as follows:

session-cache timeout *seconds*

The *seconds* argument is the time in seconds that the ACE reuses the key stored in the cache before removing the session IDs. Enter an integer from 0 to 72000 (20 hours). By default, session ID reuse is disabled. A value of 0 causes the ACE to remove the session IDs from the cache when the cache is full and to implement the least-recently used (LRU) timeout policy.

For example, to set the session cache timeout to 600 seconds, enter:

```
host1/Admin(config-parammap-ssl)# session-cache timeout 600
```

To disable the timer and allow the SSL full handshake to occur for each new connection with the ACE, enter:

```
host1/Admin(config-parammap-ssl)# no session-cache timeout
```

To clear the session cache information for the context, use the **clear crypto session-cache** command. The syntax of this command is as follows:

clear crypto session-cache [**all**]

The **all** optional keyword clears all session cache information for all contexts. This option is available in the Admin context only.

Rejecting Expired CRL Client Certificates

When you configure Certificate Revocation Lists (CRLs) on the ACE for client authentication, as described in the "Using CRLs During Client Authentication" section, the CRLs contain an update field that specifies the date when a new version would be available. By default, the ACE does not use CRLs that contain an update field with an expired date and, thus, does not reject incoming client certificates using the CRL.

To configure the ACE to reject a client certificate when the CRL in use has expired, use the **expired-crl reject** command in parameter map SSL configuration mode. The syntax of this command is as follows:

expired-crl reject

For example, enter:

```
host1/Admin(config-parammap-ssl)# expired-crl reject
```

To reset the default behavior of the ACE accepting a client certificate after the CRL in use has expired, enter:

```
host1/Admin(config-parammap-ssl)# no expired-crl reject
```

Enabling SSL Rehandshake for All VIPs in a Context

By default, SSL rehandshake is disabled for all VIPs in a context. To enable SSL rehandshake for all VIPs in a context, use the **crypto rehandshake enabled** command in configuration mode. The syntax of this command is as follows:

crypto rehandshake enabled

For information about enabling SSL rehandshake in a parameter map for an SSL proxy service, see the "Enabling SSL Session Rehandshake" section.

Note The **crypto rehandshake enabled** configuration mode command overrides the **rehandshake enable** parameter map command that you can configure individually in an SSL proxy service.

For example, to enable SSL rehandshake for all VIPs in a context, enter the following command:

```
host1/Admin(config)# crypto rehandshake enabled
```

To return the ACE behavior to the default of rehandshake being disabled for all VIPs in a context, enter the following command:

```
host1/Admin(config)# no crypto rehandshake enabled
```

Creating and Defining an SSL Proxy Service

The SSL proxy service defines the SSL parameter map, key pair, certificate, and chain group that the ACE uses during the SSL handshake. For SSL termination, you configure the ACE with an SSL proxy *server* service because the ACE acts as an SSL server.

You can create an SSL proxy server service by using the **ssl-proxy service** command in configuration mode.

The syntax of this command is as follows:

ssl-proxy service *pservice_name*

The *pservice_name* argument is the name of the SSL proxy server service. Enter an unquoted alphanumeric string with a maximum of 64 characters.

For example, to create the SSL proxy server service PSERVICE_SERVER, enter:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
```

After you create an SSL proxy server service, the CLI enters SSL proxy configuration mode.

```
host1/Admin(config-ssl-proxy)#
```

To delete an existing SSL proxy server service, enter:

```
host1/Admin(config)# no ssl-proxy PSERVICE_SERVER
```

This section contains the following topics:

- Associating an SSL Parameter Map with the SSL Proxy Server Service
- Specifying the Key Pair
- Specifying the Certificate
- Specifying the Certificate Chain Group
- Enabling Client Authentication
- Using CRLs During Client Authentication
- Configuring the Download Location for CRLs
- Configuring Global CRL Parameters

Associating an SSL Parameter Map with the SSL Proxy Server Service

You can associate an SSL parameter map with the SSL proxy server service by using the **ssl advanced-options** command in SSL proxy configuration mode.

The syntax of this command is as follows:

ssl advanced-options *parammap_name*

The *parammap_name* argument is the name of an existing SSL parameter map (see the "Creating and Defining an SSL Parameter Map" section). Enter an unquoted alphanumeric string with a maximum of 64 characters.

For example, to associate the parameter map PARAMMAP_SSL with the SSL proxy service, enter:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# ssl advanced-options PARAMMAP_SSL
```

To remove the association of an SSL parameter map with the SSL proxy service, enter:

```
host1/Admin(config-ssl-proxy)# no ssl advanced-options PARAMMAP_SSL
```

Specifying the Key Pair

You can specify the key pair that the ACE uses during the SSL handshake for data encryption by using the **key** command in SSL proxy configuration mode.

Note The public key in the key pair file that you select must match the public key embedded in the certificate that you select (see the "Specifying the Certificate" section). For information on verifying a public key match, see the "Verifying a Certificate Against a Key Pair" section in Chapter 2, Managing Certificates and Keys.

The syntax of this command is as follows:

key {*key_filename* | **cisco-sample-key**}

The argument and keyword are as follows:

- *key_filename*—Name of an existing key pair file loaded on the ACE. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **cisco-sample-key**—Specifies the sample RSA 1024-bit key pair named cisco-sample-key that is preinstalled on the ACE. This file is available for use in any context with the filename remaining the same in each context. For more information on this key pair, see the "Using the ACE Sample Certificate and Key Pair" section.

For example, to specify the private key in the key pair file MYKEY.PEM, enter:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# key MYKEY.PEM
```

To delete a private key from the SSL proxy service, enter:

```
host1/Admin(config-ssl-proxy)# no key MYKEY.PEM
```

Specifying the Certificate

You can specify the certificate that the ACE uses during the SSL handshake process to prove its identity by using the **cert** command in SSL proxy configuration mode.

Note The public key embedded in the certificate that you select must match the public key in the key pair file that you select (see the "Specifying the Key Pair" section). For information on verifying a public key match, see the "Verifying a Certificate Against a Key Pair" section in Chapter 2, Managing Certificates and Keys.

The syntax of this command is as follows:

cert {*cert_filename* | **cisco-sample-cert**}

The argument and keyword are as follows:

- *cert_filename*—Name of an existing certificate file loaded on the ACE. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **cisco-sample-cert**—Specifies the self-signed certificate named cisco-sample-cert that is preinstalled on the ACE. This file is available for use in any context with the filename remaining the same in each context. For more information on this certificate, see the "Using the ACE Sample Certificate and Key Pair" section.

For example, to specify the certificate in the certificate file MYCERT.PEM, enter:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# cert MYCERT.PEM
```

To delete a certificate file from the SSL proxy service, enter:

```
host1/Admin(config-ssl-proxy)# no cert MYCERT.PEM
```

Specifying the Certificate Chain Group

You can specify the certificate chain that the ACE sends to its peer during the SSL handshake by using the **chaingroup** command in SSL proxy configuration mode. The ACE includes the certificate chain with the certificate that you specified for the SSL proxy service (see the "Specifying the Certificate" section).

The syntax of this command is as follows:

chaingroup *group_name*

The *group_name* argument is the name of an existing certificate chain group (see the "Creating a Chain Group" section in Chapter 2, Managing Certificates and Keys). The maximum size of a chain group is 11 KB. Enter an unquoted alphanumeric string with a maximum of 64 characters.

Note When you make a change to a chain-group certificate, the change takes effect only after you respecify the associated chain group in the SSL proxy service using the **chaingroup** command.

For example, to specify the certificate chain group MYCHAINGROUP, enter:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# chaingroup MYCHAINGROUP
```

To delete a certificate chain group from the SSL proxy service, enter:

```
host1/Admin(config-ssl-proxy)# no chaingroup MYCHAINGROUP
```

Enabling Client Authentication

During the flow of a normal SSL handshake, the server sends its certificate to the client. The client verifies the identity of the server through the certificate. However, the client does not send any identification of its own to the server. When you enable the client authentication feature on the ACE, the ACE requires that the client sends a certificate to the server. The server then verifies the following information on the certificate:

- A recognized CA issued the certificate.
- The valid period of the certificate is still in effect.

- The certificate signature is valid.
- The CA has not revoked the certificate.

You can specify the certificate authentication group that the ACE uses during the SSL handshake and enable client authentication on this SSL proxy service by using the **authgroup** command in SSL proxy configuration mode. The ACE includes the certificates configured in the group with the certificate that you specified for the SSL proxy service (see the "Specifying the Certificate" section).

The syntax of this command is as follows:

authgroup *group_name*

The *group_name* argument is the name of an existing certificate authentication group (see the "Configuring a Group of Certificates for Authentication" section in Chapter 2, Managing Certificates and Keys). Enter an unquoted alphanumeric string with a maximum of 64 characters.

Note When you enable client authentication on the ACE, a significant performance decrease may occur on the ACE causing access failures. Additional latency may occur when you configure CRL retrieval (see the "Using CRLs During Client Authentication" section) or when VIP traffic exceeds 100-200 TPS and access failures occur due to Certificate Revoked errors.

Note When you make a change to an authgroup, the change takes effect only after you respecify the associated authgroup in the SSL proxy service using the **authgroup** command.

For example, to specify the certificate authentication group AUTH-CERT1, enter:

```
host1/Admin(config-ssl-proxy)# authgroup AUTH-CERT1
```

To delete a certificate authentication group from the SSL proxy service, enter:

```
host1/Admin(config-ssl-proxy)# no authgroup AUTH-CERT1
```

Using CRLs During Client Authentication

By default, the ACE does not use certificate revocation lists (CRLs) during client authentication. The ACE supports CRL downloads through HTTP or LDAP. You can configure the SSL proxy service to use a CRL in one of the following ways:

- The ACE can scan each client certificate for the service to determine if it contains a CRL Distribution Point (CDP) pointing to a CRL in the certificate extension and then retrieve the CRL from that location if the CDP is valid. If the CDP has an http:// or ldap:// based URL, it uses the URL to download the CRL to the ACE.
- You can manually configure the download location for the CRL from which the ACE retrieves it (see the "Configuring the Download Location for CRLs" section).

Note By default, the ACE does not reject client certificates when the CRL in use has passed its update date. To configure the ACE to reject certificates when the CRL is expired, use the **expired-crl reject** command. For more information, see the "Rejecting Expired CRL Client Certificates" section.

When attempting to download a CRL when best-effort CRLs are configured, the following apply:

- The ACE considers only the first four CDPs in the certificate or configured on the ACE. For the CDPs obtained from the certificate, the ACE only considers valid and complete CDPs for the downloading of the CRLs. If a CDP leads to the successful downloading of the CRL, ACE does not consider the subsequent CDPs for CRL downloads.
- If none of the first four CDPs are valid to proceed with the downloading of the CRL, the ACE considers the certificate as revoked unless you configured the **authentication-failure ignore** command in parameter map SSL configuration mode.
- If the ACE fails to download a CRL after trying four valid CDPs, the ACE aborts its initiated SSL connection unless you configured the **authentication-failure ignore** command in parameter map SSL configuration mode.

- If the ACE detects CDP errors in the presented certificates or errors that occur during a CRL download, the ACE rejects the SSL connection unless you configured the **cdp-errors ignore** command in parameter map SSL configuration mode
- The ACE skips malformed CDPs and processes subsequent CDPs. To display CDP error statistics including the number of malformed CDPs, use the **show crypto cdp-errors** command.

For detailed CRL download statistics, see the "Displaying CRL Information" section in Chapter 6, "Displaying SSL Information and Statistics."

You can determine which CRL information to use for client authentication by using the **crl** command in SSL proxy configuration mode. The syntax of this command is as follows:

crl {*crl_name* | **best-effort**}

The argument and keyword are as follows:

- *crl_name*—Name that you assigned to the CRL when you downloaded it with the configuration mode **crypto crl** command. See the "Configuring the Download Location for CRLs" section.
- **best-effort**—Specifies that the ACE scans each client certificate to determine if it contains a CDP pointing to a CRL in the certificate extension and then retrieves the CRLs from that location, if the CDP is valid.

For example, to enable the CRL1 CRL for client authentication on an SSL proxy service, enter:

```
host1/Admin(config-ssl-proxy)# crl CRL1
```

To scan the client certificate for CRL information, enter:

```
host1/Admin(config-ssl-proxy)# crl best-effort
```

To disable the use of a downloaded CRL during client authentication, enter:

```
host1/Admin(config-ssl-proxy)# no crl CRL1
```

To disable the use of client certificates for CRL information during client authentication, enter:

```
host1/Admin(config-ssl-proxy)# no crl best-effort
```

Configuring the Download Location for CRLs

You can configure the location that the ACE uses to download the CRL to the SSL proxy service for client authentication. If the service is not configured on a policy map or the policy map is not active, the ACE does not download the CRL. The ACE downloads the CRL under the following conditions:

- When you first configure the CRL and apply it to an active Layer 4 policy map as an action (see the "Associating an SSL Proxy Server Service with the Policy Map" section).
- When you reload the ACE.
- When the NextUpdate arrives, as provided within the CRL itself, the ACE reads this information and updates the CRL based on it. The ACE downloads the updated CRL upon the next client authentication request.

You can configure a maximum of eight CRLs per context. After you configure the CRL, assign it to an SSL proxy service for client authentication (see the "Using CRLs During Client Authentication" section).

The ACE translates the hostnames within the CRLs to IP addresses using a Domain Name System (DNS) client that you configure. For details about configuring a DNS client, see the "Configuring a DNS Client" section.

To configure a downloaded CRL, use the **crypto crl** command in configuration mode. The syntax of this command is as follows:

crypto crl *crl_name url*

The arguments are as follows:

- *crl_name*—Name that you want to assign to the CRL. Enter an unquoted alphanumeric string with a maximum of 64 characters.
- *url*—URL where the ACE retrieves the CRL. Enter the URL full path including the CRL filename in an unquoted alphanumeric string with a maximum of 255 characters. Both HTTP and LDAP URLs are supported. Start the URL with the http:// prefix or the ldap:// prefix.

  The ldap:/// prefix is not considered a valid LDAP CRL link in the CDP portion of the server certificate. Valid formats for LDAP URLs are as follows:

  - ldap://10.10.10.1:389/dc=cisco,dc=com?o=bu?certificateRevocationList
  - ldap://10.10.10.1/dc=cisco,dc=com?o=bu?certificateRevocationList

- ldap://ldapsrv.cisco.com/dc=cisco,dc=com?o=bu?certificateRevocationList
- ldap://ldapsrv.cisco.com:389/dc=cisco,dc=com?o=bu?certificateRevocationList

To use a question mark (?) character as part of the URL, press Ctrl-v before entering it. Otherwise the ACE interprets the question mark as a help command.

Note The hostname in ldap:// links are resolved using DNS configurations. LDAP uses TCP port 389. If the LDAP server that publishes the CRL listens on a non-standard LDAP port, then a non-standard LDAP port needs to be configured in the CDP.

For example, to configure a CRL that you want to name CRL1 from http://crl.verisign.com/class1.crl, enter:

```
host1/Admin(config)# crypto crl CRL1
http://crl.verisign.com/class1.crl
```

To remove the CRL, enter:

```
host1/Admin(config)# no crypto crl CRL1
```

For example, Figure 3-4 illustrates a sample configuration for CRL downloading through LDAP in client authentication.

Figure 3-4 CRL Download through the LDAP Protocol

The following example is the configuration of the authentication group with the root certificate that signed the client certificate:

```
crypto authgroup root_ca_pool
    cert root-cert-2.cer
```

The following example provides the configuration for the ldap:// based CDP URL:

```
crypto crl win2003crl1
ldap://windows2003-srv.win2003.cisco.com/CN=root-ca(2),CN=windows2003-
srv,CN=CDP,CN=Public%20Key%20Services,CN=Services,CN=Configuration,DC=
win2003,DC=cisco,DC=com?certificateRevocationList?base?objectClass=cRL
DistributionPoint

access-list capture-acl line 8 extended permit tcp any any
access-list permit-http line 8 extended permit tcp any any eq https
```

The following example provides the DNS configuration for the ACE to successfully resolve the hostname in the ldap:// URL during the CRL download:

```
ip domain-lookup
ip domain-name win2003.cisco.com
ip name-server 10.1.1.147

rserver host SERVER1
    ip address 10.1.1.122
    inservice

ssl-proxy service SSL_PSERVICE_SERVER
    key MYKEY.PEM
    cert MYSCERT.PEM
    authgroup root_ca_pool
    crl win2003crl1

serverfarm host SFARM1
    rserver SERVER1 80
        inservice

class-map match-any L4_SSL-TERM_CLASS
    3 match virtual-address 192.168.1.100 tcp eq https
class-map type http loadbalance match-all URLCLASS1
    2 match http url .*

policy-map type loadbalance first-match L7_SSL-TERM_POLICY
    class URLCLASS1
        serverfarm SFARM1

policy-map multi-match L4_SSL-VIP_POLICY
    class L4_SSL-TERM_CLASS
        loadbalance vip inservice
        loadbalance policy L7_SSL-TERM_POLICY
        loadbalance vip icmp-reply
        ssl-proxy server SSL_PSERVICE_SERVER

interface vlan 50
```

```
    ip address 10.1.1.138 255.255.0.0
    no shutdown

interface vlan 200
    ip address 192.168.1.254 255.255.0.0
    access-group input permit-http
    service-policy input L4_SSL-VIP_POLICY
    no shutdown
```

Configuring Global CRL Parameters

You can configure signature verification on a Certificate Revocation List (CRL) to determine that it is from a trusted certificate authority or you can configure a timeoute for CRL downloads to specify the maximum wait time for the ACE to retrieve the CRL data from a server by using the **crypto crlparams** command in configuration command mode. The syntax of this command is as follows:

crypto crlparams *crl_name* **cacert** *ca_cert_filename* | **timeout** *number*

The keywords and arguments are as follows:

- *crl_name*—Name of an existing CRL.
- **cacert** *ca_cert_filename*—Name of the CA certificate file used for signature verification.
- **timeout** *number*—Specifies the time in seconds that the ACE waits for the CRL data before closing the connection with the server. For static CRLs, enter an integer from 2 to 300. For best-effort CRLs, the timeout is 60 seconds and not user-configurable. If the ACE does not receive the entire CRL data within the timeout limit, the ACE closes the socket connection with the server. For static CRLs, you can abort the CRL data download by removing the static CRL from the configuration.

For example, to configure signature verification on a CRL, enter:

```
host1/Admin(config)# crypto crlparams CRL1 cacert MYCERT.PEM
```

To remove signature verification from a CRL, enter:

```
host1/Admin(config)# no crypto crlparams CRL1
```

For example, to configure a 200-second CRL download timeout for CRL1, enter the following command:

host1/Admin(config)# **crypto crl-params CRL1 timeout 200**

When the CRL data download timeout expires and the download is aborted, the ACE generates a syslog to log the event as follows:

```
%ACE-6-253008: CRL crl_name could not be retrieved, reason: crl data
dnld timeout error
```

The *crl_name* variable indicates the name of an existing CRL whose download was aborted because the CRL download timeout expired.

Configuring the Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) is a method that is used between a client and a server to determine the revocation status of a certificate. An OCSP server (OCSP responder) maintains or obtains certificate information from different certificate authorities (CAs) and provides this information to the client upon request. The advantages of OCSP are that it can provide the latest information about a certificate and it eliminates the need to download and store a certificate revocation list (CRL), which can require a large amount of memory.

The ACE acts as an OCSP client and sends requests to an OCSP server over HTTP. If required, requests may be signed by the public key of a designated CA certificate. The requests contain the following major pieces of information:

- Certificate identifier
- Hash of the certificate issuer's distinguished name
- Hash of the certificate issuer's public key

The server responds with the certificate revocation information that indicates whether the certificate is revoked, not revoked, or the status could not be determined.

The ACE uses OCSP as an alternative to using CRLs. This OCSP implementation is compliant with RFC 2560, but does not include all the optional features in the RFC. The ACE supports the following RFC 2560 optional features:

- HTTP is the only OCSP transport mechanism

- A signer certificate may or may not contain the id-kp-OCSPsigning extendedKeyUsage extension (section 4.2.2.2 of RFC) because requiring that extension in the certificates could make the certificates' selection eligible for signing the requests very restrictive.
- The optional request signer certificate is not included in the requests that the ACE sends to the OCSP server.
- The ACE does not perform either an authentication or a revocation check of the authorized OCSP responder's certificate.
- Except for optionally supporting Nonce, no other extension defined in section 4.4 of RFC is supported.

The handling and treatment of multiple OCSP server access locations in the certificate are the same as that for multiple certificate CRL distribution points (CDPs) within a certificate. Up to four authority information access (AIA) fields are extracted from the certificates.

You can configure both CRLs and OCSP servers in the same SSL proxy. In this case, you can configure the order in which the ACE performs the revocation check. By default, the ACE tries the OCSP servers first and then the CRLs to obtain revocation status.

Note The coexistence of CRLs and OCSP server information and traversal through them may lead to extended handshake completion time and the overall performance of the ACE may degrade.

This section contains the following subsections:

Guidelines and Restrictions

Configuring OCSP has the following guidelines and restrictions:

- You can configure a maximum of 64 OCSP servers in the ACE.

- You can configure a maximum of 10 OCSP servers in an SSL proxy service.
- The ACE can handle a maximum of 64 OCSP server connections with both static and best effort OCSP servers combined.
- If you configure best-effort OCSP servers and best-effort CRLs in the same proxy list, the ACE extracts a maximum of four AIAs and four CDPs to conserve resources.
- Client authentication may be delayed when you configure OCSP servers and CRLs in the same SSL proxy service.
- The ACE does not perform authentication and revocation checks on response signer certificates.

Configuring an OCSP Server

You can configure an OCSP server that the ACE uses for revocation checks. The maximum number of OCSP servers that you can configure in an ACE is 64. You can configure all 64 OCSP servers in one context or spread them out over multiple contexts. The maximum number of OCSP server connections is also 64.

To configure an OCSP server that you will later apply to an SSL proxy service, use the **crypto ocspserver** command in configuration mode. The syntax of this command is as follows:

crypto ocspserver *ocsp_server_name url* [**conninactivitytout** *timeout*] [**nonce enable | disable**] [**reqsigncert** *signer_cert_filename* {**reqsignkey** *signer_key_filename*}] [**respsigncert** *response_signer_cert*]

The keywords, options, and arguments are as follows:

- *ocsp_server_name*—Identifier of the OCSP server. You use this name to apply the OCSP server to an SSL proxy service. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *url*—HTTP URL in the form: http://ocsphost.com:port_id/. The port ID is optional. If you do not specify a port, the default value of 2560 is used. You can specify either an IPv4- or an IPv6-based URL. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters.
- **conninactivitytout** *timeout*—(Optional) TCP connection inactivity timeout. in seconds. Enter an integer from 2 to 3600. The default is 300 seconds.

- **nonce enable | disable**—(Optional) Enables or disables the use of a nonce. By default, **nonce** is disabled. A nonce is a unique string that is used to bind OCSP requests and responses. When a nonce is enabled, the ACE includes a unique string in the requests that is sends to the OCSP server. The server must include the string in its responses to the ACE to verify the response.
- **reqsigncert** *signer_cert_filename*—(Optional) Signer's certificate filename to sign outgoing requests to the OCSP server. By default, the request is not signed.
- **reqsignkey** *signer_key_filename*—(Optional) Signer's private key filename to sign outgoing requests to the OCSP server. By default, the request is not signed. If you enter the **reqsigncert** option, you must enter the **reqsignkey** option.
- **respsigncert** *response_signer_cert*—(Optional) Certificate to verify the signature of the OCSP server responses. By default, the signature in the response from the OCSP server are not verified.

For example, to configure an OCSP server that the ACE uses to check the revocation status of SSL certificates, enter the following command:

```
host1/Admin(config)# crypto ocspserver OCSP_SSERVER1
http://10.10.10.10/ nonce enable conninactivitytout 60
```

To remove an OCSP server from the configuration, enter the following command:

```
host1/Admin(config)# no crypto ocspserver OCSP_SSERVER1
```

Applying an OCSP Server to an SSL Proxy Service

You can apply a maximum of 10 OCSP servers to an SSL proxy service. To apply an OCSP server to an SSL proxy service, use the **ocspserver** command in SSL proxy configuration mode. The syntax of this command is as follows:

ocspserver *ocsp_server_name* | **best-effort**

The argument and option are as follows:

- *ocsp_server_name*—Identifier of an OCSP server that you want to apply to this SSL proxy service. Enter the name of an existing OCSP server as a text string with no spaces and a maximum of 64 alphanumeric characters.

- **best-effort**—Specifies that the ACE attempts to obtain certificate revocation information from an OCSP server on a best-effort basis. When you configure this keyword, the ACE extracts the OCSP server information (up to four OCSP server information elements) from the client certificate. This keyword forces the ACE to look for the AuthorityInfoAccess (AIA) extension in the incoming client or server certificates. The format of this extension is as follows:

  ```
  authorityInfoAccess = OCSP;URI:
  http://test1.ocsp.ve/,OCSP;URI:http://test2.ocsp.ve/
  ```

  If this extension is missing from a certificate when **best-effort** is configured, the certificate is considered to be revoked.

For example, to apply the OCSP_SERVER1 OCSP server to the PSERVICE_SERVER SSL proxy service, enter the following commands:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# ocspserver OCSP_SERVER1
```

To apply a best-effort OCSP server to an SSL proxy service, enter the following commands:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# ocspserver best-effort
```

To remove an OCSP server from an SSL proxy service, enter the following command;

```
host1/Admin(config-ssl-proxy)# no ocspserver OCSP_SERVER1
```

Configuring the Priority of Revocation Checking

When you configure both OCSP and CRLs in the same SSL proxy service, you can control the order in which the ACE uses these two resources to check the revocation status of SSL certificates. To configure the order of revocation checking, use the **revcheckprio** command in SSL proxy configuration mode. If either OCSP or CRLs, but not both methods, are applied to an SSL proxy service, this command is not configurable. The syntax of this command is as follows:

revcheckprio crl-ocsp | ocsp-crl

The *keywords* are as follows:

- **crl-ocsp**—Instructs the ACE to use a CRLs first and then OCSP to determine the revocation status of a client SSL certificate
- **ocsp-crl**—(Default) Instructs the ACE to use OSCP first and then CRLs to determine the revocation status of a client SSL certificate.

When this command is not configured, the ACE uses OSCP first and then CRLs to determine the revocation status of an SSL certificate. If both methods fail to determine the certificate's status, the certificate is considered to be revoked.

Note The default revocation check priority order (**revcheckprio ocsp-crl**) is not displayed in the output of the **show running-config** command even if that priority order is configured.

For example, to configure the ACE to check revocation status with CRLs first and then OCSP, enter the following commands:

```
host1/Admin(config)# ssl-proxy service PSERVICE_SERVER
host1/Admin(config-ssl-proxy)# revcheckprio crl-ocsp
```

To reset the ACE behavior to the default of checking the OCSP server first and then the CRLs for certificate revocation, enter the following command:

```
host1/Admin(config-ssl-proxy)# no revcheckprio crl-ocsp
```

Configuring a DNS Client

With the client authentication feature, you can configure a Domain Name System (DNS) client on the ACE to communicate with a DNS server to provide hostname-to-IP-address translation for hostnames in CRLs. For details about client authentication, see the "Using CRLs During Client Authentication" section.

Before you configure a DNS client on the ACE, ensure that one or more DNS name servers are properly configured and reachable. Otherwise, translation requests (domain lookups) from the DNS client will be discarded. You can configure a maximum of three name servers. The ACE attempts to resolve the hostnames with the configured name servers in order until the translation succeeds. If the translation fails, the ACE reports an error.

For unqualified hostnames (hostnames that do not contain a domain name), you can configure a default domain name or a list of domain names that the ACE can use to perform the following tasks:

- Complete the hostname
- Attempt a host-name-to-IP-address resolution with a DNS server

To display the DNS client configuration, use the **show running-config** command.

This section contains the following topics:

Enabling Domain Lookups

To enable the ACE to perform a domain lookup (host-to-address translation) with a DNS server, use the **ip domain-lookup** command in configuration mode. By default, this command is disabled. The syntax of this command is as follows:

ip domain-lookup

For example, to enable domain lookups, enter:

```
host1/Admin(config)# ip domain-lookup
```

To return the state of domain lookups to the default value of disabled, enter:

```
host1/Admin(config)# no ip domain-lookup
```

Configuring a Default Domain Name

The DNS client feature allows you to configure a default domain name that the ACE uses to complete unqualified hostnames. An unqualified hostname is one that does not contain a domain name (any name without a dot). When domain lookups are enabled and a default domain name is configured, the ACE appends a dot (.) and the configured default domain name to the unqualified hostname and attempts a domain lookup.

To configure a default domain name, use the **ip domain-name** command in configuration mode. The syntax of this command is as follows:

ip domain-name *name*

The *name* argument is an unquoted text string with no spaces and a maximum of 85 alphanumeric characters.

For example, to specify a default domain name of cisco.com, enter:

```
host1/Admin(config)# ip domain-name cisco.com
```

In the above example, the ACE appends .cisco.com to any unqualified hostname in a CRL before the ACE attempts to resolve the hostname to an IP address using a DNS name server.

To remove the default domain from the configuration, enter:

```
host1/Admin(config)# no ip domain-name cisco.com
```

Configuring a Domain Name Search List

Instead of configuring a single default domain name, you can configure a domain name search list that the ACE uses to complete unqualified hostnames. The domain name list can contain a maximum of three domain names. If you configure both a domain name list and a default domain name, the ACE uses only the domain name list and not the single default name. After you have enabled domain name lookups and configured a domain name list, the ACE uses each domain name in turn until it can resolve a single domain name into an IP address.

To configure a domain name search list, use the **ip domain-list** command. The syntax of this command is as follows:

ip domain-list *name*

The *name* argument is an unquoted text string with no spaces and a maximum of 85 alphanumeric characters.

For example, to configure a domain name list, enter:

```
host1/Admin(config)# ip domain-list cisco.com
host1/Admin(config)# ip domain-list foo.com
host1/Admin(config)# ip domain-list xyz.com
```

To remove a domain name from the list, enter:

```
host1/Admin(config)# no ip domain-list xyz.com
```

Configuring a Domain Name Server

To translate a hostname to an IP address, you must configure one or more (maximum of three) existing DNS name servers on the ACE. Ping the IP address of each name server before you configure it to ensure that the server is reachable.

To configure a name server, use the **ip name-server** command in configuration mode. The syntax of this command is as follows:

ip name-server *ip_address*

The *ip_address* argument is the IP address of a name server in dotted decimal notation (for example, 192.168.12.15). You can enter up to three name server IP addresses in one command line.

For example, to configure three name servers for the DNS client feature, enter:

```
host1/Admin(config)# ip name-server 192.168.12.15 192.168.12.16
192.168.12.17
```

To remove a name server from the list, enter:

```
host1/Admin(config)# no ip name-server 192.168.12.15
```

Configuring SSL URL Rewrite and HTTP Header Insertion

When a client sends encrypted traffic to the ACE in an SSL termination configuration, the ACE terminates the SSL traffic and then sends clear text to the server, which is unaware of the encrypted traffic flowing between the client and the ACE. Using an action list associated with a Layer 7 HTTP load-balancing policy map, you can instruct the ACE to perform the following tasks:

- SSL URL Rewrite—The ACE changes the redirect URL from http:// to https:// in the Location response header from the server before sending the response to the client.

- SSL HTTP Header Insertion—The ACE provides the server with the following SSL session information by inserting HTTP headers into the HTTP requests that it receives over the connection:
  - Session Parameters—SSL session parameters that the ACE and client negotiate during the SSL handshake.
  - Server Certificate Fields—Information regarding the SSL server certificate that resides on the ACE.
  - Client Certificate Fields—Information regarding the SSL client certificate that the ACE retrieves from the client when you configure the ACE to perform client authentication.

The following sections describe how to configure the ACE for SSL URL rewrite and HTTP header insertion using an action list that provides the ACE with the necessary instructions.

This section contains the following topics:

Configuring the Action List

To configure SSL URL rewrite or HTTP header insertion, you must first create a new action list or use an existing action list of type modify.

Caution An action list that you configure for SSL HTTP header insertion must be associated with the class-default class map only; therefore, you cannot configure an existing action list for SSL HTTP header insertion if the action list is currently associated with a class map that is not the class-default class map.

An action list is a named group of related actions that you want the ACE to perform. For example, to create an action list, enter the following command in configuration mode:

```
host1/Admin(config)# action-list type modify http SSL_ACTLIST
host1/Admin(config-actlist-modify)#
```

The **action-list type modify http** command enters the action list modify configuration mode from which you define the parameters for the following features:

- SSL URL Rewrite (see the "Configuring SSL URL Rewrite" section)
- SSL Session Parameters Insertion (see the "Configuring HTTP Header Insertion of SSL Session Parameters" section)
- SSL Server Certificate Field Insertion (see the "Configuring HTTP Header Insertion of SSL Server Certificate Information" section)
- SSL Client Certificate Field Insertion (see the "Configuring HTTP Header Insertion of SSL Client Certificate Information" section)

For more information about action lists, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

Configuring SSL URL Rewrite

Because the server is unaware of the encrypted traffic flowing between the client and the ACE, the server may return to the client a URL in the Location header of HTTP redirect responses (301: Moved Permanently or 302: Found) in the form http://www.cisco.com instead of https://www.cisco.com. In this case, the client makes a request to the unencrypted insecure URL, even though the original request was for a secure URL. Because the client connection changes to HTTP, the requested data may not be available from the server using a clear text connection.

To solve this problem, the ACE provides SSLURL rewrite, which changes the redirect URL from http:// to https:// in the Location response header from the server before sending the response to the client. By using URL rewrite, you can avoid nonsecure HTTP redirects. All client connections to the web server will be SSL, ensuring the secure delivery of HTTPS content back to the client. The ACE uses regular expression matching to determine whether the URL needs rewriting.

If a Location response header matches the specified regular expression, the ACE rewrites the URL. In addition, the ACE provides commands to add or change the SSL and the clear port numbers.

You can define the SSL URL, SSL port, and clear port for rewrite by using the **ssl url rewrite** command in action list modify configuration mode. The syntax of this command is as follows:

ssl url rewrite location *expression* [**sslport** *number1*] [**clearport** *number2*]

The arguments, keywords, and options are as follows:

- **location** *expression*—Specifies the rewriting of the URL in the Location response header based on a URL regular expression match. If the URL in the Location header matches the URL regular expression string that you specify, the ACE rewrites the URL from http:// to https:// and rewrites the port number. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“).

  The location regex that you enter must be a pure URL (for example, www\.cisco\.com) with no port or path designations. To match a port, use the **sslport** and **clearport** keywords as described later in this section. If you need to match a path, use the HTTP header rewrite feature to rewrite the string. For information about the HTTP header rewrite feature, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

  The ACE supports the use of regular expressions for matching data strings. See Table 3-4 for a list of the supported characters that you can use in regular expressions.

  Note When matching data strings, the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use the brackets ([ ]) character classes to match these symbols (for example, enter www[.]xyz[.]com instead of www.xyz.com). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

- **sslport** *number1*—(Optional) Specifies the SSL port number from which the ACE translates a clear port number before sending the server redirect response to the client. Enter an integer from 1 to 65535. The default is 443.

- **clearport** *number2*—(Optional) Specifies the clear port number to which the ACE translates the SSL port number before sending a server redirect response to the client. Enter an integer from 1 to 65535. The default is 80.

For example, to specify SSL URL rewrite for the URL www.cisco.com or www.cisco.net using the default SSL port of 443 and a clear port of 8080, enter:

```
host1/Admin(config-actlist-modify)# ssl url rewrite location
www\.cisco\..* sslport 443 clearport 8080
```

In the above example, the ACE attempts to perform the following tasks:

- Match all HTTP redirects to http://www.cisco.com:8080 or http://www.cisco.net:8080
- Rewrite the HTTP redirects as https://www.cisco.com:443 or https://www.cisco.net:443
- Forward the HTTP redirects to the client

After you enter the **ssl url rewrite** command, associate the action list with a Layer 3 and Layer 4 policy map. See the "Associating an Action List with a Layer 7 HTTP Load-balancing Policy Map" section.

Table 3-4 Special Characters for Matching String Expressions

Convention	Description
.	One of any character.
.*	Zero or more of any character.
\.	Period (escaped).
[*charset*]	Match any single character from the range.
[^charset]	Do not match any character in the range. All other characters represent themselves.
()	Expression grouping.
(expr1 | expr2)	OR of expressions.
(expr)*	0 or more of expression.
(expr)+	1 or more of expression.
expr{m,n}	Repeat the expression between *m* and *n* times, where *m* and *n* have a range of 1 to 255.

Table 3-4 Special Characters for Matching String Expressions (continued)

Convention	Description
expr{m}	Match the expression exactly *m* times. The range for *m* is from 1 to 255.
expr{m,}	Match the expression *m* or more times. The range for *m* is from 1 to 255.
\a	Alert (ASCII 7).
\b	Backspace (ASCII 8).
\f	Form-feed (ASCII 12).
\n	New line (ascii 10).
\r	Carriage return (ASCII 13).
\t	Tab (ASCII 9).
\v	Vertical tab (ASCII 11).
\0	Null (ASCII 0).
\\	Backslash.
\x##	Any ASCII character as specified in two-digit hexadecimal notation.

Configuring HTTP Header Insertion of SSL Session Parameters

You can instruct the ACE to provide the server with SSL session parameter information that the ACE and the client negotiate during the SSL handshake, such as the cipher suite to use for encrypting the information or the SSL session ID. To forward this SSL session information to the server, the ACE inserts HTTP headers containing the negotiated session parameter fields that you specify into the HTTP requests that it receives over the client connection. The ACE then forwards the HTTP request to the server.

Note To prevent HTTP header spoofing, the ACE deletes any incoming HTTP headers that match one of the headers that it is going to insert into the HTTP request.

When you instruct the ACE to insert SSL session information, by default, the ACE inserts the HTTP header information into every HTTP request that it receives over the client connection because persistence rebalance is enabled by default. When the ACE and client need to renegotiate their connection, the ACE updates the HTTP header information that it sends to the server to reflect the new session parameters. If you do not want the ACE to insert the SSL header information into every HTTP request, disable persistence rebalance in an HTTP parameter map. You can also instruct the ACE to insert the session information into every HTTP request that it receives over the connection by creating an HTTP parameter map with the **header modify per-request** command enabled. You then reference the parameter map in the policy map that the ACE applies to the traffic. For information about creating an HTTP parameter map, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

Note The maximum amount of data that the ACE can insert is 2048 bytes. The ACE truncates the data if it exceeds this limit.

You can insert an HTTP header that contains specific SSL session information by using the **ssl header-insert session** command in action list modify configuration mode. To remove an HTTP header that contains an SSL session information field, use the **no** form of the command.

The syntax of this command is as follows:

ssl header-insert session *specific_field* [**prefix** *prefix_string* | **rename** *new_field_name*]

The keywords and arguments are as follows:

- *specific_field*—Session field name to insert into the HTTP header. See Table 3-5 for a list of the valid session field names.
- **prefix** *prefix_string*—(Optional) Inserts a prefix string before the specified SSL session field. For example, if you specify the prefix Acme-SSL for the SSL session field name Cipher-Name, then the field name becomes Acme-SSL-Session-Cipher-Name. Enter a text string. The maximum combined number of prefix string and field name characters that the ACE permits is 32.
- **rename** *new_field_name*—(Optional) Assigns a new name to the specified SSL session field. Enter an unquoted text string with no spaces. The maximum number of field name characters that the ACE permits is 32.

Note You can cannot configure both the **prefix** and **rename** options because they are mutually exclusive. Use the **rename** option when assigning a prefix to an SSL session field name that you are also renaming.

Table 3-5 lists the supported SSL session fields.

Table 3-5 SSL Session Information: SSL Session Fields

Session Field	Description
Cipher-Key-Size	Symmetric cipher key size. Format: Whole integer that specifies the length in bytes of the shared key. Example: Session-Cipher-Key-Size: 32
Cipher-Name	Symmetric cipher suite name. Format: OpenSSL version name of the cipher suite negotiated during the session. Example: Session-Cipher-Name: EXP1024-RC4-SHA
Cipher-Use-Size	Symmetric cipher use size. Format: Whole integer that specifies how many bytes of the Cipher-Key-Size are used. Depending on the algorithm in use, the entire number of bytes may not be used. Example: Session-Cipher-Use-Size: 7
Id	SSL Session ID. The default is 0. Format: 32-byte session ID negotiated during this session if a session ID is or has been negotiated, printed in big-endian format; hexadecimal without leading 0x and lowercase alphanumeric characters separated by a colon (:). Example: Session-Id: 75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25:75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25

Table 3-5 SSL Session Information: SSL Session Fields (continued)

Session Field	Description
Protocol-Version	Version of SSL or TLS. Format: String that indicates whether the SSL or TLS protocol is used followed by a version number. Example: Session-Protocol-Version: TLSv1

Table 3-5 SSL Session Information: SSL Session Fields (continued)

Session Field	Description
Step-Up	Use of SGC or StepUp cryptography. Format: String (yes/no) that indicates whether or not the ACE used Server Gated Cryptography (SGC) or Step-Up cryptography to increase the level of security by using 128-bit encryption. Example: Session-Step-Up: YES
Verify-Result	SSL session verify result. Format: String value that indicates the SSL session verify result. Possible values are as follows: • ok—The SSL session is established. • certificate is not yet valid—The client certificate is not yet valid. • certificate is expired—The client certificate has expired. • bad key size—The client certificate has a bad key size. • invalid not before field—The client certificate notBefore field is in an unrecognized format. • invalid not after field—The client certificate notAfter field is in an unrecognized format. • certificate has unknown issuer—The client certificate issuer is unknown. • certificate has bad signature—The client certificate contains a bad signature. • certificate has bad leaf signature—The client certificate contains a bad leaf signature. • unable to decode issuer public key—The ACE is unable to decode the issuer public key. • unsupported certificate—The client certificate is not supported. • certificate revoked— The client certificate has been revoked. • internal error—An internal error exists. Example: Session-Verify-Result: ok

For example, to insert the name of the cipher suite being used for the SSL session into the HTTP header, enter:

```
host1/Admin(config-actlist-modify)# ssl header-insert session
Cipher-Name
```

Repeat the **ssl header-insert session** command for each session parameter field that you want the server to receive.

For information about the counters that track the success rate of inserting the SSL HTTP header information, see Chapter 6, "Displaying SSL Information and Statistics."

Configuring HTTP Header Insertion of SSL Server Certificate Information

You can instruct the ACE to provide the server with information about the server certificate that resides on the ACE, such as the algorithm used for the public key or the certificate serial number. To forward this SSL session information to the server, the ACE inserts HTTP headers containing the server certificate fields that you specify into the HTTP requests that it receives over the client connection. The ACE then forwards the HTTP requests to the server.

Note To prevent HTTP header spoofing, the ACE deletes any incoming HTTP headers that match one of the headers that it is going to insert into the HTTP request.

When you instruct the ACE to insert SSL server certificate information, by default, the ACE inserts the HTTP header information into every HTTP request that it receives over the client connection because persistence rebalance is enabled by default. If you do not want the ACE to insert the information into every HTTP request that it receives over the connection, disable persistence rebalance in an HTTP parameter map. You can also instruct the ACE to insert the information into every HTTP request that it receives over the connection by creating an HTTP parameter map with the **header modify per-request** command enabled. You then reference the parameter map in the policy map that the ACE applies to the traffic. For information about creating an HTTP parameter map, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

Note The maximum amount of data that the ACE can insert is 512 bytes. The ACE truncates the data if it exceeds this limit.

You can insert an HTTP header that contains specific SSL server certificate information fields by using the **ssl header-insert server-cert** command in action list modify configuration mode. To remove an SSL HTTP header that contains a server certificate information field, use the **no** form of the command.

The syntax of this command is as follows:

ssl header-insert server-cert *specific_field* [**prefix** *prefix_string* | **rename** *new_field_name*]

The keywords and arguments are as follows:

- *specific_field*—Server certificate (ServerCert) field name to insert into the HTTP header. See Table 3-6 for a list of the valid server certificate field names.
- **prefix** *prefix_string*—(Optional) Inserts a prefix string before the specified server certificate field name. For example, if you specify the prefix Acme-SSL for the server certificate field name Authority-Key-Id, then the field name becomes Acme-SSL-ServerCert-Authority-Key-Id. Enter a text string. The maximum combined number of prefix string and field name characters that the ACE permits is 32.
- **rename** *new_field_name*—(Optional) Assigns a new name to the specified server certificate field. Enter an unquoted text string with no spaces. The maximum combined number of field name and prefix string characters that the ACE permits is 32.

Note You can cannot configure both the **prefix** and **rename** options because they are mutually exclusive. Use the **rename** option when assigning a prefix to a server certificate field name that you are also renaming.

Table 3-6 lists the supported SSL server certificate fields. Depending on how the certificate was generated and what key algorithm was used, all these fields may not be present for the certificate.

Table 3-6 SSL Session Information: Server Certificate Fields

ServerCert Field	Description
Authority-Key-Id	X.509 authority key identifier. Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 Authority Key Identifier. Example: ServerCert-Authority-Key-Identifier:16:13:15:97:FD:8E:16:B9:D2:99
Basic-Constraints	X.509 basic constraints. Format: String listing whether the certificate subject can act as a certificate authority. Possible values are CA=TRUE or CA=FALSE. Example: ServerCert-Basic-Constraints: CA=TRUE
Certificate-Version	X.509 certificate version. Format: Numerical X.509 version (3, 2, or 1), followed by the ASN.1 defined value for X.509 version (2, 1, or 0) in parentheses. Example: ServerCert-Certificate-Version: 3 (0x2)
Data-Signature-Alg	X.509 hashing and encryption method. Format: md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 algorithm used to sign the certificate and algorithm parameters. Example: ServerCert-Signature-Algorithm: md5WithRSAEncryption
Fingerprint-SHA1	SHA1 hash output of the certificate. Format: ASCII string of hexadecimal bytes separated by colons. Example: ServerCert-Fingerprint-SHA1: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1A:BA

Table 3-6 SSL Session Information: Server Certificate Fields (continued)

ServerCert Field	Description
Issuer	X.509 certificate issuer's distinguished name. Format: String of characters representing the certificate authority that issued this certificate. Example: ServerCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@expampleca.com, 0=Root
Issuer-CN	X.509 certificate issuer's common name. Format: String of characters representing the common name of the certificate issuer. Example: ServerCert-Issuer-CN: www.expampleca.com
Not-After	Date after which the certificate is not valid. Format: Universal time string or generalized time string in the Not After date of the Validity field. Example: ServerCert-Not-After: Dec 12 22:45:13 2014 GMT
Not-Before	Date before which the certificate is not valid. Format: Universal time string or generalized time string in the Not Before date of the Validity field. Example: ServerCert-Not-Before: Dec 12 22:45:13 2011 GMT
Public-Key-Algorithm	Algorithm used for the public key. Format: rsaEncryption, rsa, or dsaEncryption public key algorithm used to create the public key in the certificate. Example: ServerCert-Public-Key-Algorithm: rsaEncryption
RSA-Exponent	Public RSA exponent. Format: Whole integer representing the RSA algorithm exponent (e). Example: ServerCert-RSA-Exponent: 65537

Table 3-6 SSL Session Information: Server Certificate Fields (continued)

ServerCert Field	Description
RSA-Modulus	RSA algorithm modulus. Format: RSA algorithm modulus (n) printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character. Together with the exponent (e), this modulus forms the public key portion in the RSA certificate. Example: ServerCert-RSA-Modulus: + 00:d8:1b:94:de:52: a1:20:51:b1:77
RSA-Modulus-Size	Size of the RSA public key. Format: Number of bits as a whole integer of the RSA modulus (typically, 512, 1024, or 2048), followed by the word bit. Example: ServerCert-RSA-Modulus-Size: 1024 bit
Serial-Number	Certificate serial number. Format: Whole integer value assigned by the certificate authority; this can be any arbitrary integer value. Example: ServerCert-Serial-Number: 2
Signature	Certificate signature. Format: Secure hash of the other fields in the certificate and a digital signature of the hash printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters and separated by a colon (:) character. Example: ServerCert-Signature: 33:75:8e:a4:05:92:65
Signature-Algorithm	Certificate signature algorithm. Format: md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 for the secure hash algorithm. Example: ServerCert-Signature-Algorithm: nmd5WithRSAEncryption

Table 3-6 SSL Session Information: Server Certificate Fields (continued)

ServerCert Field	Description
Subject	X.509 subject's distinguished name. Format: String of characters representing the subject that owns the private key being certified. Example: ServerCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root
Subject-CN	X.509 subject's common name. Format: String of characters that represents the common name of the certificate issuer. Example: ServerCert-Subject-CN: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root
Subject-Key-Id	X.509 subject key identifier. Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 subject key identifier. Example: ServerCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99

For example, to insert the server certificate distinguished name into the HTTP header, enter:

```
host1/Admin(config-actlist-modify)# ssl header-insert server-cert
Subject
```

Repeat the **ssl header-insert server-cert** command for each server certificate field that you want the server to receive.

For information about the counters that track the success rate of inserting the SSL HTTP header information, see Chapter 6, "Displaying SSL Information and Statistics."

Configuring HTTP Header Insertion of SSL Client Certificate Information

When you configure the ACE for client authentication, you can instruct the ACE to provide the server with information about the client certificate that the ACE receives from the client. This SSL session information enables the server to properly manage the client request and can include certificate information such as the certificate serial number or the public key algorithm used to create the public key in the certificate. To forward the SSL session information to the server, the ACE inserts HTTP headers containing the client certificate fields that you specify into the HTTP requests that it receives over the client connection. The ACE then forwards the HTTP requests to the server.

Note To prevent HTTP header spoofing, the ACE deletes any incoming HTTP headers that match one of the headers that it is going to insert into the HTTP request.

When you instruct the ACE to insert SSL client certificate information, by default, the ACE inserts the HTTP header information into every HTTP request that it receives over the client connection because persistence rebalance is enabled by default. If you do not want the ACE to insert the information into every HTTP request that it receives over the connection, disable persistence rebalance in an HTTP parameter map. You can also instruct the ACE to insert the information into every HTTP request that it receives over the connection by creating an HTTP parameter map with the **header modify per-request** command enabled. You then reference the parameter map in the policy map that the ACE applies to the traffic. For information about creating an HTTP parameter map, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

Note You must have the ACE configured for client authentication to insert an HTTP header with SSL client certificate field information (see the "Enabling Client Authentication" section). If you configure header insertion but do not configure the ACE for client authentication, no header information is inserted and the counters that track the header insertion operation do not increment (see Chapter 6, "Displaying SSL Information and Statistics").

Note The maximum amount of data that the ACE can insert is 512 bytes. The ACE truncates the data if it exceeds this limit.

You can insert an HTTP header that contains specific SSL client certificate fields by using the **ssl header-insert client-cert** command in action list modify configuration mode. To remove client certificate information from the HTTP header, use the **no** form of the command.

The syntax of this command is as follows:

ssl header-insert client-cert *specific_field* [**prefix** *prefix_string* | **rename** *new_field_name*]

The keywords and arguments are as follows:

- *specific_field*—Client certificate (ClientCert) field name to insert into the HTTP header. See Table 3-7 for a list of the valid server certificate field names.
- **prefix** *prefix_string*—(Optional) Inserts a prefix string before the specified client certificate field name. For example, if you specify the prefix Acme-SSL for the client certificate field name Authority-Key-Id, then the field name becomes Acme-SSL-ClientCert-Authority-Key-Id. Enter a text string. The maximum combined number of prefix string and field name characters that the ACE permits is 32.
- **rename** *new_field_name*—(Optional) Assigns a new name to the specified client certificate field. Enter an unquoted text string with no spaces. The maximum combined number of field name and prefix string characters that the ACE permits is 32.

Note You can cannot configure both the **prefix** and **rename** options because they are mutually exclusive. Use the **rename** option when assigning a prefix to a client certificate field name that you are also renaming.

Table 3-7 lists the supported SSL client certificate fields. Depending on how the certificate was generated and what key algorithm was used, all of these fields may not be present for the certificate.

Table 3-7 SSL Session Information: SSL Client Certificate Fields

ClientCert Field	Description
Authority-Key-Id	X.509 authority key identifier. Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 Authority Key Identifier. Example: ClientCert-Authority-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99
Basic-Constraints	X.509 basic constraints. Format: String that indicates if the certificate subject can act as a certificate authority. Possible values are CA=TRUE or CA=FALSE basic constraints. Example: ClientCert-Basic-Constraints: CA=TRUE
Certificate-Version	X.509 certificate version. Format: Numerical X.509 version (3, 2, or 1), followed by the ASN.1 defined value for X.509 version (2, 1, or 0) in parentheses. Example: ClientCert-Certificate-Version: 3 (0x2)
Data-Signature-Alg	X.509 hashing and encryption method. Format: md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 algorithm used to sign the certificate and algorithm parameters. Example: ClientCert-Signature-Algorithm: md5WithRSAEncryption
Fingerprint-SHA1	SHA1 hash of the certificate. Format: ASCII string of hexadecimal bytes separated by colons. Example: ClientCert-Fingerprint-SHA1: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1:BA

Table 3-7 SSL Session Information: SSL Client Certificate Fields (continued)

ClientCert Field	Description
Issuer	X.509 certificate issuer's distinguished name. Format: String of characters representing the certificate authority that issued the certificate. Example: ClientCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@exampleca.com, 0=Root
Issuer-CN	X.509 certificate issuer's common name. Format: String of characters representing the common name of the certificate issuer. Example: ClientCert-Issuer-CN: www.exampleca.com
Not-After	Date after which the certificate is not valid. Format: Universal time string or generalized time string in the Not After date of the Validity field. Example: ClientCert-Not-After: Dec 12 22:45:13 2014 GMT
Not-Before	Date before which the certificate is not valid. Format: Universal time string or generalized time string in the Not Before date of the Validity field. Example: ClientCert-Not-Before: Dec 12 22:45:13 2011 GMT
Public-Key-Algorithm	Algorithm used for the public key. Format: rsaEncryption, rsa, or dsaEncryption public key algorithm used to create the public key in the certificate. Example: ClientCert-Public-Key-Algorithm: rsaEncryption
RSA-Exponent	Public RSA exponent. Format: Printed as a whole integer for the RSA algorithm exponent (e). Example: ClientCert-RSA-Exponent: 65537

Table 3-7 SSL Session Information: SSL Client Certificate Fields (continued)

ClientCert Field	Description
RSA-Modulus	RSA algorithm modulus. Format: RSA algorithm modulus (n) printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character. Together with the exponent (e), this modulus forms the public key portion in the RSA certificate Example: ClientCert-RSA-Modulus: +00:d8:1b:94:de:52:a1:20:51:b1:77
RSA-Modulus-Size	Size of the RSA public key. Format: Number of bits as a whole integer of the RSA modulus (typically, 512, 1024, or 2048) followed by the word bit. Example: ClientCert-RSA-Modulus-Size: 1024 bit
Serial-Number	Certificate serial number. Format: Whole integer value assigned by the certificate authority; this can be any arbitrary integer value. Example: ClientCert-Serial-Number: 2
Signature	Certificate signature. Format: Secure hash of the other fields in the certificate and a digital signature of the hash printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character. Example: ClientCert-Signature: 33:75:8e:a4:05:92:65
Signature-Algorithm	Certificate signature algorithm. Format: md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 for the secure hash algorithm. Example: ClientCert-Signature-Algorithm: md5WithRSAEncryption

Table 3-7 SSL Session Information: SSL Client Certificate Fields (continued)

ClientCert Field	Description
Subject	X.509 subject's distinguished name. Format: String of characters representing the subject that owns the private key being certified. Example: ClientCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root
Subject-CN	X.509 subject's common name. Format: String of characters that represent the common name of the subject to whom the certificate has been issued. Example: ClientCert-Subject-CN: www.cisco.com
Subject-Key-Id	X.509 subject key identifier. Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 subject key identifier. Example: ClientCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99

For example, to insert the client certificate distinguished name into the HTTP header, enter:

```
host1/Admin(config-actlist-modify)# ssl header-insert client-cert
Subject
```

Repeat the **ssl header-insert client-cert** command for each client certificate field that you want the server to receive.

For information about the counters that track the success rate of inserting the SSL HTTP header information, see Chapter 6, "Displaying SSL Information and Statistics."

Associating an Action List with a Layer 7 HTTP Load-balancing Policy Map

You can associate an action list with a Layer 7 HTTP loadbalancing policy map by using the **action** command in policy map load balance class configuration mode. For more information about creating class maps and policy maps, see the "Creating a Layer 3 and Layer 4 Class Map for SSL Termination" and "Creating a Layer 3 and Layer 4 Policy Map for SSL Termination" sections.

Caution You must associate an action list configured with SSL HTTP header insertion with the class-default class map only.

The syntax of this command is as follows:

action *name*

The *name* argument is the identifier of an existing action list. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to associate an action list for SSL URL rewrite with a Layer 7 HTTP load-balancing policy map, enter:

```
host1/Admin(config)# policy-map type loadbalance http first-match
L7_POLICY
host1/Admin(config-pmap-lb)# class CLASS-DEFAULT
host1/Admin(config-pmap-lb-c)# action SSL_ACTLIST
```

To disassociate the action list from the policy map, enter:

```
host1/Admin(config-pmap-lb-c)# no action SSL_ACTLIST
```

Example Configurations Containing HTTP Header Insertion

This section contains the following example configurations:

- Inserting SSL Session Information Into the First HTTP Request Only
- Inserting SSL Session Information Into All HTTP Requests

Inserting SSL Session Information Into All HTTP Requests

This section contains a configuration example that includes an action list (ACTION-SSL-INS) for inserting SSL session information. The configuration uses the default method of inserting the session information into each HTTP request that it receives over the connection.

The configuration example is as follows:

```
serverfarm host SFARM-1
  rserver SERVER1
    inservice
 rserver SERVER2
    inservice

crypto authgroup A1
  cert CACERT3.PEM

ssl-proxy service SSL_PSERVICE_TERMINATION
  key RSAKEY.PEM
  cert RSACERT.PEM
  authgroup A1

class-map type http loadbalance match-all CM-1
  2 match http url /index.html

action-list type modify http ACTION-SSL-INS
  ssl header-insert session Id prefix SSL-
  ssl header-insert server-cert Issuer
  ssl header-insert client-cert Serial-Number rename
Client-Serial-Number

policy-map type loadbalance http first-match PM-HTTP-LB
  class CM-1
    serverfarm SFARM-1
   class class-default
      action ACTION-SSL-INS

policy-map multi-match SP-HTTP-LB-POLICY
  class VIP-MERCURY
    loadbalance vip inservice
    loadbalance policy PM-HTTP-LB
    loadbalance vip icmp-reply
    inspect http
    appl-parameter http advanced-options HTTP-PMAP
    ssl-proxy server SSL_PSERVICE_TERMINATION
```

```
interface vlan 2524
  ip address 2001:DB8:1::1/64 <---------- IPv6 address
  or
  ip address 192.168.1.1 255.255.255.0 <--IPv4 address
  access-group input ALL
  service-policy input SP-HTTP-LB-POLICY
  service-policy input MGMT-POLICY
  no shutdown
```

Inserting SSL Session Information Into the First HTTP Request Only

This section contains a configuration example that includes an action list (ACTION-SSL-INS) for inserting SSL session information. The configuration includes an HTTP parameter map (HTTP-PMAP) that instructs the ACE to insert the session information into only the first HTTP request that the ACE receives over the connection. For this example, the parameter map uses the **no persistence-rebalance** command to disable HTTP header insertion into every HTTP request. For information about creating an HTTP parameter map, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

The configuration example is as follows:

```
serverfarm host SFARM-1
  rserver SERVER1
    inservice
 rserver SERVER2
    inservice

crypto authgroup A1
  cert CACERT3.PEM

ssl-proxy service SSL_PSERVICE_TERMINATION
  key RSAKEY.PEM
  cert RSACERT.PEM
  authgroup A1

class-map type http loadbalance match-all CM-1
  2 match http url /index.html

parameter-map type http HTTP-PMAP
  no persistence-rebalance

action-list type modify http ACTION-SSL-INS
  ssl header-insert session Id prefix SSL-
  ssl header-insert server-cert Issuer
```

```
  ssl header-insert client-cert Serial-Number rename
Client-Serial-Number

policy-map type loadbalance http first-match PM-HTTP-LB
  class CM-1
    serverfarm SFARM-1
   class class-default
    action ACTION-SSL-INS

policy-map multi-match SP-HTTP-LB-POLICY
  class VIP-MERCURY
    loadbalance vip inservice
    loadbalance policy PM-HTTP-LB
    loadbalance vip icmp-reply
    inspect http
    appl-parameter http advanced-options HTTP-PMAP
    ssl-proxy server SSL_PSERVICE_TERMINATION

interface vlan 2524
  ip address 2001:DB8:1::1/64 <---------- IPv6 address
  or
  ip address 192.168.1.1 255.255.255.0 <--IPv4 address
  access-group input ALL
  service-policy input SP-HTTP-LB-POLICY
  service-policy input MGMT-POLICY
  no shutdown
```

Creating a Layer 3 and Layer 4 Class Map for SSL Termination

The class map that you associate with a policy map acts as a filter for traffic that matches the criteria that you specify. For SSL termination, you can define the match criteria based on one or more of the following traffic characteristics:

- Access list
- Virtual IP address
- Source IP address and subnet mask
- Destination IP address and subnet mask
- TCP/UDP port number or port range

You can create a Layer 3 and Layer 4 class map by using the **class-map** command in configuration mode. For details on creating and configuring a Layer 3 and Layer 4 class map, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

Creating a Layer 3 and Layer 4 Policy Map for SSL Termination

For SSL termination, you configure the ACE so that it is recognized as an SSL server by a client. To accomplish this, you configure a Layer 3 and Layer 4 policy map that the ACE applies to the inbound traffic. The policy map uses the Layer 3 and Layer 4 class map that you associate with it to determine whether the inbound traffic matches the criteria that you specify. When a match is found, the ACE engages the client in the SSL handshake and establishes an SSL session using the parameters that you specify in the associated SSL proxy server service.

This section contains the following topics:

Creating a Layer 3 and Layer 4 Policy Map

You can create an SSL termination policy map by using the **policy-map** command in configuration mode.

The syntax of this command is as follows:

policy-map multi-match *policy_name*

The *policy_name* argument is the name that you assign to the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create the policy map L4POLICY, enter:

```
host1/Admin(config)# policy-map multi-match L4POLICY
```

After you create a policy map, the CLI enters into policy map configuration mode.

```
host1/Admin(config-pmap)#
```

To delete an existing policy map, enter:

```
host1/Admin(config)# no policy-map L4POLICY
```

For information on associating an SSL class map with the policy map, see the "Associating the Layer 3 and Layer 4 Class Map with the Policy Map" section.

Associating the Layer 3 and Layer 4 Class Map with the Policy Map

You can associate the Layer 3 and Layer 4 class map with the policy map by using the **class** command in policy map configuration mode.

The syntax of this command is as follows:

class *class-map*

The *class-map* argument is the name of an existing class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to associate the class map L4VIPCLASS with the policy map, enter:

```
host1/Admin(config)# policy-map multi-match L4POLICY
host1/Admin(config-pmap)# class L4VIPCLASS
```

After you associate a class map with the policy map, the CLI enters into policy-map class-map configuration mode.

```
host1/Admin(config-pmap-c)#
```

To remove the association of a class map to the policy map, enter:

```
host1/Admin(config-pmap)# no class L4VIPCLASS
```

For information on associating an SSL proxy service with the class map, see the "Associating an SSL Proxy Server Service with the Policy Map" section.

Associating an SSL Proxy Server Service with the Policy Map

You can associate an SSL proxy server service with the policy map by using the **ssl-proxy server** command in policy map class configuration mode.

The syntax of this command is as follows:

ssl-proxy server *pservice*

The *pservice* argument is the name of an existing SSL proxy server service. Enter an unquoted alphanumeric string with a maximum of 64 characters.

For example, to associate the SSL proxy server service PSERVICE_SERVER with the policy map, enter:

```
host1/Admin(config)# policy-map multi-match L4POLICY
host1/Admin(config-pmap)# class L4VIPCLASS
host1/Admin(config-pmap-c)# ssl-proxy server PSERVICE_SERVER
```

To remove the class map association, enter:

```
host1/Admin(config-pmap-c)# no ssl-proxy server PSERVICE_SERVER
```

Applying the Policy Map to the VLANs

This section describes how to apply the Layer 3 and Layer 4 policy map to the VLAN traffic. The ACE allows you to apply the policy globally to all VLANs within the current context or to a specific VLAN in the context.

This section contains the following topics:

- Applying the Policy Map Globally
- Applying the Policy Map to a Specific VLAN

Applying the Policy Map Globally

You can globally apply the policy map to all VLANs in the context by using the **service-policy** command in configuration mode.

The syntax of this command is as follows:

service-policy input *policy_name*

The *policy_name* argument is the name of an existing policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to globally apply the policy map L4POLICY to all VLANs in the context, enter:

```
host1/Admin(config)# service-policy input L4POLICY
```

To globally remove the policy from all VLANs, enter:

```
host1/Admin(config)# no service-policy input L4POLICY
```

Applying the Policy Map to a Specific VLAN

To apply a policy map to a specific VLAN interface, you must enter interface configuration mode by using the **interface** command in configuration mode.

The syntax of this command is as follows:

interface vlan *vlan*

The *vlan* argument is the context VLAN number. Enter an integer from 2 to 4094.

For example, to enter interface configuration mode for VLAN 10, enter:

```
host1/Admin(config)# interface vlan 10
host1/Admin(config-if)#
```

You can apply the policy map to the interface by using the **service-policy** command in interface configuration mode.

The syntax of this command is as follows:

service-policy input *policy-name*

The *policy-name* argument is the name of an existing policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to apply the policy map L4POLICY to VLAN 10, enter:

```
host1/Admin(config)# interface vlan 10
host1/Admin(config-if)# service-policy input L4POLICY
```

To remove the policy from the interface, enter:

```
host1/Admin(config-if)# no service-policy input L4POLICY
```

Example of an SSL Termination Configuration

The following example illustrates a running configuration of the ACE acting as an SSL proxy server; terminating SSL or TLS connections from a client and then establishing a TCP connection to an HTTP server. When the ACE terminates the SSL or TLS connection, it decrypts the cipher text from the client and transmits the data as clear text to the HTTP server. The SSL termination configuration appears in bold in the example.

IPv6 Example

```
access-list ACL1 line 10 extended permit ip any any

probe https GEN-HTTPS
  port 80
  interval 50
  faildetect 5
  expect status 200 200

serverfarm host SFARM1
  description SERVER FARM 1 FOR SSL TERMINATION
  probe GEN-HTTPS
  rserver SERVER1 80
    inservice
  rserver SERVER2 80
    inservice
  rserver SERVER3 80
    inservice
  rserver SERVER4 80
    inservice

serverfarm host SFARM2
  description SERVER FARM 2 FOR SSL TERMINATION
  probe GEN-HTTPS
  rserver SERVER5 80
    inservice
  rserver SERVER6 80
    inservice
  rserver SERVER7 80
    inservice
```

```
  rserver SERVER8 80
    inservice

parameter-map type ssl PARAMMAP_SSL_TERMINATION
  cipher RSA_WITH_3DES_EDE_CBC_SHA
  cipher RSA_WITH_AES_128_CBC_SHA priority 2
  cipher RSA_WITH_AES_256_CBC_SHA priority 3
  version all
parameter-map type connection TCP_PARAM
  syn-data drop
  exceed-mss allow

ssl-proxy service SSL_PSERVICE_SERVER
  ssl advanced-options PARAMMAP_SSL_TERMINATION
  key MYKEY.PEM
  cert MYCERT.PEM

class-map type http loadbalance match-all L7_SERVER_CLASS
  description Sticky for SSL Testing
  2 match http url .*.jpg
  3 match source-address 2001:DB8:1::1/64
class-map type http loadbalance match-all L7_SLB-HTTP_CLASS
  2 match http url .*
  3 match source-address 2001:DB8:1::1/64
class-map match-all L4_SSL-TERM_CLASS
  description SSL Termination VIP
  2 match virtual-address 2001:DB8:1::130/64 tcp eq https

policy-map type loadbalance first-match L7_SSL-TERM_POLICY
  class L7_SERVER_CLASS
    serverfarm SFARM1
    insert-http I_AM header-value "SSL_TERM"
    insert-http SRC_Port header-value "%ps"
    insert-http DEST_IP header-value "%id"
    insert-http DEST_Port header-value "%pd"
    insert-http SRC_IP header-value "is"
  class L7_SLB-HTTP_CLASS
    serverfarm SFARM1
    insert-http I_AM header-value "SSL_TERM"
    insert-http SRC_Port header-value "%ps"
    insert-http DEST_IP header-value "%id"
    insert-http DEST_Port header-value "%pd"
    insert-http SRC_IP header-value "is"
policy-map multi-match L4_SSL-VIP_POLICY
  class L4_SSL-TERM_CLASS
    loadbalance vip inservice
    loadbalance policy L7_SSL-TERM_POLICY
    loadbalance vip icmp-reply
```

```
    ssl-proxy server SSL_PSERVICE_SERVER
    connection advanced-options TCP_PARAM

interface vlan 120
  description Upstream VLAN_120 - Clients and VIPs
  ip address 2001:DB8:120::1/64
  fragment chain 20
  fragment min-mtu 68
  access-group input ACL1
  nat-pool 1 2001:DB8:120::70 2001:DB8:120::7F/64 pat
  service-policy input L4_SSL-VIP_POLICY
  no shutdown
ip route 2001:DB8:120::100/64 2001:DB8:120::B
```

IPv4 Example

```
access-list ACL1 line 10 extended permit ip anyv6 anyv6

probe https GEN-HTTPS
  port 80
  interval 50
  faildetect 5
  expect status 200 200

serverfarm host SFARM1
  description SERVER FARM 1 FOR SSL TERMINATION
  probe GEN-HTTPS
  rserver SERVER1 80
    inservice
  rserver SERVER2 80
    inservice
  rserver SERVER3 80
    inservice
  rserver SERVER4 80
    inservice

serverfarm host SFARM2
  description SERVER FARM 2 FOR SSL TERMINATION
  probe GEN-HTTPS
  rserver SERVER5 80
    inservice
  rserver SERVER6 80
    inservice
  rserver SERVER7 80
    inservice
  rserver SERVER8 80
    inservice
```

```
parameter-map type ssl PARAMMAP_SSL_TERMINATION
  cipher RSA_WITH_3DES_EDE_CBC_SHA
  cipher RSA_WITH_AES_128_CBC_SHA priority 2
  cipher RSA_WITH_AES_256_CBC_SHA priority 3
  version all
parameter-map type connection TCP_PARAM
  syn-data drop
  exceed-mss allow

ssl-proxy service SSL_PSERVICE_SERVER
  ssl advanced-options PARAMMAP_SSL_TERMINATION
  key MYKEY.PEM
  cert MYCERT.PEM

class-map type http loadbalance match-all L7_SERVER_CLASS
  description Sticky for SSL Testing
  2 match http url .*.jpg
  3 match source-address 192.168.130.0 255.255.255.0
class-map type http loadbalance match-all L7_SLB-HTTP_CLASS
  2 match http url .*
  3 match source-address 192.168.130.0 255.255.255.0
class-map match-all L4_SSL-TERM_CLASS
  description SSL Termination VIP
  2 match virtual-address 192.168.130.11 tcp eq https

policy-map type loadbalance first-match L7_SSL-TERM_POLICY
  class L7_SERVER_CLASS
    serverfarm SFARM1
    insert-http I_AM header-value "SSL_TERM"
    insert-http SRC_Port header-value "%ps"
    insert-http DEST_IP header-value "%id"
    insert-http DEST_Port header-value "%pd"
    insert-http SRC_IP header-value "is"
  class L7_SLB-HTTP_CLASS
    serverfarm SFARM1
    insert-http I_AM header-value "SSL_TERM"
    insert-http SRC_Port header-value "%ps"
    insert-http DEST_IP header-value "%id"
    insert-http DEST_Port header-value "%pd"
    insert-http SRC_IP header-value "is"
policy-map multi-match L4_SSL-VIP_POLICY
  class L4_SSL-TERM_CLASS
    loadbalance vip inservice
    loadbalance policy L7_SSL-TERM_POLICY
    loadbalance vip icmp-reply
    ssl-proxy server SSL_PSERVICE_SERVER
    connection advanced-options TCP_PARAM
```

```
interface vlan 120
  description Upstream VLAN_120 - Clients and VIPs
  ip address 192.168.120.1 255.255.255.0
  fragment chain 20
  fragment min-mtu 68
  access-group input ACL1
  nat-pool 1 192.168.120.70 192.168.120.80 netmask 255.255.255.0 pat
  service-policy input L4_SSL-VIP_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```