Cisco Application Control Engine Module SSL Configuration Guide

Software Version A4(1.0) and A4(2.0)
February 2011

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-23570-01

Cisco Application Control Engine Module SSL Configuration Guide

CONTENTS

Preface

This guide describes how to configure the Secure Sockets Layer (SSL) features on a Cisco Application Control Engine (ACE) module installed in a Catalyst 6500 series switch or a Cisco 7600 series router, hereinafter referred to as the switch or router, respectively.

This preface contains the following major sections:

Audience

This guide is intended for the following trained and qualified service personnel who are responsible for configuring the ACE:

- System administrator
- System operator

How to Use This Guide

This guide is organized into the following chapters:

Chapter	Description
Chapter 1, Overview	Provides an overview of Secure Sockets Layer (SSL) cryptography and the ACE SSL features.
Chapter 2, Managing Certificates and Keys	Describes how to manage SSL certificate and key pair files on the ACE, including how to import and export certificate and key pair files.
Chapter 3, Configuring SSL Termination	Describes how to configure the ACE as an SSL proxy server to perform SSL termination between itself and a client.
Chapter 4, Configuring SSL Initiation	Describes how to configure the ACE as an SSL proxy client to perform SSL initiation between itself and a web server.
Chapter 5, Configuring End-to-End SSL	Describes how to configure the ACE as an SSL proxy client and an SSL proxy server to perform both SSL termination and SSL initiation, providing an end-to-end SSL solution between the client and the web server.
Chapter 6, Displaying SSL Information and Statistics	Describes how to display data and statistics related to your ACE SSL configuration.

Related Documentation

In addition to this guide, the ACE documentation set includes the following documents:

Document Title	Description
Release Note for the Cisco Application Control Engine Module	Provides information about operating considerations, caveats, and command-line interface (CLI) commands for the ACE.
Cisco Application Control Engine Module Hardware Installation Note	Provides information for installing the ACE into the Catalyst 6500 series switch or a Cisco 7600 series router.
Cisco Application Control Engine Module Getting Started Guide	Describes how to perform the initial setup and configuration tasks for the ACE.
Cisco Application Control Engine Module Administration Guide	Describes how to perform the following administration tasks on the ACE: • Setting up the ACE • Establishing remote access • Managing software licenses • Configuring class maps and policy maps • Managing the ACE software • Configuring SNMP • Configuring redundancy • Configuring the XML interface • Upgrading the ACE software
Cisco Application Control Engine Module Virtualization Configuration Guide	Describes how to operate your ACE in a single context or in multiple contexts.

Document Title	Description
Cisco Application Control Engine Module Routing and Bridging Configuration Guide	Describes how to perform the following routing and bridging tasks on the ACE: • Configuring VLAN interfaces • Configuring routing • Configuring bridging • Configuring Dynamic Host Configuration Protocol (DHCP)
Cisco Application Control Engine Module Server Load-Balancing Configuration Guide	Describes how to configure the following server load-balancing features on the ACE: • Real servers and server farms • Class maps and policy maps to load balance traffic to real servers in server farms • Server health monitoring (probes) • Stickiness • Firewall load balancing • TCL scripts
Cisco Application Control Engine Module Security Configuration Guide	Describes how to configure the following ACE security features: • Security access control lists (ACLs) • User authentication and accounting using a Terminal Access Controller Access Control System Plus (TACACS+), Remote Authentication Dial-In User Service (RADIUS), or Lightweight Directory Access Protocol (LDAP) server • Application protocol and HTTP deep packet inspection • TCP/IP normalization and termination parameters • Network Address Translation (NAT)

Document Title	Description
Cisco Application Control Engine Module System Message Guide	Describes how to configure system message logging on the ACE. This guide also lists and describes the system log (syslog) messages generated by the ACE.
Cisco Application Control Engine Module Command Reference	Provides an alphabetical list and descriptions of all CLI commands by mode, including syntax, options, and related commands.
Cisco CSM-to-ACE Conversion Tool User Guide	Describes how to use the CSM-to-ACE conversion tool to migrate Cisco Content Switching Module (CSM) running- or startup-configuration files to the ACE.
Cisco CSS-to-ACE Conversion Tool User Guide	Describes how to use the CSS-to-ACE conversion tool to migrate Cisco Content Services Switches (CSS) running-configuration or startup-configuration files to the ACE.
Cisco Application Control Engine (ACE) Troubleshooting Wiki	Describes the procedures and methodology in wiki format to troubleshoot the most common problems that you may encounter during the operation of your ACE.
Cisco Application Control Engine (ACE) Configuration Examples Wiki	Provides examples of common configurations for load balancing, security, SSL, routing and bridging, virtualization, and so on.

Symbols and Conventions

This publication uses the following conventions:

Convention	Description
boldface font	Commands, command options, and keywords are in **boldface**. Bold text also indicates a command in a paragraph.
italic font	Arguments for which you supply values are in *italics*. Italic text also indicates the first occurrence of a new term, book title, emphasized text.
{ }	Encloses required arguments and keywords.
[ ]	Encloses optional arguments and keywords.
{x | y | z}	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
`screen` font	Terminal sessions and information the system displays are in `screen` font.
`boldface screen` font	Information you must enter in a command line is in **`boldface screen`** font.
italic screen font	Arguments for which you supply values are in *italic screen* font.
^	The symbol ^ represents the key labeled Control—for example, the key combination ^D in a screen display means hold down the Control key while you press the D key.
< >	Nonprinting characters, such as passwords are in angle brackets.

The documentations use the following conventions:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the publication.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

For additional information about syntax formatting and navigating the ACE CLI, see the *Cisco Application Control Engine Module Command Reference*.

Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Open Source License Acknowledgements

The following acknowledgements pertain to this software license.

OpenSSL/Open SSL Project

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/).

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

License Issues

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgment: “This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)”
4. The names “OpenSSL Toolkit” and “OpenSSL Project” must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org.
5. Products derived from this software may not be called “OpenSSL” nor may “OpenSSL” appear in their names without prior written permission of the OpenSSL Project.
6. Redistributions of any form whatsoever must retain the following acknowledgment:

   “This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)”

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT “AS IS”' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO

EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

Original SSLeay License:

© 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com).

The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)".

   The word 'cryptographic' can be left out if the routines from the library being used are not cryptography-related.

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)".

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The license and distribution terms for any publicly available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution license [including the GNU Public License].

CHAPTER 1

Overview

Secure Sockets Layer (SSL) is an application-layer protocol that provides encryption technology for the Internet. SSL ensures the secure transmission of data between a client and a server through a combination of privacy, authentication, and data integrity. SSL relies upon certificates and private-public key exchange pairs for this level of security.

This chapter contains the following major sections:

SSL Cryptography Overview

The Cisco Application Control Engine (ACE) module uses a special set of SSL commands to perform the SSL cryptographic functions between a client and a server. The SSL functions include server authentication, private-key and public-key generation, certificate management, and data packet encryption and decryption.

The ACE supports SSL Version 3.0 and Transport Layer Security (TLS) Version 1.0. The ACE understands and accepts an SSL Version 2.0 ClientHello message, known as a hybrid 2/3 hello message, allowing dual-version clients to communicate with the ACE. When the client indicates SSL Version 3.0 in the Version 2.0 ClientHello, the ACE understands that the client can support SSL Version 3.0 and returns a Version 3.0 ServerHello message.

Note The ACE cannot pass network traffic if the client supports only SSL Version 2.0.

A typical SSL session with the ACE requires encryption ciphers to establish and maintain the secure connection. Cipher suites provide the cryptographic algorithms required by the ACE to perform key exchange, authentication, and Message Authentication Code (MAC). See the "Adding a Cipher Suite" section in Chapter 3, Configuring SSL Termination, for details about the supported cipher suites.

This section provides an overview of SSL cryptography as implemented in the ACE and contains the following topics:

- SSL Public Key Infrastructure
- SSL Handshake

SSL Public Key Infrastructure

SSL provides authentication, encryption, and data integrity in a Public Key Infrastructure (PKI). PKI is a set of policies and procedures that establishes a secure information exchange between devices. Three fundamental elements characterize the PKIs used in asymmetric cryptography:

- Confidentiality
- Authentication
- Message Integrity

These three elements provide a secure system for deploying e-commerce and a reliable environment for building virtually any type of electronic transactions, from corporate intranets to Internet-based e-business applications.

Confidentiality

Confidentiality ensures that unintended users cannot view the data. In PKIs, confidentiality is achieved by encrypting the data through a variety of methods. In SSL, specifically, large amounts of data are encrypted using one or more symmetric keys that are known only by the two endpoints. Because the symmetric key is usually generated by one endpoint, it must be transmitted securely to the other endpoint. The ACE supports the use of the *key exchange* mechanism for the secure transmission of a symmetric key between the ACE and its SSL peer.

In key exchange, one device generates the symmetric key and then encrypts it using an asymmetric encryption scheme before transmitting the key to its SSL peer. Asymmetric encryption requires that each device has a unique key pair that consists of a public key and a private key. The two keys are mathematically related; data that is encrypted using the public key can only be decrypted using the corresponding private key, and vice versa. A device shares its public key with its SSL peer but must keep its private key a secret.

The security of asymmetric encryption depends entirely on the fact that the private key is known only by the owner and not by any other party. If this key were compromised for any reason, a fraudulent web user (or website) could decrypt the stream containing the symmetric key and the entire data transfer. The most commonly used key exchange algorithm is the Rivest Shamir Adelman (RSA) algorithm.

For SSL, the sender encrypts the symmetric keys with the public key of the receiver to ensure that the private key of the receiver is the only key that can decrypt the transmission.

Authentication

Authentication ensures that one or more devices in the exchange can verify the identity of the other device. For example, assume a client is connecting to an e-commerce website. Before sending sensitive information such as a credit card number, the client verifies that the server is a legitimate e-commerce website. Both the client and the server may need to authenticate themselves to each other before beginning the transaction. In a financial transaction between two banks, both the client and the server must be confident of the other's identity. SSL facilitates this authentication through the use of digital certificates.

Digital certificates are a form of digital identification to prove the identity of the server to the client, or optionally, the client to the server. A certificate ensures that the identification information is correct and that the public key embedded in the certificate actually belongs to that client or server.

A Certificate Authority (CA) issues digital certificates in the context of a PKI, which uses public-key and private-key encryption to ensure security. CAs are trusted authorities who sign certificates to verify their authenticity. Digital certificates contain the following information:

- Details about the owner (the certificate subject)
- Details about CA (the certificate issuer)
- Public key for the certificate's subject
- Certificate validity and expiration dates
- Privileges associated with the certificate

As the certificate issuer, the CA uses a private key to sign the certificate. Upon receiving a certificate, a client uses the issuer's public key to decrypt and verify the certificate signature. This procedure ensures that the certificate was actually issued and signed by an authorized entity.

Public key certificates support *certificate hierarchies*. A CA creates a hierarchy of subsidiary authorities that share in the responsibility of issuing signed certificates. The CA that sits at the head of the hierarchy is known as the *root authority*. Each level in the hierarchy certifies the level below it, creating a hierarchy of trusted relationships known as *certificate chaining*. This process enables an entity that is verifying a certificate to trace the CA certificates back to the root authority, if needed, to find a CA in the hierarchy that it trusts.

A certificate remains valid until it expires or is revoked by the CA. When a CA revokes a certificate, it adds the certificate to a certificate revocation list (CRL) that lists any certificates that it previously issued but no longer considers valid.

Clients or servers connected to the ACE must have trusted certificates from the same CA or from different CAs in a hierarchy of trusted relationships (for example, A trusts B, and B trusts C, therefore, A trusts C).

Message Integrity

Message integrity assures the recipient of a message that the contents of the message have not been tampered with during transit. To ensure message integrity, SSL applies a message digest to the data before transmitting it. A message digest takes an arbitrary-length message and outputs a fixed-length string that is characteristic of the message.

An important property of the message digest is that it is extremely difficult to reverse. Simply appending a digest of the message to itself before sending it is not enough to guarantee integrity. An attacker can change the message and then change the digest accordingly.

Each message exchanged between SSL peers is protected by a message authentication code (MAC), which can be calculated by using a hash algorithm such as SHA or MD5. The MAC is a hash value of several pieces of data, including a secret value, the actual data being sent, and a sequence number. The secret value is the write session key. The sequence number is a 32-bit counter value. This data is processed by the hash algorithm to derive the MAC. Upon receipt of a message, the receiver verifies the MAC by using the read session key and the predicted sequence number and calculates the hash over the received data. If the two hash values do not match, the data stream has been modified in some way.

SSL Handshake

The client and server use the SSL handshake protocol to establish an SSL session between the two devices. During the handshake, the client and server negotiate the SSL parameters that they will use during the secure session. Figure 1-1 shows the client/server actions that occur during the SSL handshake.

Note The ACE does not replicate SSL and other terminated (proxied) connections from the active context to the standby context.

Figure 1-1 SSL Handshake

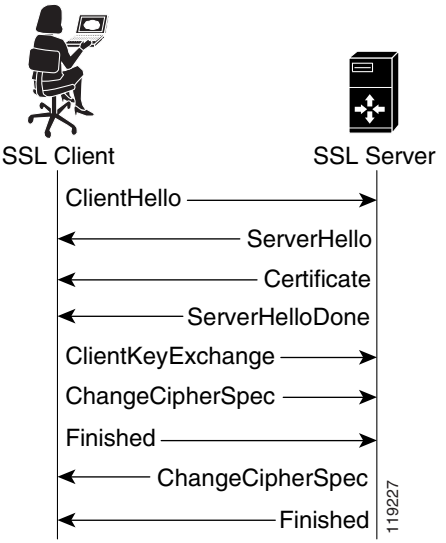


Table 1-1 describes the actions that take place between the client and the server during the SSL handshake.

Table 1-1 SSL Handshake Actions

Step	Message	Action
1	ClientHello	Client initiates the handshake by sending the ClientHello message that proposes the SSL parameters to use during the SSL session.
2	ServerHello	Server responds with the ServerHello message that contains the SSL parameters that it selects for use during the SSL session.
3	Certificate	Server sends the client its public key certificate.
4	ServerHelloDone	Server concludes its part of the SSL negotiations.
5	ClientKeyExchange	Client sends session key information that it encrypts using the server's public key.

Table 1-1 SSL Handshake Actions (continued)

Step	Message	Action
6	ChangeCipherSpec	Client instructs the server to activate the negotiated SSL parameters for all future messages that it sends.
7	Finished	Client instructs the server to verify that the SSL negotiation has been successful.
8	ChangeCipherSpec	Server instructs the client to activate the negotiated SSL parameters for all future messages that it sends.
9	Finished	Server instructs the client to verify that the SSL negotiation has been successful.

ACE SSL Capabilities

Table 1-2 provides information on the SSL capabilities of the ACE.

Table 1-2 ACE SSL Capabilities

SSL Feature	Feature Type or Specification Supported by the ACE
SSL versions	• SSL Version 3.0 and Transport Layer Security (TLS) Version 1.0 • SSL Version 2.0 ClientHello message (hybrid 2/3 hello)
Public key exchange algorithm	RSA—512 bits, 768 bits, 1024 bits, 1536 bits, 2048 bits
Encryption types	• Data Encryption Standard (DES) • Triple-Strength Data Encryption Standard (3DES) • RC4 • AES

Table 1-2 ACE SSL Capabilities (continued)

SSL Feature	Feature Type or Specification Supported by the ACE
Hash types	• SSL MAC-MD5 • SSL MAC-SHA1 • SHA-224 • SHA-256 • SHA-384 • SHA-512
Cipher suites	• RSA_WITH_RC4_128_MD5 • RSA_WITH_RC4_128_SHA • RSA_WITH_DES_CBC_SHA • RSA_WITH_3DES_EDE_CBC_SHA • RSA_EXPORT_WITH_RC4_40_MD5 • RSA_EXPORT_WITH_DES40_CBC_SHA • RSA_EXPORT1024_WITH_RC4_56_MD5 • RSA_EXPORT1024_WITH_DES_CBC_SHA • RSA_EXPORT1024_WITH_RC4_56_SHA • RSA_WITH_AES_128_CBC_SHA • RSA_WITH_AES_256_CBC_SHA
Digital certificates	Supports all major digital certificates from Certificate Authorities (CAs), including the following: • VeriSign • Entrust • Netscape iPlanet • Windows 2000 Certificate Server • Thawte • Equifax • Genuity

Table 1-2 ACE SSL Capabilities (continued)

SSL Feature	Feature Type or Specification Supported by the ACE
Maximum number of certificates	4096
Maximum size of a certificate or key file	32 KB
Maximum number of key pairs	4096
Maximum number of concurrent SSL connections	250,000
Maximum number of SSL transactions per second (TPS)	1000 TPS (default) 30,000 TPS (requires an optional bundle license) See the *Cisco Application Control Engine Module Administration Guide* guide for information on ACE licensing options.
Maximum amount of SSL bandwidth	6 Gbps

ACE SSL Functions

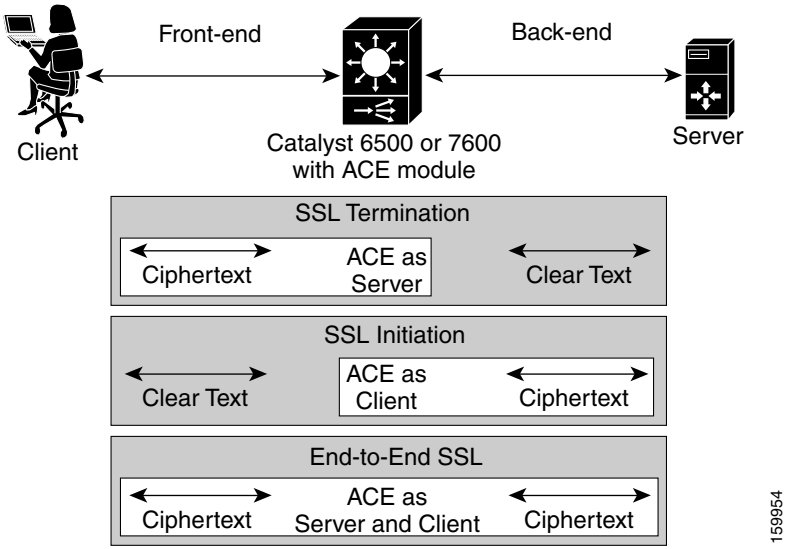
The ACE is responsible for all user authentication, public/private key generation, certificate management, and packet encryption and decryption functions between the client and the server.

You can partition the ACE into multiple contexts (virtual ACE devices). You configure each context with the certificate and key files that the context needs to establish an SSL session with its SSL peer. The ACE creates a secure storage area in flash memory for storing the certificates and keys associated with each context that you create.

To establish and maintain an SSL session between the ACE and its SSL peer, the ACE applies policy maps to the traffic that it receives. When the traffic characteristics match the attributes of a specific policy map, the ACE executes the actions associated with the policy map.

Depending on how you define the policy map, you can configure the ACE to act as a client or a server during an SSL session. Figure 1-2 shows the three basic SSL configurations of the ACE in which the ACE is used to encrypt and decrypt data between the client and the server.

Figure 1-2 ACE SSL Applications



The following sections provide an overview of the three ACE SSL applications:

- SSL Termination
- SSL Initiation
- End-to-End SSL

SSL Termination

SSL termination refers to configuring an ACE context for a front-end application in which the ACE operates as an SSL server that communicates with a client. When you create a Layer 3 and Layer 4 policy map to define the flow between an

ACE and a client, the ACE operates as a virtual SSL server by adding security services between a web browser (the client) and the HTTP connection (the server). All inbound SSL flows from a client terminate at the ACE.

After the connection is terminated, the ACE decrypts the ciphertext from the client and sends the data as clear text to an HTTP server. For information about configuring the ACE for SSL termination, see Chapter 3, Configuring SSL Termination.

SSL Initiation

SSL initiation refers to configuring an ACE context for a back-end application in which the ACE operates as a client that communicates with an SSL server. When you create a Layer 7 policy map to define the flow between an ACE and an SSL server, the ACE operates as a client and initiates the SSL session between the ACE and the server. SSL initiation enables the ACE to receive clear text from a client and then to establish an SSL session with an SSL server and join the client connection with the SSL server connection. The ACE encrypts the clear text that it receives from the client and sends the data as ciphertext to an SSL server. The SSL server can either be an ACE configured for SSL termination (virtual SSL server) or a real SSL server (web server).

On the outbound flow from the SSL server, the ACE decrypts the ciphertext from the server and sends clear text back to the client.

For more information on configuring the ACE for SSL initiation, see Chapter 4, Configuring SSL Initiation.

End-to-End SSL

End-to-end SSL refers to configuring an ACE context for both SSL termination and SSL initiation. You can configure the ACE for end-to-end SSL when you have an application that requires establishing a secure SSL channel between the client, the ACE, and the SSL server. For example, a transaction between banks requires end-to-end SSL to protect the financial information exchanged between the client and the server.

End-to-end SSL also allows the ACE to insert load-balancing and security information into the data. The ACE decrypts the ciphertext it receives and inserts the load-balancing and firewall information into the clear text. The ACE then reencrypts the data and passes the ciphertext to its intended destination.

For more information on configuring the ACE for end-to-end SSL initiation, see Chapter 5, Configuring End-to-End SSL.

ACE SSL Configuration Prerequisites

Before configuring your ACE for SSL operation, you must first configure it for server load balancing (SLB). During the SLB 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, modify the existing SLB class maps and policy maps with the SSL configuration requirements described in this guide for SSL termination, SSL initiation, or end-to end SSL.

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

CHAPTER 2

Managing Certificates and Keys

This chapter describes how to use the import and export functions to manage the various certificate and RSA key pair files on the Cisco Application Control Engine (ACE) module. This chapter also describes the process for creating and submitting a Certificate Signing Request (CSR), which you use to obtain a certificate from a Certificate Authority (CA).

This chapter contains the following major sections:

SSL Digital Certificates and Key Pairs

Digital certificates and key pairs are a form of digital identification for user authentication. CAs, such as VeriSign and Thawte, issue certificates that attest to the validity of the public keys they contain. A client or server certificate includes the following identification attributes:

- Name of the CA (the certificate issuer) and CA digital signature
- Serial number
- Name of the client or server (the certificate subject) that the certificate authenticates
- Subject's public key
- Time stamps that indicate the certificate's expiration date

A CA has one or more signing certificates that it uses for creating SSL certificates and certificate revocation lists (CRLs). Each signing certificate has a matching private key that is used to create the CA signature. The CA makes the signing certificates (with the public key embedded) available to the public, enabling anyone to access and use the signing certificates to verify that an SSL certificate or CRL was actually signed by a specific CA.

The ACE requires certificates and corresponding key pairs for the following applications:

- SSL termination—The ACE acts as an SSL proxy server and terminates the SSL session between it and the client. For SSL termination, you must obtain a server certificate and corresponding key pair.
- SSL initiation—The ACE acts as a client and initiates the SSL session between it and the SSL server. For SSL initiation, a client certificate and corresponding key pair can be used, but is not required unless the SSL server has client authentication enabled.

Note The ACE supports 4096 certificates and 4096 key pairs. It also supports wildcard certificates.

RSA key pairs are required by an ACE and its peer during the SSL handshake in order for the two devices to establish an SSL session. The key pair refers to a public key and its corresponding private (secret) key. During the handshake, the RSA key pairs are used to encrypt the session key that both devices will use to encrypt the data that follows the handshake.

For more information on the SSL handshake process, see the "SSL Handshake" section in Chapter 1, Overview.

Before you configure the ACE for SSL termination or SSL initiation, you must import a digital certificate and its corresponding public and private key pair to the desired ACE context.

In a redundant configuration, the ACE does not synchronize the SSL certificates and key pairs that are present in the active context with the standby context of a Fault Tolerant (FT) group. If the ACE performs a configuration synchronization and does not find the necessary certificates and keys on the standby, configuration synchronization fails and the standby context enters the STANDBY_COLD state.

To copy the certificates and keys to the standby context, you can export the certificates and keys from the active context to an FTP or TFTP server using the **crypto export** command, and then import the certificates and keys to the standby context using the **crypto import** command. You can also import the certificates and keys directly to the standby context using the same method that you used to import the certificates to the active context. This second method is required if the certificates and keys were imported to the active context as non-exportable. For more information about importing and exporting certificates and keys, see the "Importing or Exporting Certificate and Key Pair Files" section.

To return the standby context to the STANDBY_HOT state in this case, you must import the necessary SSL certificates and keys to the standby context, and then perform a bulk synchronization of the active context configuration by entering the following commands in configuration mode in the active context of the FT group:

1. **no ft auto-sync running-config**
2. **ft auto-sync running-config**

For more information about redundancy, see the *Cisco Application Control Engine Module Administration Guide*.

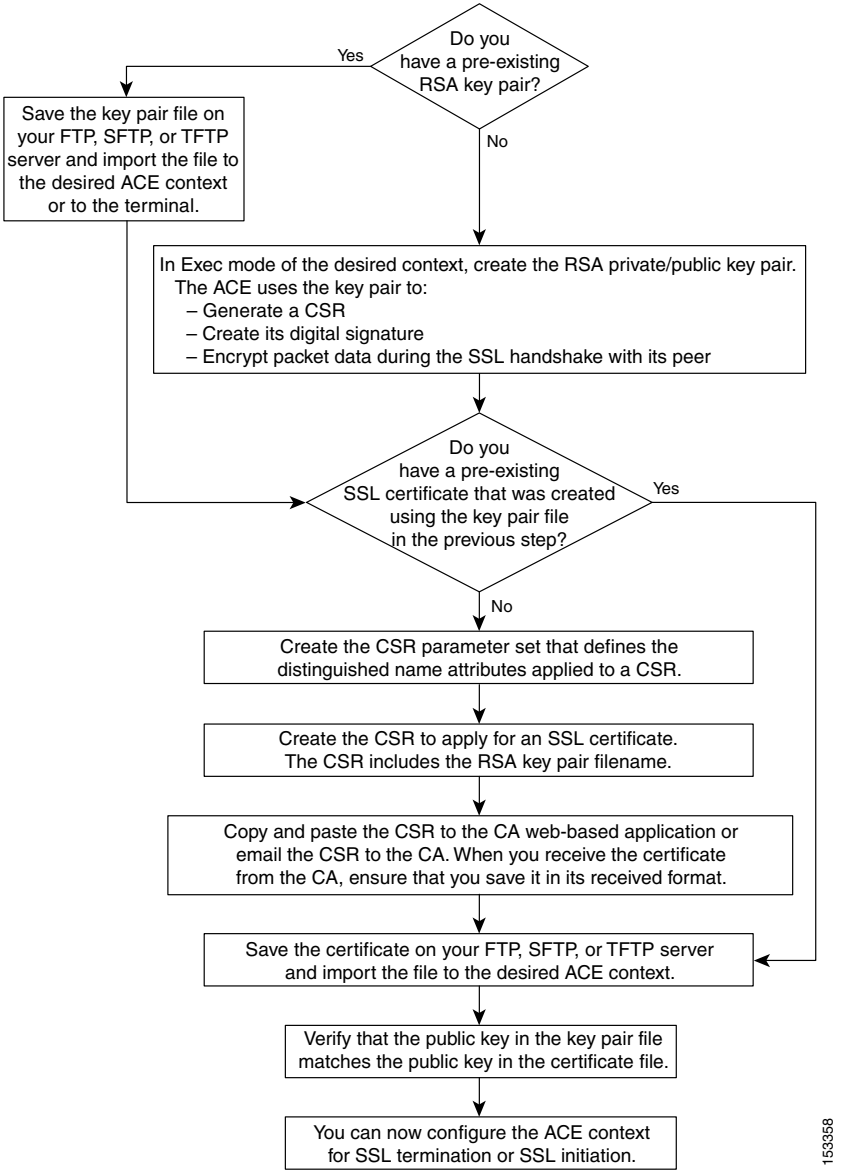
If you do not have a certificate and corresponding key pair, you can use the ACE to generate an RSA key pair of up to 2048 bits and a certificate signing request (CSR). You can create a CSR when you need to apply for a certificate from a CA. The CA signs the CSR and returns the authorized digital certificate to you.

The ACE also provides sample certificate and key pair files for demonstration purposes. For more information on these files, see the "Using the ACE Sample Certificate and Key Pair" section.

Note To implement strong security policies when generating key pairs or importing certificates and key pairs, you should understand the user roles of the ACE. For more information on user roles, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

Figure 2-1 provides an overview of how to configure an RSA key pair and SSL certificate for an ACE.

Figure 2-1 SSL Key and Certificate Configuration Overview



Using the ACE Sample Certificate and Key Pair

The ACE includes a preinstalled certificate and key pair. The certificate is intended for demonstration purposes only and does not have a valid domain. The certificate is a self-signed certificate with basic extensions named cisco-sample-cert. The key pair is an RSA 1024-bit key pair named cisco-sample-key. You can view these files by using the **show crypto files**, **show crypto key** and **show crypto certs** commands.

You can add these files to an SSL-proxy service by using the **cert** and **key** commands. For more information, see the "Specifying the Certificate" and "Specifying the Key Pair" sections, respectively. These files are available for use in any context with the filenames remaining the same in each context. You can verify the key pair by using the **crypto verify** command.

The ACE allows you to export these files but does not allow you to import any files with these names. You cannot delete these files by using the **crypto delete** command. However, when you upgrade the ACE, these files are overwritten with the files provided in the upgrade image.

Generating Key Pairs and Certificate Signing Requests

If you do not have preexisting certificates and matching key pairs, the ACE includes a series of certificate and key management utilities to generate a key pair or a CSR. When the CA signs your CSR, it becomes the certificate that you can use on the ACE.

If you have preexisting certificates and matching key pairs, you can import them to the desired context on the ACE. For information on importing certificates and private keys, see the "Importing or Exporting Certificate and Key Pair Files" section.

This section contains the following topics:

- Generating an RSA Key Pair
- Creating and Defining a CSR Parameter Set
- Generating a Certificate Signing Request

Generating an RSA Key Pair

The ACE supports the generation of RSA key pairs of up to 2048 bits. To generate an RSA key pair, use the **crypto generate key** command in Exec mode.

The syntax of this command is as follows:

crypto generate key [**non-exportable**] *bitsize filename*

The arguments and keywords are as follows:

- **non-exportable**—(Optional) Specifies that the ACE marks the key pair file as nonexportable, which means that you cannot export the key pair file from the ACE.
- *bitsize*—Key pair security strength. The number of bits in the key pair file defines the size of the RSA key pair used to secure web transactions. Longer keys produce a more secure implementation by increasing the strength of the RSA security policy. Available entries (in bits) are as follows:
  - 512 (least security)
  - 768 (normal security)
  - 1024 (high security, level 1)
  - 1536 (high security, level 2)
  - 2048 (high security, level 3)
- *filename*—Name that you assign to the generated RSA key pair file. Enter an unquoted alphanumeric string with a maximum of 40 characters. The key pair filename is used only for identification purposes by the ACE.

For example, to generate the RSA key pair file MYRSAKEY.PEM, enter:

```
host1/Admin# crypto generate key non-exportable 2048 MYRSAKEY.PEM
 Generating 2048 bit RSA key pair
host1/Admin#
```

After you generate an RSA key pair, you can do the following tasks:

- Create the CSR parameter set—The CSR parameter set defines the distinguished name attributes for the ACE to use during the CSR-generating process. For details on creating a CSR configuration file, see the "Creating and Defining a CSR Parameter Set" section.

- Generate a CSR for the RSA key pair file and transfer the CSR request to the CA for signing—This action provides an added layer of security because the RSA private key originates directly within the ACE and does not have to be transported externally. Each generated key pair must be accompanied by a corresponding certificate. For details on generating a CSR, see the "Generating a Certificate Signing Request" section.

Creating and Defining a CSR Parameter Set

A CSR parameter set defines the *distinguished name* attributes that the ACE applies to the CSR during the CSR-generating process. The distinguished name attributes provide the CA with the information that it needs to authenticate your site. Creating a CSR parameter set allows you to generate multiple CSRs with the same distinguished name attributes.

Each context on the ACE can contain up to eight CSR parameter sets.

This section contains the following topics:

Creating a CSR Parameter Set

You can create a CSR parameter set by using the **crypto csr-params** command in configuration mode. You can create up to eight CSR parameter sets per context.

The syntax of this command is as follows:

crypto csr-params *csr_param_name*

The *csr_param_name* argument is the name of the CSR parameter set. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create the CSR parameter set CSR_PARAMS_1, enter:

```
host1/Admin(config)# crypto csr-params CSR_PARAMS_1
```

After you create a CSR parameter set, the CLI enters CSR parameter configuration mode, where you define the distinguished name parameters.

```
host1/Admin(config-csr-params)#
```

The distinguished name consists of several required and optional parameters. The ACE requires that you define the following CSR parameter set attributes:

- Country name
- State or province
- Common name

Note If you do not configure the required CSR parameter set attributes, the ACE displays an error message when you try to generate a CSR using the CSR parameter set.

To delete an existing CSR parameter set, enter:

```
host1/Admin(config)# no csr-params CSR_PARAMS_1
```

To display information related to existing CSR parameter sets, use the **show crypto csr-params** command (see Chapter 6, Displaying SSL Information and Statistics).

Specifying a Common Name

You can define the required common name parameter in the CSR parameter set by using the **common-name** command in CSR parameter configuration mode.

The syntax of this command is as follows:

common-name *name*

The *name* argument should be the domain name or individual hostname of the SSL site. Enter an unquoted text string with no spaces or a quoted text string with spaces, and a maximum of 64 alphanumeric characters.

For example, to specify the common name WWW.ABC123.COM, enter:

```
host1/Admin(config-csr-params)# common-name WWW.ABC123.COM
```

To delete an existing common name from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no common-name
```

Specifying a Country

You can define the required country name parameter in the CSR parameter set by using the **country** command in CSR parameter configuration mode.

The syntax of this command is as follows:

country *name*

The *name* argument is the two-character code of the country where the SSL site resides (see the ISO 3166 list of country codes). Enter an unquoted text string with no spaces a maximum of two characters.

For example, to specify the country US (United States), enter:

```
host1/Admin(config-csr-params)# country US
```

To delete an existing country from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no country
```

Specifying a State or Province

You can define the required state name parameter in the CSR parameter set by using the **state** command in CSR parameter configuration mode.

The syntax of this command is as follows:

state *name*

The *name* argument is the name of the state where the SSL site resides. Enter an unquoted text string with a maximum of 40 alphanumeric characters including spaces.

For example, to specify the state GA (Georgia), enter:

```
host1/Admin(config-csr-params)# state GA
```

To delete an existing state from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no state
```

Specifying a Serial Number

You can define the required serial number parameter in the CSR parameter set by using the **serial-number** command in CSR parameters configuration mode.

Note The CA may choose to overwrite the serial number that you provide with their own serial number.

The syntax of this command is as follows:

serial-number *number*

The *number* argument is the serial number to assign to the certificate. Enter an unquoted text string with no spaces and a maximum of 16 alphanumeric characters.

For example, to specify the serial number 1001, enter:

```
host1/Admin(config-csr-params)# serial-number 1001
```

To delete an existing serial number from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no serial-number
```

Specifying a Locality

You can define the optional locality parameter in the CSR parameter set by using the **locality** command in CSR parameters configuration mode.

The syntax of this command is as follows:

locality *name*

The *name* argument is the locality name to include in the certificate. Enter an unquoted alphanumeric string with a maximum of 40 characters including spaces.

For example, to specify the locality ATHENS, enter:

```
host1/Admin(config-csr-params)# locality ATHENS
```

To delete an existing locality from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no locality ATHENS
```

Specifying an Organization Name

You can define the optional organization name parameter in the CSR parameter set by using the **organization-name** command in CSR parameters configuration mode.

The syntax of this command is as follows:

organization-name *name*

The *name* argument is the name of the organization to include in the certificate. Enter an unquoted alphanumeric string with a maximum of 64 characters including spaces.

For example, to specify the organization ABC123 SYSTEMS INC, enter:

```
host1/Admin(config-csr-params)# organization-name ABC123 SYSTEMS INC
```

To delete an existing organization name from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no organization-name ABC123 SYSTEMS
INC
```

Specifying an Organizational Unit

You can define the optional organization unit parameter in the CSR parameter set by using the **organization-unit** command in CSR parameters configuration mode.

The syntax of this command is as follows:

organization-unit *unit*

The *unit* argument is the name of the unit within an organization. Enter an unquoted alphanumeric string with a maximum of 64 characters including spaces.

For example, to specify the organization unit SSL ACCELERATOR, enter:

```
host1/Admin(config-csr-params)# organization-unit SSL ACCELERATOR
```

To delete an existing organization unit from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no organization-unit SSL ACCELERATOR
```

Specifying an E-mail Address

You can define the optional e-mail address parameter in the CSR parameter set by using the **email** command in CSR parameter configuration mode.

The syntax of this command is as follows:

email *address*

The *address* argument is the site e-mail address. Enter an unquoted alphanumeric string with no spaces and a maximum of 40 characters.

For example, to specify the e-mail address WEBADMIN@ABC123.COM, enter:

```
host1/Admin(config-csr-params)# email WEBADMIN@ABC123.COM
```

To delete an existing e-mail address from the CSR parameter set, enter:

```
host1/Admin(config-csr-params)# no email
```

Generating a Certificate Signing Request

You must generate a Certificate Signing Request (CSR) file if you are requesting a new certificate or renewing a certificate. When you submit the generated CSR to a CA, the CA signs the CSR using its RSA private key and the CSR becomes the certificate.

To generate a CSR file for an RSA key pair file and to transfer the certificate request to the CA, use the **crypto generate csr** command in Exec command mode of the context containing the RSA key pair file. This command generates a CSR in PKCS10 encoded in PEM format.

The syntax of this command is as follows:

crypto generate csr *csr_params key_filename*

The arguments are as follows:

- *csr_params*—CSR parameter set that contains the distinguished name attributes (see the "Creating and Defining a CSR Parameter Set" section). Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters. The ACE applies the distinguished name attributes contained in the CSR parameter set to the CSR.
- *key_filename*—RSA key pair filename that contains the key on which the CSR is built. (This key is the public key that the ACE embeds in the CSR.) Enter an unquoted alphanumeric string with no spaces and a maximum of 40 characters. Ensure that the RSA key pair file is loaded on the ACE for the current context. If the appropriate key pair does not exist, the ACE logs an error message.

For example, to generate a CSR that is based on the CSR parameter set CSR_PARAMS_1 and the RSA key pair in the file MYRSAKEY_1.PEM, enter:

```
host1/Admin# crypto generate csr CSR_PARAMS_1 MYRSAKEY_1.PEM
-----BEGIN CERTIFICATE REQUEST-----
MIIBcDCCARoCAQAwgbQxCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlTb211U3RhdGUx
ETAPBgNVBAcTCFNvbWVDaXR5MRcwFQYDVQQKEw5BIENvbXBhbnkgTmFtZTEbMBkG
A1UECxMSV2ViIEFkbWluaXN0cmF0aW9uMR0wGwYDVQQDExR3d3cuYWNvbXBhbnlu
YW1lLmNvbTEpMCcGCSqGSIb3DQEJARYad2ViYWRtaW5AYWNvbXBhbnluYW1lLmNv
bSAwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAtBNcNXMBqh5cJHbWFsqe9LMUO90T
pYG7gF5ODvtFGREMkHh7s6S1GF131IBWCSelG4Q/qEztjCO7y3pyjruVNQIDAQAB
oAAwDQYJKoZIhvcNAQEEBQADQQCMmXRdNPBDtMQPFvylpED5UMbeaMRm2iaC+1uZ
IaHmdoX4h5eckauu9pPgSxczau8w68PF+PDS9DAAMeRDxisL
-----END CERTIFICATE REQUEST-----
(config)#
```

The **crypto generate csr** command generates the PKCS10 CSR in PEM format and outputs the CSR to the screen. Most major CAs have web-based applications that require you to cut and paste the certificate request to the screen. If necessary, you can also cut and paste the CSR to a file. The ACE does not save a copy of the CSR locally. You can, however, regenerate the same request again at any time by using the same CSR parameter set and key pair file.

Note If you require a global site certificate that allows 128-bit encryption for export-restricted browsers, you must apply for a StepUp/GSC or chained certificate from the CA. After you receive the certificate, you must prepare it for use with the ACE. For more information, see the "Preparing a Global Site Certificate" section.

After you submit your CSR to the CA, you will receive your signed certificate in one to seven business days. When you receive your certificate, you must import the certificate to the desired ACE context (see the "Importing Certificate and Key Pair Files" section).

Preparing a Global Site Certificate

Export browsers may use 40-bit encryption to initiate connections to SSL servers. With a conventional server certificate, a browser and server complete the SSL handshake and use a 40-bit key to encrypt application data.

A global site certificate is an extended server certificate that allows 128-bit encryption for export-restricted browsers. When the server responds to a browser with a global certificate, the client automatically renegotiates the connection to use 128-bit encryption.

If you applied for a global site certificate from the CA, you must obtain both the global certificate and its intermediate CA certificate. The intermediate CA certificate validates the global certificate. You can obtain a VeriSign Intermediate certificate from the following URL:

http://www.verisign.com/support/install/intermediate.html

When you receive your global site certificate and intermediate CA certificate, you must import them to the desired ACE context (see the "Importing Certificate and Key Pair Files" section). Then you create a certificate chain group that includes both certificates (see the "Creating a Chain Group" section). The ACE sends the chain group to the client during the initial SSL handshake.

Importing or Exporting Certificate and Key Pair Files

You can import PEM-encoded certificate and key pair files to the ACE from a remote secure server. To transfer these files, we recommend that you use a secure encrypted transport mechanism between the ACE and the remote server.

The ACE supports the Secure Shell (SSHv2) protocol, which provides secure encryption communications between two hosts over an insecure network. For file transport between network devices, the ACE supports Secure File Transfer Protocol (SFTP), File Transfer Protocol (FTP), and Trivial File Transfer Protocol (TFTP). We recommend that you use SFTP because it is the only one of these three protocols that can provide a secure and encrypted connection.

Before you import a certificate or key pair file to the ACE, you must perform the following tasks:

- On the ACE, ensure that SSH access to the ACE is enabled to accept connections from SSH clients. By default, SSH access is enabled. If you restrict SSH access, the ACE will not accept connections from SSH clients and the import command will fail (an error message will be generated).

Note For details about configuring the Secure Shell daemon on the Catalyst 6500 series switch or Cisco 7600 series router, see the *Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide* or *Cisco 7600 Series Router Cisco IOS Software Configuration Guide*.

- On the SFTP server, verify that the server is properly configured. The user directory must point to the directory where the certificates and key pairs reside. This path is required by the ACE to ensure that certificates and keys are properly copied from or to the SFTP server.

This section contains the following topics:

Importing Certificate and Key Pair Files

The ACE supports the importation of PEM-encoded key pairs and certificates (including wildcard certificates) signed by keys of up to and including 2048 bits. You can import a certificate or key pair file to the ACE from a remote server by using the **crypto import** command in Exec mode. You can import either individual certificates and keys or multiple certificates and keys. Because a network device uses its certificate and corresponding public key together to prove its identity during the SSL handshake, be sure to import both the certificate file and its corresponding key pair file.

Note If you attempt to import a file that has the same filename of an existing local file, the ACE module does not overwrite the existing file. Before importing the updated file, you must either delete the local file or rename the imported file. For more information, see the "Deleting Certificate and Key Pair Files" or "Upgrading an SSL Certificate" section.

The syntax of this command is as follows:

crypto import [**non-exportable**] {**bulk sftp** [**passphrase** *passphrase*] *ip_addr username remote_url*} | {{**ftp** | **sftp**} [**passphrase** *passphrase*] *ip_addr username remote_filename local_filename*} | {**tftp** [**passphrase** *passphrase*] *ip_addr remote_filename local_filename*} | **terminal** *local_filename* [**passphrase** *passphrase*]

The keywords, arguments, and options are as follows:

- **non-exportable**—(Optional) Marks the imported file as nonexportable, which means that you cannot export the file from the ACE.
- **bulk**—Specifies the importing of multiple certificate or key pair files simultaneously.
- **ftp**—Specifies the File Transfer Protocol file transfer process.
- **sftp**—Specifies the Secure File Transfer Protocol file transfer process.

- **tftp**—Specifies the Trivial File Transfer Protocol file transfer process.
- **passphrase** *passphrase*—(Optional) Indicates that the file was created with a passphrase, which you must submit with the file transfer request in order to use the file. The passphrase can contain a maximum of 40 characters and pertains only to encrypted PEM files and PKCS files.
- *ip_addr*—IP address of the remote server. Enter an IP address in dotted-decimal notation (for example, 192.168.12.15).
- *username*—Username required to access the remote server. When you execute the command, the ACE prompts you for the password of the username on the remote server. Enter a name with a maximum of 64 characters. Do not include spaces or the following special characters:

  ;<>\|‘@$&()
- *remote_url*—Path to the certificate or key pair files that reside on the remote server to import with the **bulk** keyword. The ACE fetches only files specified by the path; it does not recursively fetch remote directories. Enter a filename path including wildcards (for example, /remote/path/*.pem). The ACE supports POSIX pattern matching notation, as specified in section 2.13 of the “Shell and Utilities” volume of IEEE Std 1003.1-2004. This notation includes the “*,” “?” and “[“ metacharacters.

  To fetch all files from a remote directory, specify a remote path that ends with a wildcard character (for example, /remote/path/*). Do not include spaces or the following special characters:

  ;<>\|‘@$&()

  The ACE module fetches all files on the remote server that matches the wildcard criteria. However, it imports only files with names that have a maximum of 40 characters. If the name of a file exceeds 40 characters, the ACE module does not import the file and discards it.
- *remote_filename*—Name of the certificate or key pair file that resides on the remote server to import.
- *local_filename*—Name to save the file to when imported to the ACE. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **terminal**—Allows you to import a file using cut and paste by pasting the certificate and key pair information to the terminal display. You must use the terminal method to display PEM files, which are in ASCII format.

The ACE supports the importation of PEM-encoded SSL certificates and keys with a maximum line width of 130 characters using the terminal. If an SSL certificate or key is not wrapped or it exceeds 130 characters per line, use a text editor such as the visual (vi) editor or Notepad to manually wrap the certificate or key to less than 130 characters per line. Alternatively, you can import the certificate or key by using SFTP, FTP, or TFTP with no regard to line width. Of these methods, we recommend SFTP because it is secure.

For example, to perform a bulk import all of the RSA key files from an SFTP server:

```
host1/Admin# crypto import bulk sftp 1.1.1.1 JOESMITH /USR/KEYS/*.PEM
Initiating bulk import. Please wait, it might take a while...
Connecting to 1.1.1.1...
Password: password
...
Bulk import complete. Summary:
    Network errors:                          0
    Bad file URL: 0
    Specified local files already exists:    0
    Invalid file names:                      1
    Failed reading remote files:             5
    Failed reading local files:              0
    Failed writing local files:              0
    Other errors:                            0
    Successfully imported:                   10
host1/Admin#
```

Note You cannot cancel the **crypto import bulk** command by entering Ctrl-C while it is executing.

After the bulk import, the ACE displays status counters for the number of errors and successful imports that occurred. The Other errors counter is a bucket for errors not included in the other counters. This counter increments for the following errors:

- Reaching the maximum number of crypto files
- Files with an unrecognized format (PEM, DER or PKCS12) or unacceptable key size (less than or equal to the 2,048-bit key)
- SFTP errors encountered during the import that are not accounted for in one of the other counters

Also, when the storage capacity of the crypto files is reached, the ACE displays the following message:

```
Warning: Crypto files' storage limit hit, further file import stopped.
```

To import the RSA key file MYRSAKEY.PEM from an SFTP server, enter:

```
host1/Admin# crypto import non-exportable sftp 1.1.1.1 JOESMITH
/USR/KEYS/MYRSAKEY.PEM MYKEY.PEM
Password: ********
Passive mode on.
Hash mark printing on (1024 bytes/hash mark).
#
Successfully imported file from remote server.
host1/Admin#
```

The following example shows how to use the **terminal** keyword to allow pasting of the certificate information to the file MYCERT.PEM:

```
host1/Admin# crypto import terminal MYCERT.PEM
Enter PEM formatted data ending with a blank line or "quit" on a line
by itself
--------BEGIN CERTIFICATE-----------------------
MIIC1DCCAj2gAwIBAgIDCCQAMA0GCSqGSIb3DQEBAgUAMIHEMQswCQYDVQQGEwJa
QTEVMBMGA1UECBMMV2VzdGVybiBDYXBlMRIwEAYDVQQHEwlDYXBlIFRvd24xHTAb
BgNVBAoTFFRoYXd0ZSBDb25zdWx0aW5nIGNjMSgwJgYDVQQLEx9DZXJ0aWZpY2F0
aW9uIFNlcnZpY2VzIERpdmlzaW9uMRkwFwYDVQQDExBUaGF3dGUgU2VydmVyIENB
MSYwJAYJKoZIhvcNAQkBFhdzZXJ2ZXItY2VydHNAdGhhd3RlLmNvbTAeFw0wMTA3
-----------END CERTIFICATE-----------------------
quit
```

Exporting Certificate and Key Pair Files

You can export a certificate or key pair file from the ACE to a remote server or the terminal screen by using the **crypto export** command in Exec command mode.

Note You cannot export a certificate or key pair file that you marked as nonexportable when you imported the file to the ACE.

The syntax of this command is as follows:

crypto export *local_filename* {**ftp** | **sftp** | **tftp** | **terminal**} {*ip_addr*} {*username*} {*remote_filename*}

The keywords, arguments, and options are as follows:

- *local_filename*—Name of the file that resides on the ACE to export. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **ftp**—Specifies the File Transfer Protocol file transfer process.
- **sftp**—Specifies the Secure File Transfer Protocol file transfer process. We recommend that you use SFTP because it is more secure than FTP or TFTP.
- **tftp**—Specifies the TFTP Trivial File Transfer Protocol file transfer process.
- **terminal**—Displays the file content on the terminal for copy and paste purposes. Use the **terminal** keyword when you need to cut and paste certificate or private key information from the console. You must use the terminal method to display PEM files, which are in ASCII format.
- *ip_addr*—IP address or name of the remote server. Enter an IP address in dotted-decimal notation (for example, 192.168.12.15).
- *username*—Username required to access the remote server. The ACE prompts you for your password when you execute the command.
- *remote_filename*—Name to save the file to on the remote server.

The remote server variables listed after the **terminal** keyword are used by the ACE when you select a transport type of **ftp**, **sftp**, or **tftp** (the variables are not used for **terminal**). If you select one of these transport types and do not define the remote server variables, the ACE prompts you for the variable information.

For example, to use SFTP to export the key file MYKEY.PEM from the ACE to a remote SFTP server, enter:

```
host1/Admin# crypto export MYKEY.PEM sftp 192.168.1.2 JOESMITH
/USR/KEYS/MYKEY.PEM
User password: ****
Writing remote file /usr/keys/mykey.pem
host1/Admin#
```

Upgrading an SSL Certificate

To upgrade an SSL certificate without disrupting active SSL sessions or pending SSL sessions, use the following steps:

Step 1 Import the new SSL certificate using the **crypto import** command in Exec mode and save it with a new name. See the "Importing Certificate and Key Pair Files" section. For example, to import a certificate from an SFTP server, enter the following command:

```
host1/Admin# crypto import non-exportable sftp 1.1.1.1 JOESMITH
/USR/CERTS/MY_CERT.PEM MY_NEW_CERT.PEM
Password: ********
Passive mode on.
Hash mark printing on (1024 bytes/hash mark).
#
Successfully imported file from remote server.
host1/Admin#
```

Step 2 While the SSL proxy service is actively processing flows, change the certificate file association within the SSL proxy service to the new certificate by using the **cert** command in SSL proxy configuration mode. See the "Specifying the Certificate" section in Chapter 3, Configuring SSL Termination.

For example, to associate the certificate in the MY_NEW_CERT.PEM certificate file with the PSERVICE_SERVER SSL proxy service, enter the following commands:

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

Verifying a Certificate Against a Key Pair

A digital certificate is built around the public key of a key pair and can only be used with one key pair. You can compare the public key in a certificate file with the public key in a key pair file and verify that they are identical by using the **crypto verify** command in Exec command mode.

Note If the public key in the certificate does not match the public key in the key pair file, the ACE logs an error message.

The syntax of this command is as follows:

crypto verify *key_filename cert_filename*

The arguments are as follows:

- *key_filename*—Name of the context key pair file that the ACE uses to verify against the specified certificate. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- *cert_filename*—Name of the context certificate file that the ACE uses to verify against the specified key pair. Enter an unquoted alphanumeric string with a maximum of 40 characters.

For example, to verify the public keys in the files MYRSAKEY.PEM and MYCERT.PEM match, enter:

```
host1/Admin# crypto verify myrsakey.pem mycert.pem
keypair in myrsakey.pem matches certificate in mycert.pem
```

The following example shows what the ACE displays when the public keys do not match:

```
host1/Admin# crypto verify myrsakey_2.pem mycert.pem
Keypair in myrsakey_2.pem does not match certificate in mycert.pem
host1/Admin#
```

Deleting Certificate and Key Pair Files

You can delete certificate and key pair files that are no longer valid by using the **crypto delete** command in Exec command mode. Because the ACE module does not overwrite existing certificate or key pair files, deleting the file allows you to import an updated file.

The syntax of this command is as follows:

crypto delete {*filename* | **all**}

The keywords and arguments are as follows:

- *filename*—Name of a specific certificate or key pair file to delete. Enter an unquoted alphanumeric string with a maximum of 40 characters.
- **all**—Deletes all certificate and key pair files from the context. It does not delete the preinstalled sample certificate and key files. When you use the **all** keyword, the ACE prompts you with the following message to verify the deletion:

```
This operation will delete all crypto files for this context from
the disk, but will not interrupt existing SSL services.  If new
SSL files are not applied SSL services will be disabled upon next
vip inservice or device reload.
Do you wish to proceed? (y/n)  [n]
```

To display a list of available certificate and key pair files loaded on the ACE, use the **show crypto files** command.

Note The **crypto delete** command deletes the specified context crypto files from flash memory; however, existing SSL services are not interrupted. If you do not replace the deleted SSL files, the SSL services are disabled the next time that you enter the **vip inservice** command or when a device reload occurs.

For example, to delete the key pair file MYRSAKEY.PEM, enter:

```
host1/Admin# crypto delete MYRASKEY.PEM
```

Creating a Chain Group

A chain groups specifies the *certificate chains* that the ACE sends to its peer during the handshake. A certificate chain is a hierarchal list of certificates that includes the subject's certificate, the root CA certificate, and any intermediate CA certificates. Using the information provided in a certificate chain, the certificate verifier can search for a trusted authority in the certificate hierarchal list back to the root CA. The verifier may find what it considers a trusted authority before reaching the root CA certificate, in which case, the verifier stops searching.

When defining an SSL proxy service, you can configure the service with a chain group (see the "Creating and Defining an SSL Proxy Service" section in Chapter 3, Configuring SSL Termination).

The ACE supports the following certificate chain group capabilities:

- A chain group can contain up to eight certificate chains.
- Each context on the ACE can contain up to eight chain groups.
- The maximum size of a chain group is 11 KB.

To create a chain group, use the **crypto chaingroup** command in configuration mode.

The syntax of this command is as follows:

crypto chaingroup *group_name*

The *group_name* argument is the name of the chain group. Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters.

For example, to create the chain group MYCHAINGROUP, enter:

```
host1/Admin(config)# crypto chaingroup MYCHAINGROUP
```

After you create a chain group, the CLI enters chaingroup configuration mode, where you add the required certificate files to the group.

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

To delete an existing chain group, enter:

```
host1/Admin(config)# no crypto chaingroup MYCHAINGROUP
```

You can add certificate files to the chain group by using the **cert** command in chaingroup configuration mode. You can configure a chaingroup with up to nine certificates.

The syntax of this command is as follows:

cert *cert_filename*

The *cert_filename* argument is the name of an existing certificate file stored on the ACE. Enter an unquoted alphanumeric string with a maximum of 40 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. See the "Creating and Defining an SSL Proxy Service" section in Chapter 3, Configuring SSL Termination.

Typically, it is not necessary to add the certificates to the chain group in any type of hierarchical order because the device that verifies the certificates determines the correct order. However, some mobile devices may not be able to order the certificates properly and will display an error message. In this case, you need to add the certificates to the chain group in the correct order.

To display a list of existing certificate files, use the **show crypto files** command (see the "Displaying Certificate Information" section in Chapter 6, Displaying SSL Information and Statistics).

For example, to add the certificate files MYCERTS.PEM and MYCERTS_2.PEM to the chain group, enter:

```
host1/Admin(config-chaingroup)# cert MYCERTS.PEM
host1/Admin(config-chaingroup)# cert MYCERTS_2.PEM
```

To remove a certificate file from the chain group, enter:

```
host1/Admin(config-chaingroup)# no cert MYCERTS_2.PEM
```

Configuring a Group of Certificates for Authentication

On the ACE, you can implement a group of ten SSL certificates that are trusted as certificate signers by creating an authentication group. After creating the authentication group and assigning its certificates, you can assign the authentication group to a service in an SSL termination configuration to enable client authentication. For information on client authentication, see the "Enabling Client Authentication" section in Chapter 3, Configuring SSL Termination.

You can also assign the group to a service in an SSL initiation configuration to allow the ACE to authenticate the server certificate with the group certificates. For information on server authentication, see the "Configuring an Authentication Group for Server Authentication" section in Chapter 4, Configuring SSL Initiation.

To create an authentication group and access authgroup configuration mode, use the **crypto authgroup** command in configuration mode. The syntax of this command is as follows:

crypto authgroup *group_name*

The *group_name* argument is the name of the certificate authentication group. Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters.

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

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

After you create an authentication group, you access authgroup configuration mode, where you add the required certificate files to the group.

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

To delete an existing authentication group, enter:

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

To add certificate files to the authentication group, use the **cert** command in authgroup configuration mode. You can configure an authentication group with up to ten certificates.

The syntax of this command is as follows:

cert *cert_filename*

The *cert_filename* argument is the name of an existing certificate file stored on the ACE. Enter an unquoted alphanumeric string with a maximum of 40 characters.

Note When you make a change to an authgroup, the change takes effect immediately.

To display a list of existing certificate files, use the **show crypto files** command (see the "Displaying Certificate Information" section in Chapter 6, Displaying SSL Information and Statistics). It is not necessary to add the certificates in any type of hierarchal order because the device verifying the certificates determines the correct order.

For example, to add the certificate files MYCERTS.PEM and MYCERTS_2.PEM to the authentication group, enter:

```
host1/Admin(config-authgroup)# cert MYCERTS.PEM
host1/Admin(config-authgroup)# cert MYCERTS_2.PEM
```

To remove a certificate file from the authentication group, enter:

```
host1/Admin(config-authgroup)# no cert MYCERTS_2.PEM
```

CHAPTER 3

Configuring SSL Termination

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

- SSL Termination Overview
- ACE SSL Termination Configuration Prerequisites
- SSL Termination Configuration Quick Start
- Creating and Defining an SSL Parameter Map
- Enabling SSL Rehandshake for All VIPs in a Context
- Creating and Defining an SSL Proxy Service
- Configuring a DNS Client
- Configuring SSL URL Rewrite and HTTP Header Insertion
- Creating a Layer 3 and Layer 4 Class Map for SSL Termination
- Creating a Layer 3 and Layer 4 Policy Map for SSL Termination
- Applying the Policy Map to the VLANs
- 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.

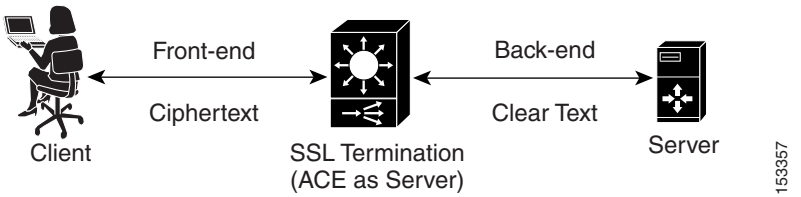
SSL Termination Overview

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



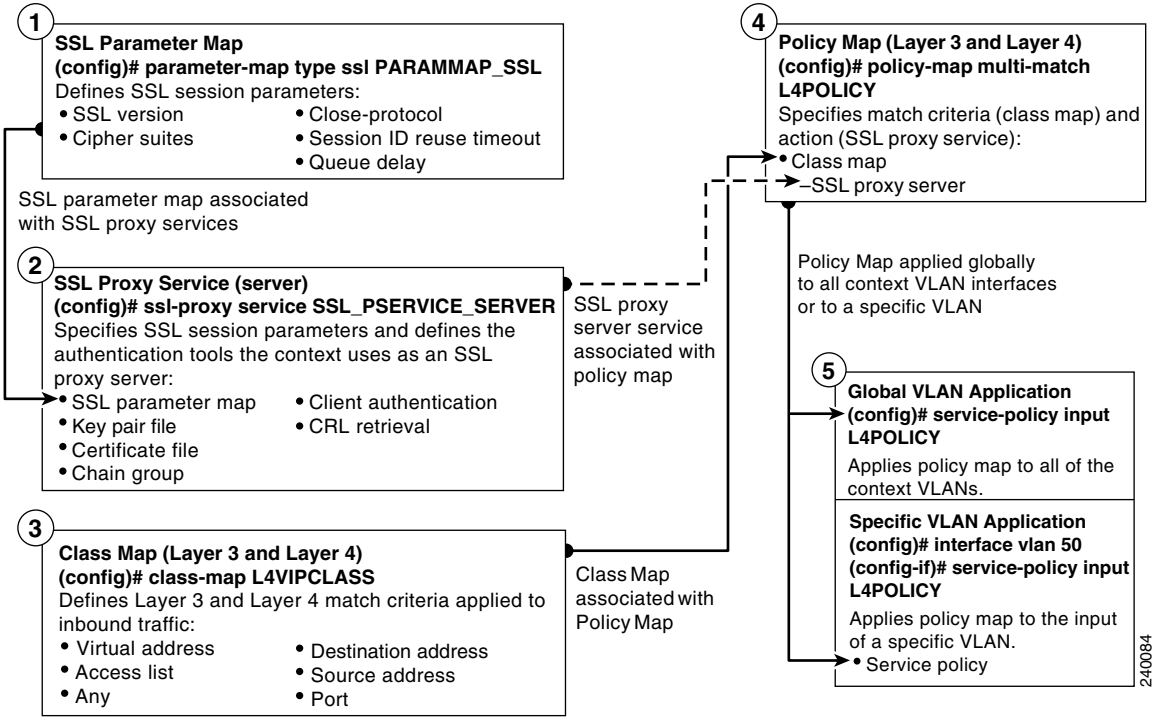
The ACE uses parameter maps, SSL proxy services, and class maps to build the policy maps that determine the flow of information between the client, the ACE, and the server. SSL termination is a Layer 3 and Layer 4 application because it is based on the destination IP addresses of the inbound traffic 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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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 *Cisco Application Control Engine Module Virtualization Configuration Guide*.

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)#`

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

Task and Command Example

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 192.168.10.24 tcp any`
`host1/Admin(config-cmap)# exit`
`host1/Admin(config)#`

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`

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

Task and Command Example
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 descrbes 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.

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

SSL Session Parameter	Description	Default Value/Behavior
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
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

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

SSL Session Parameter	Description	Default Value/Behavior
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

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

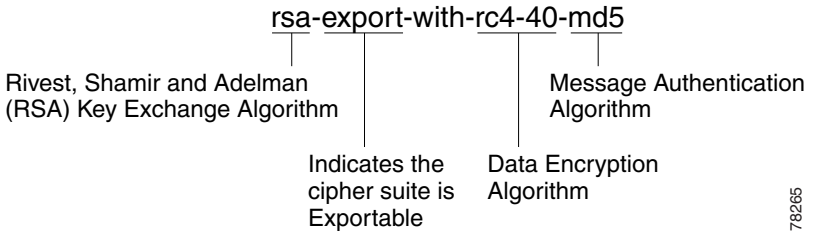
The SSL protocol supports a variety of different cryptographic algorithms, or ciphers, for use in operations such as the following:

- 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

Table 3-3 SSL Cipher Suites Supported by the ACE (continued)

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
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 example instructs 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:

```
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 tlsl
```

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 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 Signature Verification on a CRL

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. Additional latency may occur when you configure CRL retrieval (see the "Using CRLs During Client Authentication" section).

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 module.
- 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 module 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 Signature Verification on a CRL

You can configure signature verification on a Certificate Revocation List (CRL) to determine that it is from a trusted certificate authority by using the **crypto crlparams** command in Exec command mode. The syntax of this command is as follows:

crypto crlparams *crl_name* **cacert** *ca_cert_filename*

The arguments are as follows:

- *crl_name*—Name of an existing CRL.
- *ca_cert_filename*—Name of the CA certificate file used for signature verification.

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
```

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 HTTP Header Insertion of SSL Session Parameters
- Configuring HTTP Header Insertion of SSL Server Certificate Information
- Configuring HTTP Header Insertion of SSL Client Certificate Information
- Associating an Action List with a Layer 7 HTTP Load-balancing Policy Map
- Example Configurations Containing HTTP Header Insertion

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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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.
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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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 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 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

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

Session Field	Description
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
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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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 192.168.1.1 255.255.255.0
  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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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 192.168.1.1 255.255.255.0
  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 *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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.

```
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 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
```

CHAPTER 4

Configuring SSL Initiation

This chapter describes how to configure a context on the Cisco Application Control Engine (ACE) module as an SSL client for SSL initiation.

This chapter contains the following major sections:

Note To verify that the SSL connection from a server to the ACE was properly initiated, you can monitor the handshake counters in the **show stats crypto client** 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.

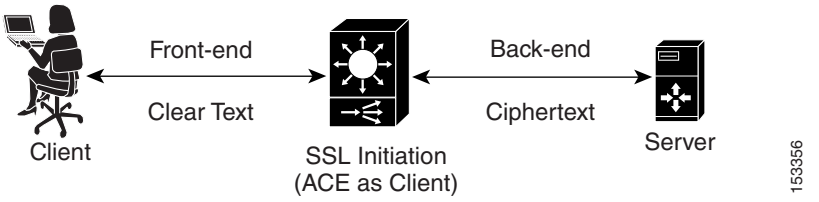
SSL Initiation Overview

SSL initiation occurs when an ACE, acting as an SSL proxy client, initiates and maintains an SSL connection between itself and an SSL server. In this particular application, the ACE receives clear text from an HTTP client, and encrypts and transmits the data as ciphertext to the SSL server. On the reverse side, the ACE decrypts the ciphertext that it receives from the SSL server and sends the data to the client as clear text.

Figure 4-1 shows the following network connections in which the ACE initiates the SSL connection with the SSL server:

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

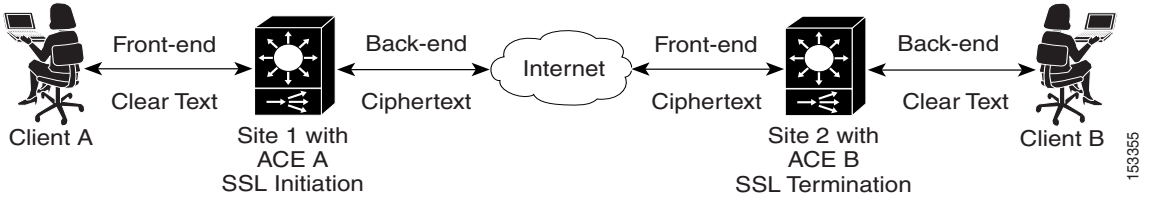
Figure 4-1 SSL Initiation with an SSL Server



SSL initiation allows you to send clear text between devices within a site for maximum speed, while sending ciphertext through the Internet between sites or to an SSL server for maximum security. For each SSL server or ACE (acting as an SSL proxy server) to which you want to establish an SSL connection from a clear text connection, you must configure an SSL initiation policy service on the ACE that maps to that SSL server or other ACE.

Figure 4-2 shows an SSL initiation flow with another ACE configured for SSL termination. In this case, ACE B acts as a virtual front-end SSL server.

Figure 4-2 SSL Initiation with a Second ACE Running SSL Termination

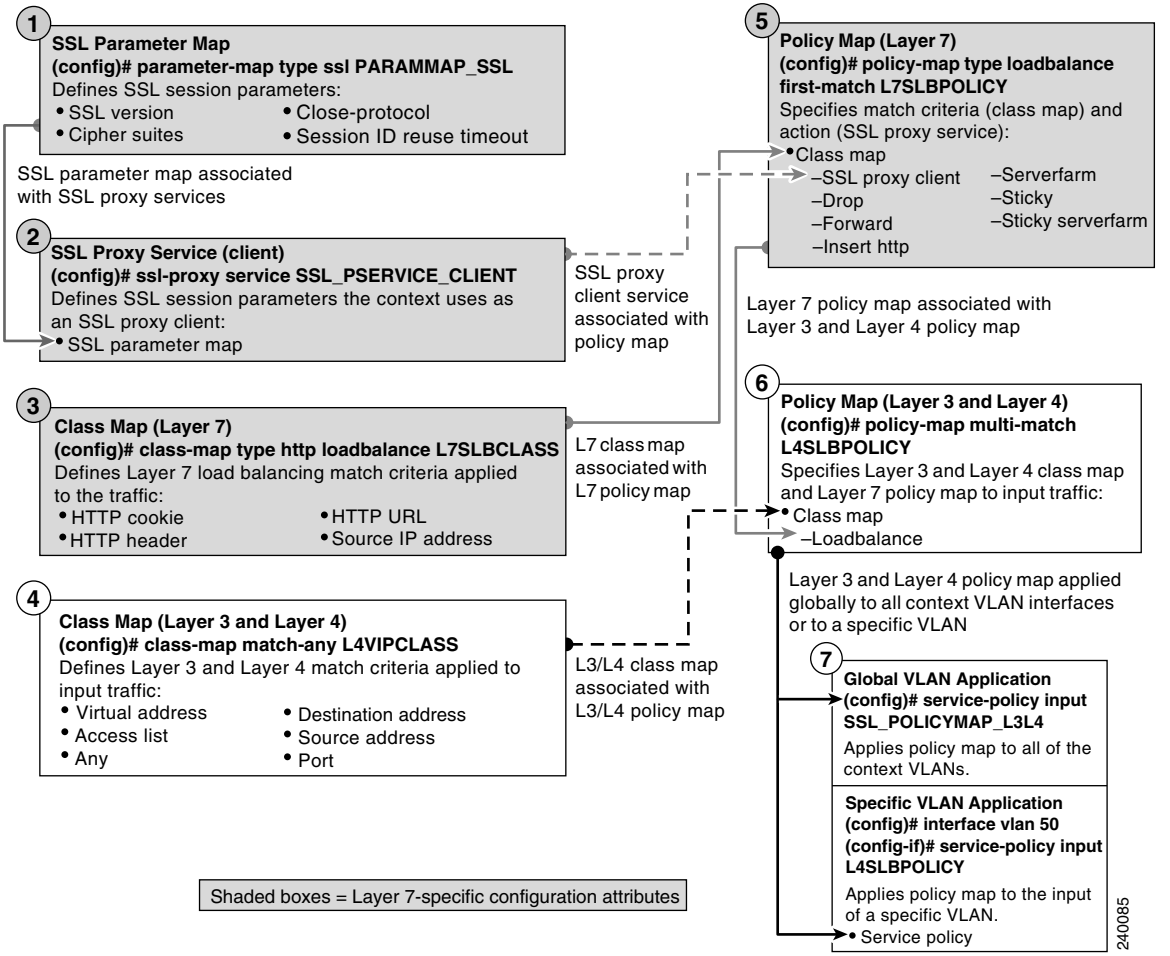


The ACE uses a combination of parameter maps, SSL proxy services, and class maps to build the policy maps that determine the flow of information among the client, the ACE, and the SSL server. For SSL initiation, you configure the ACE so that it is recognized as an SSL client by an SSL server. To accomplish this, you configure the following policy map types:

- Layer 7 policy map—This policy map contains an association with a Layer 7 class map and an SSL proxy client service. The class map acts as a traffic filter and looks for traffic that matches the server load-balancing (SLB) criteria that you specify. For SSL initiation, the match criteria is in the form of HTTP load-balancing attributes, such as an HTTP cookie or URL. The SSL proxy client service defines the SSL parameters that the ACE uses during the handshake and subsequent SSL session.
- Layer 3 and Layer 4 policy map—You associate the Layer 7 policy map with a Layer 3 and Layer 4 policy map. The ACE applies the Layer 3 and Layer 4 policy map to the context traffic first to determine if the traffic contains specific Layer 3 and Layer 4 match criteria, such as a particular destination, source, or virtual IP address. You specify the match criteria in the Layer 3 and Layer 4 class map that you create and associate with this policy map. When a match is found, the ACE applies the associated Layer 7 policy map to the traffic.

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

Figure 4-3 Basic SSL Initiation Configuration Flow Diagram



ACE SSL Initiation Configuration Prerequisites

Before configuring your ACE for SSL operation, you must first configure it for server load balancing (SLB). During the SLB 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, modify the existing SLB class maps and policy maps with the SSL configuration requirements described in this guide for SSL initiation.

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

SSL Initiation Configuration Quick Start

Figure 4-1 provides a quick overview of the steps required to configure the ACE for SSL initiation. 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 4-1.

Note The following quick start does not include the procedures for creating a parameter map as shown in Figure 4-3. The ACE uses the default parameter map settings as described in Table 4-2.

Table 4-1 SSL Initiation 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 *Cisco Application Control Engine Module Virtualization Configuration Guide*.

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

3. Create an SSL proxy client service to associate with the Layer 7 policy map. For the purposes of this Quick Start, you do not define any parameters of the proxy client service; associating this generic proxy client service with the policy map is all that is required to configure the ACE to perform as an SSL client.
`host1/Admin(config)# ssl-proxy service SSL_PSERVICE_CLIENT`
`host1/Admin(config-ssl-proxy)# exit`

4. Create a Layer 7 class map and configure it with the required load-balancing match criteria.
`host1/Admin(config)# class-map type http loadbalance L7SLBCLASS`
`host1/Admin(config-cmap-http-lb)# match url XYZ.ORG`
`host1/Admin(config-cmap-http-lb)# exit`
`host1/Admin(config)#`

5. Create a Layer 3 and Layer 4 class map and configure it with the required input traffic match criteria.
`host1/Admin(config)# class-map match-any L4VIPCLASS`
`host1/Admin(config-cmap)# match virtual-address 192.168.12.2 255.255.255.0`
`host1/Admin(config-cmap)# exit`
`host1/Admin(config)#`

Table 4-1 SSL Initiation Configuration Quick Start (continued)

Task and Command Example

6. Create a Layer 7 policy map and associate the Layer 7 class map created in Step 4 with it.
`host1/Admin(config)# policy-map type loadbalance first-match L7SLBPOLICY`
`host1/Admin(config-pmap-lb)# class L7SLBCLASS`
`host1/Admin(config-pmap-lb-c)#`

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

8. Create a Layer 3 and Layer 4 policy map and associate the Layer 3 and Layer 4 class map created in Step 5 with it.
`host1/Admin(config)# policy-map multi-match L4SLBPOLICY`
`host1/Admin(config-pmap)# class CLASSMAP_L3`
`host1/Admin(config-pmap-c)#`

9. Associate the load-balancing Layer 7 policy map created in Step 6 with the Layer 3 and Layer 4 policy map.
`host1/Admin(config-pmap-c)# loadbalance L7SLBPOLICY`
`host1/Admin(config-pmap-c)# exit`
`host1/Admin(config-pmap)# exit`
`host1/Admin(config)#`

Table 4-1 SSL Initiation Configuration Quick Start (continued)

Task and Command Example

10. Apply the Layer 3 and Layer 4 policy map to the input traffic of the desired interface as follows:
 	
Apply the policy map globally to all VLANs in the context.
 	
`host1/Admin(config)# service-policy input L4SLBPOLICY`
 	
Apply the policy map to a specific VLAN within the context.
 	
`host1/Admin(config)# interface vlan 50`
`host1/Admin(config-if)# service-policy input L4SLBPOLICY`

11. Display the running configuration to verify that the information that you just added is configured properly. `host1/Admin(config-if)# do show running-config`
12. (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 4-2 describes each SSL session parameter with its default value.

Table 4-2 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 4-3 for a list of available cipher suites the ACE supports)	The ACE supports all of the available cipher suites
Authentication-failure ignore	Enables the ACE to ignore expired or invalid server certificates and to continue setting up the back-end connection in an SSL initiation configuration.	The ACE terminates the SSL handshake when a 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 the client/server when closing a session but has no expectation of receiving one back from the client/server
Purpose-check disabled	When this command is configured, this parameter disables the ACE from performing purpose checking on certificates during authentication.	Enabled

Table 4-2 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
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 4-2 to the proxy service.

The parameter map SSL configuration mode includes the **queue-delay timeout** command. The queue delay applies only to encrypted data that the ACE sends to the client. For this reason, this timer has no effect on SSL initiation connections handled by the ACE.

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 no spaces and 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 the 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 4-2.

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
- Ignoring Expired or Invalid Server Certificates
- 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 Versions
- Configuring the SSL Session Cache Timeout
- Rejecting Expired CRL Server 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

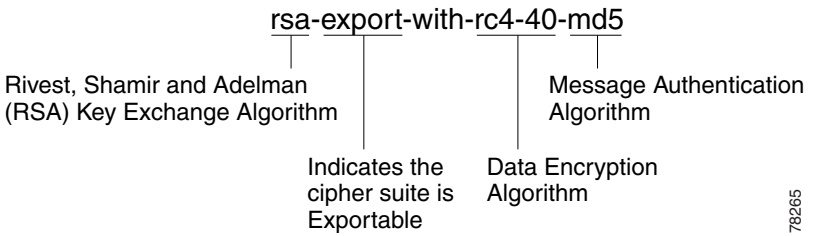
The SSL protocol supports a variety of different cryptographic algorithms, or ciphers, for use in operations such as the following:

- 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 4-4, a cipher suite consists of the following three algorithms: key exchange algorithm, data encryption algorithm, and message authentication (hash) algorithm.

Figure 4-4 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.

You can define each of the cipher suites that you want the ACE to support during a secure session by using 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 4-3. 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 4-3 lists the cipher suites that the ACE supports. Enter one of the supported cipher suites from the table.

- **priority**—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 a value of 1 to 10. The default priority value is 1.

For example, to add the cipher suite rsa_with_aes_128_cbc_sha and assign it 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 4-3 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 supports all of the ciphers suites listed in Table 4-3, 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 4-3 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_EXPORT_WITH_RC4_40_MD5	Yes	RSA certificate	RSA key exchange
RSA_EXPORT_WITH_DES40_CBC_SHA	Yes	RSA certificate	RSA key exchange
RSA_EXPORT1024_WITH_RC4_56_MD5	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
RSA_WITH_AES_128_CBC_SHA	No	RSA certificate	RSA key exchange
RSA_WITH_AES_256_CBC_SHA	No	RSA certificate	RSA key exchange

Ignoring Expired or Invalid Server Certificates

You can enable the ACE to ignore expired or invalid server certificates and to continue setting up the back-end connection in an SSL initiation configuration by using the **authentication-failure ignore** command in parameter map SSL configuration mode. This command allows the ACE to ignore the following nonfatal errors with respect to server certificates:

- Certificate not yet valid
- Certificate has expired
- Unable to get issuer certificate
- Certificate revoked

The syntax of this command is as follows:

authentication-failure ignore

For example, to ignore expired or invalid server certificates, enter:

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

```
host1/Admin(config-parammap-ssl)# authentication-failure ignore
```

To return to the default setting of disabled, use the **no** form of the command:

```
host1/Admin(config-parammap-ssl)# no authentication-failure ignore
```

Configuring the ACE to Ignore Authentication Failures Due to CDP Errors

By default, when you configure the **crl best-effort** command for server 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 ssl parameter-map configuration mode.

The syntax for 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 the client/server when closing a session with no expectation of receiving one back from the client/server.
- **none**—Specifies that the ACE sends a close notify alert message to the client/server when closing a session but has no expectation of receiving one back from the client/server.

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

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

To configure the **close-protocol** command to the default setting to send a close notify alert message to the client/server, enter:

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

Disabling Purpose Checking on the Certificates

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

- The server 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

The SSL session rehandshake enables the ACE to send the SSL HelloRequest message to a client to restart SSL handshake negotiation. The rehandshake is useful when you want to ensure security by reestablishing the SSL session.

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

The syntax of this command is:

rehandshake enable

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

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

To disable the rehandshake function, enter:

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

To display the status of the rehandshake enable 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 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 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.

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 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 Server Certificates

When you configure Certificate Revocation Lists (CRLs) on the ACE for server authentication, as described in the "Using CRLs During Server 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 server certificates using the CRL.

To configure the ACE to consider a server certificate as revoked 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 of not considering a server certificate as revoked after the CRL in use has expired, enter:

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

Creating and Defining an SSL Proxy Service

The SSL proxy service defines the SSL parameter map that the ACE uses during the SSL handshake. For SSL initiation, you configure the ACE with an SSL proxy *client* service because the ACE acts as an SSL client.

Note You do not need to import or associate keys and certificates in an SSL initiation configuration.

You can create an SSL proxy client 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 client service. Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters.

For example, to create the SSL proxy client service PSERVICE_CLIENT, enter:

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

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

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

To delete an existing SSL proxy client service, enter:

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

This section contains the following topics:

Associating an SSL Parameter Map with the SSL Proxy Client Service

You can associate an SSL parameter map with the SSL proxy client 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 no spaces and 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_CLIENT
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
```

Configuring an Authentication Group for Server Authentication

By default, server authentication is always enabled in an SSL initiation configuration. The server must send a certificate to the ACE. The ACE authenticates the certificate by verifying that it is a server certificate and that it has not expired. However, the ACE does not check that the certificate has been

signed by an approved CA. If the server certificate has expired, the ACE rejects the backend connection by sending a reset (RST) to the server. Otherwise, the ACE sets up the SSL connection with the server normally.

You can override this behavior by using the **authentication-failure ignore** command in parameter map SSL configuration mode. For details about this command, see the "Ignoring Expired or Invalid Server Certificates" section.

An authentication group consists of certificates that are trusted as certificate signers (see the "Configuring a Group of Certificates for Authentication" section in Chapter 2, Managing Certificates and Keys). When you assign an authentication group to an SSL-proxy server in an SSL initiation configuration, the ACE checks the server certificate with the certificates in the group, which includes checking the issuer and the signature of the server certificate.

To use an authentication group for server authentication on this SSL-proxy service, use the **authgroup** command in SSL proxy configuration mode. The syntax of the **authgroup** command is as follows:

authgroup *group_name*

The *group_name* argument is the name of an existing certificate authentication group. Enter an unquoted alphanumeric string with no spaces and a maximum of 64 characters.

Note When you enable server authentication, a significant performance decrease of the ACE may occur.

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 Server Authentication

By default, the ACE does not use certificate revocation lists (CRLs) during server 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 server 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 module.
- You can manually configure the download location of 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 server 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 Server 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 server 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 server 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 server authentication on an SSL proxy service, enter the following command:

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

To scan the client certificate for CRL information, enter:

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

When the ACE accepts a server certificate in the downloaded CRL database, a successful SSL connection to an SSL real server increments the following **show stats crypto client** counters:

- Total SSL server authentications
- SSL static CRL lookups

When the ACE accepts a server certificate on a best-effort-CRL-enabled connection and the certificate is not found in the downloaded CRL database, a successful SSL connection to an SSL real server increments the following **show stats crypto client** counters:

- Total SSL server authentications
- SSL best effort CRL lookups

After the certificate is validated and cached in the ACE, subsequent SSL connections without session reuse to the same SSL server increments the following **show stats crypto client** counters:

- Total SSL server authentications
- SSL best effort CRL lookups
- SSL CRL lookup cache hits
- SSL authentication cache hits

If a valid non-expired CRL is cached in the ACE, no CRL lookups are performed and the following **show stats crypto client** counters will not increment together by the same connection:

- SSL best effort CRL lookups
- SSL CRL lookup cache hits

When the SSL connection to the SSL real server fails because of a revoked server certificate, the following **show stats crypto client** counters increment:

- SSL alert CERTIFICATE_REVOKED sent
- Total SSL server authentications
- Failed SSL server authentications
- SSL best effort CRL lookups or SSL static CRL lookups

To disable the use of a downloaded CRL during server authentication, enter the following command:

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

To disable the use of server certificates for CRL information during server authentication, enter the following command:

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

Configuring the Download Location for CRLs

You can configure the load location that the ACE uses to download the CRL on the SSL proxy service for server 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 in Chapter 3, "Configuring SSL Termination").
- 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 server 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 server authentication (see the "Using CRLs During Server 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; the CDP. 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
```

Configuring Signature Verification on a CRL

You can configure signature verification on a Certificate Revocation List (CRL) to determine that it is from a trusted certificate authority by using the **crypto crlparams** command in Exec command mode. The syntax of this command is as follows:

crypto crlparams *crl_name* **cacert** *ca_cert_filename*

The arguments are as follows:

- *crl_name*—Name of an existing CRL.
- *ca_cert_filename*—Name of the CA certificate file used for signature verification.

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
```

Creating a Layer 7 Class Map for SSL Initiation

The Layer 7 class map that you associate with a policy map acts as a filter for traffic that matches the server load balancing (SLB) criteria that you specify. For SSL initiation, the match criteria is in the form of the following HTTP load-balancing attributes:

- Cookie
- HTTP header
- URL
- Source IP address

You can create a Layer 7 class map by using the **class-map type http loadbalance** command in configuration mode. For details on configuring a Layer 7 class map, see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

Creating a Layer 7 Policy Map for SSL Initiation

A Layer 7 policy map enables server load balancing on the ACE. This policy map contains an association with a Layer 7 class map and an SSL proxy client service. To use a Layer 7 SLB policy map, you first create the policy map and then define the **match** statements and policy actions. Because Layer 7 policy maps are child policies, you must associate a Layer 7 policy map with the appropriate Layer 3 and Layer 4 policy map to provide an entry point for Layer 7 SLB traffic classification. You cannot directly apply a Layer 7 policy map to an interface; you can apply only a Layer 3 and Layer 4 policy map to an interface or globally to all interfaces in a context.

This section contains the following topics:

Creating a Layer 7 Policy Map

You can create a Layer 7 SLB policy map by using the **policy-map** command in configuration mode.

The syntax of this command is as follows:

policy-map type loadbalance first-match *map_name*

The keywords and arguments are as follows:

- **type loadbalance—**Specifies a load-balancing policy map.
- **first-match**—Defines the execution for the Layer 7 load-balancing policy map. The ACE executes only the action specified against the first-matching classification.
- *map_name*—Identifier assigned 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 L7SLBPOLICY, enter:

```
host1/Admin(config)# policy-map type loadbalance first-match
L7SLBPOLICY
```

After you create a Layer 7 policy map, the CLI enters policy map load-balancing configuration mode.

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

To delete an existing policy map, enter:

```
host1/Admin(config)# no policy-map L7SLBPOLICY
```

Associating a Layer 7 Class Map with the Layer 7 Policy Map

You can associate a class map with the policy map by using the **class** command in policy map load-balancing configuration mode.

The syntax of this command is as follows:

class {*name1* | **class-default**} [**insert-before** *name2*]

The keywords, arguments, and options are as follows:

- *name1*—Name of a previously defined traffic class, configured with the **class-map** command, to associate traffic with the traffic policy. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **class-default**—Specifies the reserved, well-known class map created by the ACE. You cannot delete or modify this class. All traffic that fails to meet the other matching criteria in the named class map belongs to the default traffic class. If no specified classification matches the traffic, then the ACE performs the action specified using the **class class-default** command. The **class-default** class map has an implicit **match any** statement in it that enables it to match all traffic.
- **insert-before** *name2*—(Optional) Places the current class map ahead of an existing class map or match statement specified by the *name2* argument in the policy-map configuration. The ACE does not save the sequence reordering as part of the configuration.

For example, to associate the class map L7SLBCLASS with the policy map, enter:

```
host1/Admin(config-pmap-lb)# class L7SLBCLASS
```

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

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

The following example shows how to use the **insert-before** option to define the position of a class map in the policy map:

```
host1/Admin(config-pmap-lb)# class L7SLBCLASS insert-before HTTP_CLASS
host1/Admin(config-pmap-lb-c)#
```

The following example shows how to use the **class class-default** command:

```
host1/Admin(config-pmap-lb)# class class-default
```

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

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

```
(config-pmap-lb)# no class L7SLBCLASS
```

Specifying Layer 7 SLB Policy Actions

After you associate a Layer 7 SLB class map with a Layer 7 SLB policy map or specify inline **match** commands, you need to specify one or more of the following actions that the ACE should take when network traffic matches a class map or inline **match** command:

- Discard requests
- Forward Requests without load balancing
- Enable HTTP header information
- Enable load balancing to a server farm
- Configure a sticky server farm
- Specify the IP differentiated services code point of packets
- Associate an SSL proxy service

This section describes the process of associating an SSL proxy service with the policy map. For details on configuring additional policy actions, see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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

The syntax of this command is as follows:

ssl-proxy client *name*

The *name* argument is the identifier of an existing SSL proxy client service. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to associate the SSL client proxy service PSERVICE_CLIENT with the class map, enter:

```
host1/Admin(config)# policy-map type loadbalance first-match
L7SLBPOLICY
host1/Admin(config-pmap-lb)# class L7SLBCLASS
```

```
host1/Admin(config-pmap-lb-c)# ssl-proxy client PSERVICE_CLIENT
```

To remove the association of the SSL client proxy service to the class map, enter:

```
host1/Admin(config-pmap-lb-c)# no ssl-proxy client PSERVICE_CLIENT
```

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

The Layer 3 and Layer 4 class map that you associate with a Layer 3 and Layer 4 policy map acts as a filter for traffic that matches the criteria that you specify. For SSL initiation, 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 the configuration mode. For details on creating and configuring a Layer 3 and Layer 4 class map, see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

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

The Layer 3 and Layer 4 policy map that you create for SSL initiation contains an association with the Layer 7 policy map that the ACE uses for load balancing. Because you can apply only a Layer 3 and Layer 4 policy map directly to a context interface, you need to associate the Layer 7 policy map with the Layer 3 and Layer 4 policy map.

This section contains the following topics:

Creating a Layer 3 and Layer 4 Policy Map

You can create a Layer 3 and Layer 4 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 L4SLBPOLICY, enter:

```
host1/Admin(config)# policy-map multi-match L4SLBPOLICY
```

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 L4SLBPOLICY
```

For information on associating an SSL class map with the policy map, see the "Associating a Layer 7 Class Map with the Layer 7 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 L4SLBCLASS with the policy map, enter:

```
host1/Admin(config)# policy-map multi-match L4SLBPOLICY
host1/Admin(config-pmap)# class L4SLBCLASS
```

After you associate a class map with the policy map, the CLI enters policy map class 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 L4SLBCLASS
```

Associating a Layer 7 Policy Map with the Class Map

You can associate a Layer 7 policy map with the Layer 3 and Layer 4 class map by using the **loadbalance** command in policy map class configuration mode. This association nests the Layer 7 policy map within the Layer 3 and Layer 4 policy map that the ACE applies directly to the traffic.

The syntax of this command is as follows:

loadbalance policy *policymap*

The **policy** *policymap* keyword and argument specify the name of an existing Layer 7 policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to associate the Layer 7 policy map L7SLBPOLICY with the class map, enter:

```
host1/Admin(config)# policy-map multi-match L4SLBPOLICY
host1/Admin(config-pmap)# class L4SLBCLASS
host1/Admin(config-pmap-c)# loadbalance policy L7SLBPOLICY
```

To remove the association of the Layer 7 policy map with the class map, enter:

```
host1/Admin(config-pmap-c)# no loadbalance policy L7SLBPOLICY
```

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 of the 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 L4SLBPOLICY to all of the context VLANs, enter:

```
host1/Admin(config)# service-policy input L4SLBPOLICY
```

To globally remove a policy map from all VLANs, enter:

```
host1/Admin(config)# no service-policy input L4SLBPOLICY
```

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 into 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 L4SLBPOLICY to VLAN 10, enter:

```
host1/Admin(config)# interface vlan 10
host1/Admin(config-if)# service-policy input L4SLBPOLICY
```

To remove the policy from the interface, enter:

```
host1/Admin(config-if)# no service-policy input L4SLBPOLICY
```

Example of an SSL Initiation Configuration

The following example illustrates a running configuration of the ACE acting as a SSL proxy client, initiating and maintaining an SSL connection between itself and an SSL server. The ACE receives clear text from an HTTP client, and then encrypts and transmits the data as cipher text to the SSL server. On the reverse

side, the ACE decrypts the cipher text that it receives from the SSL server and sends the data to the client as clear text. The SSL initiation configuration appears in bold in the example.

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

probe http GEN-HTTP
  port 80
  interval 50
  faildetect 5
  expect status 200 200

serverfarm host SFARM1
  description SERVER FARM 1 FOR SSL INITIATION
  probe GEN_HTTP
  rserver SERVER1 443
    inservice
  rserver SERVER2 443
    inservice
  rserver SERVER3 443
    inservice
  rserver SERVER4 443
    inservice

serverfarm host SFARM2
  description SERVER FARM 2 FOR SSL TERMINATION
  probe GEN_HTTP
  rserver SERVER5 443
    inservice
  rserver SERVER6 443
    inservice
  rserver SERVER7 443
    inservice
  rserver SERVER8 443
    inservice

parameter-map type http PARAMMAP_HTTP
  server-conn reuse
  case-insensitive
  persistence-rebalance
parameter-map type ssl PARAMMAP_SSL_INITIATION
  cipher RSA_WITH_RC4_128_MD5
  cipher RSA_WITH_RC4_128_SHA
  cipher RSA_WITH_DES_CBC_SHA
  cipher RSA_WITH_3DES_EDE_CBC_SHA
  cipher RSA_WITH_AES_128_CBC_SHA
  cipher RSA_WITH_AES_256_CBC_SHA
  cipher RSA_EXPORT_WITH_RC4_40_MD5
```

```
  cipher RSA_EXPORT1024_WITH_RC4_56_MD5
  cipher RSA_EXPORT_WITH_DES40_CBC_SHA
  cipher RSA_EXPORT1024_WITH_DES_CBC_SHA
  cipher RSA_EXPORT1024_WITH_RC4_56_SHA
  version all
parameter-map type connection TCP_PARAM
  syn-data drop
  exceed-mss allow

ssl-proxy service SSL_PSRVICE_CLIENT
  ssl advanced-options PARAMMAP_SSL_INITIATION

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-INIT_CLASS
  description SSL Initiation VIP
  2 match virtual-address 192.168.130.12 tcp eq www
policy-map type loadbalance first-match L7_SSL-INIT_POLICY
  class L7_SERVER_CLASS
    serverfarm SFARM1
    insert-http SRC_IP header-value "%is"
    insert-http I_AM header-value "SSL_INIT"
    insert-http SRC_Port header-value "%ps"
    insert-http DEST_IP header-value "%id"
    insert-http DEST_Port header-value "%pd"
    ssl-proxy client SSL_PSERVICE_CLIENT
  class L7_SLB-HTTP_CLASS
    serverfarm SFARM2
    insert-http SRC_IP header-value "%is"
    insert-http I_AM header-value "SSL_INIT"
    insert-http DEST_Port header-value "%pd"
    insert-http DEST_IP header-value "%id"
    insert-http SRC_Port header-value "%ps"
    ssl-proxy client SSL_PSERVICE_CLIENT
policy-map multi-match L4_SSL-VIP_POLICY
  class L4_SSL-INIT_CLASS
    loadbalance vip inservice
    loadbalance policy L7_SSL-INIT_POLICY
    loadbalance vip icmp-reply active
    appl-parameter http advanced-options PARAMMAP_HTTP
    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
```

CHAPTER 5

Configuring End-to-End SSL

This chapter describes how to configure a Cisco Application Control Engine (ACE) module to provide end-to-end SSL connectivity. This process involves combining SSL termination (front end) with SSL initiation (back end) to provide a secure link between the client, the ACE, and the server. All data is encrypted and sent as ciphertext among the three devices.

This chapter contains the following major sections:

End-to-End SSL Overview

End-to-end SSL refers to the ACE's establishing and maintaining SSL connections between the client at one end of the connection and the server at the other end of the connection. When you configure the ACE for end-to-end SSL, the ACE performs the following functions:

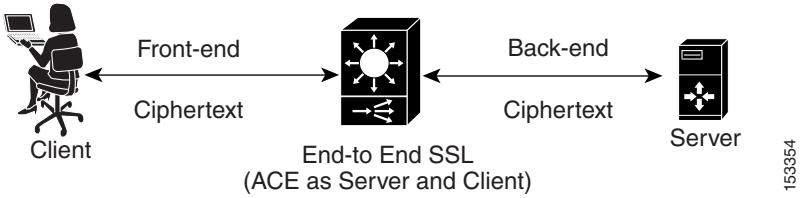
- Terminates an SSL session with the client (front-end connection)
- Initiates an SSL session with the server (back-end connection)
- Load balances the back-end content

End-to-end SSL combines the configurations that you use to configure the ACE for SSL termination and SSL initiation. For end-to-end SSL, you must create the following policy map types:

- Layer 7 policy map—Directs the back-end flow of traffic between the ACE and the server.
- Layer 3 and Layer 4 policy map—Performs the following functions:
  - Directs the front-end flow of traffic between the client and the ACE.
  - Applies the associated Layer 7 policy map to the traffic that meets the criteria of the Layer 3 and Layer 4 policy map.

Figure 5-1 shows an end-to-end SSL application in which the ACE terminates an SSL connection with an SSL client and initiates an SSL connection with an SSL server.

Figure 5-1 End-to-End SSL

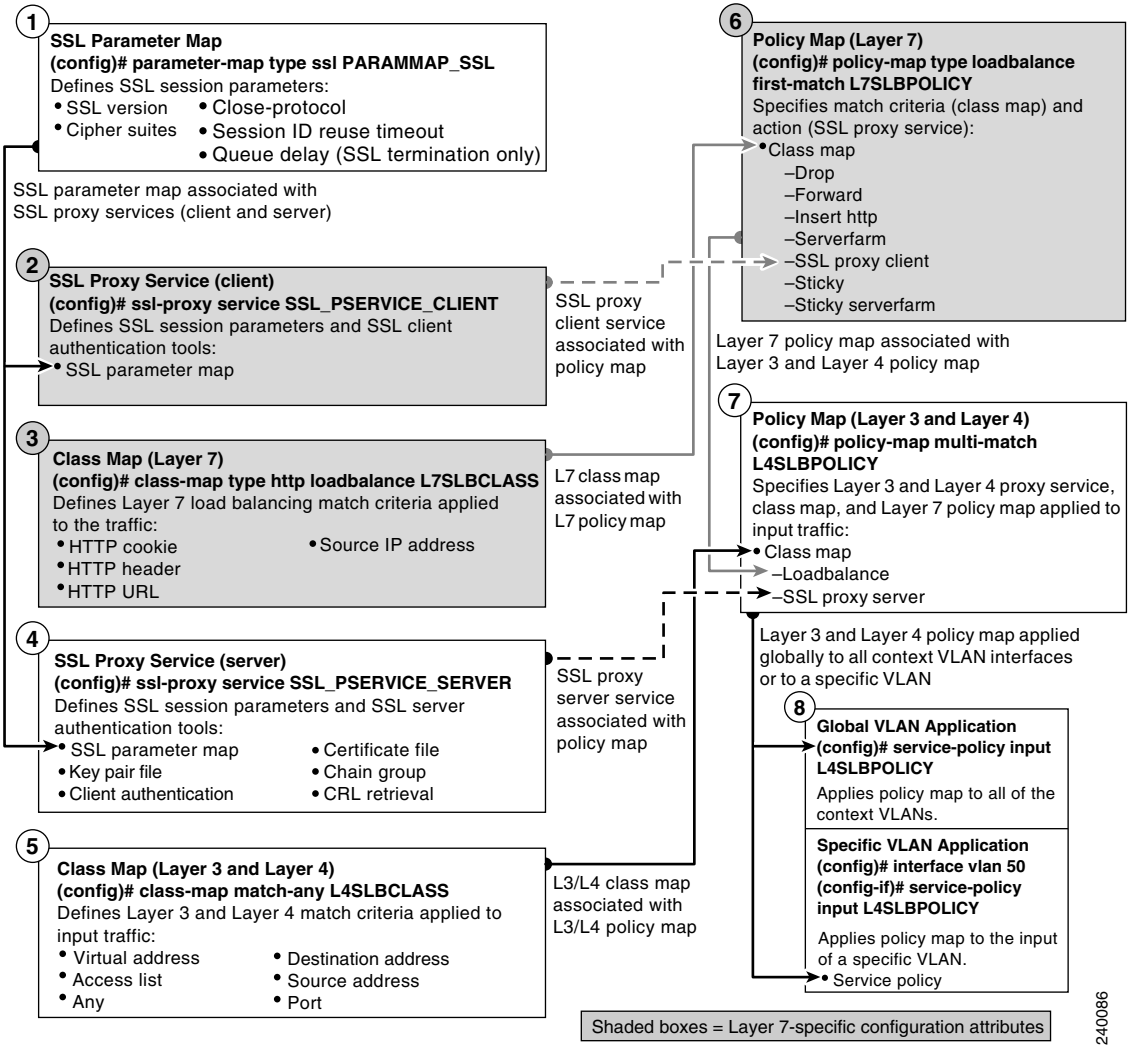


The ACE uses a combination of parameter maps, SSL proxy services, and class maps to build the policy maps that determine the flow of information between the client, the ACE, and the SSL server.

Figure 5-2 provides a basic overview of the process required to build the Layer 7 load-balancing policy map and associate it with the Layer 3 and Layer 4 policy map to create an end-to-end SSL configuration. To allow you to easily discern between the Layer 7 and Layer 3 and Layer 4 configuration attributes, the Layer 7 attributes are shaded gray.

In the final step of the process, you apply the Layer 3 and Layer 4 policy map to the input traffic of the context. The figure also shows how the various components of the policy map configurations are associated with each other.

Figure 5-2 Basic End-to-End SSL Configuration Flow Diagram



ACE End-to-End SSL Configuration Prerequisites

Before configuring your ACE for SSL operation, you must first configure it for server load balancing (SLB). During the SLB 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, modify the existing SLB class maps and policy maps with the SSL configuration requirements described in this guide for end-to end SSL.

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

Configuring End-to-End SSL

Table 5-1 provides an overview of the process required to configure the ACE for end-to-end SSL. Because end-to-end SSL combines the configuration processes of SSL termination and SSL initiation, the procedure provides links to the sections of this guide where the specified process is described in detail.

Table 5-1 End-to-End SSL Configuration Quick Start

Task
1. Configure the ACE for SSL initiation as described in Chapter 4, Configuring SSL Initiation. The SSL initiation configuration configures all of the back-end operation and a portion of the front-end operation. Do not apply the configuration to the VLANs at this time.
2. Create a parameter map for the ACE to use in the front-end operation as described in the "Creating and Defining an SSL Parameter Map" section of Chapter 3, Configuring SSL Termination. Skip this step if the ACE is to use the same parameter map that you created in Step 1 for the back-end operation.

Table 5-1 End-to-End SSL Configuration Quick Start (continued)

Task
3. Create an SSL proxy server service as described in the "Creating and Defining an SSL Proxy Service" section of Chapter 3, Configuring SSL Termination.
4. Associate the SSL proxy server service with the Layer 3 and Layer 4 policy map created in Step 1. For information on making this association, see the "Associating an SSL Proxy Server Service with the Policy Map" section of Chapter 3, Configuring SSL Termination.
5. Apply the Layer 3 and Layer 4 policy map to the VLANs as described in the "Applying the Policy Map to the VLANs" section of Chapter 3, Configuring SSL Termination.
6. (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`

Example of an End-to-End SSL Configuration

The following example illustrates an end-to-end SSL configuration, which combines front-end SSL and back-end SSL. The ACE receives encrypted text from an HTTP client, and also transmits the encrypted data as cipher text to the SSL server. On the reverse side, the ACE decrypts the cipher text that it receives from the SSL server and sends the data to the client as clear text. The SSL-specific configuration elements appear in bold in the example.

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

rserver host TEST4
  ip address 20.20.2.11
  inservice

serverfarm host TEST
  rserver TEST4
    inservice

parameter-map type ssl PM1
  session-cache timeout 300
  queue-delay timeout 1
```

```
ssl-proxy service SSL_CLIENT
  ssl advanced-options PM1

ssl-proxy service SSL_SERVER
  key KEY12.PEM
  cert CERT12.PEM
  ssl advanced-options PM1

class-map type http loadbalance match-any SSL
  2 match http url .*
class-map match-any SSL_C1
  2 match virtual-address 10.10.2.101 tcp eq https
  3 match virtual-address 10.10.2.101 tcp any

policy-map type loadbalance first-match SSL_BACK
  class SSL
    serverfarm TEST
    ssl-proxy client SSL_CLIENT

policy-map multi-match L7_1
  class SSL_C1
    loadbalance vip inservice
    loadbalance policy SSL_BACK
    loadbalance vip icmp-reply
    ssl-proxy server SSL_SERVER

interface vlan 210
  ip address 10.10.2.1 255.255.255.0
  service-policy input L7_1
  access-group input ACL
  no shutdown
interface vlan 220
  ip address 20.20.2.1 255.255.255.0
  no shutdown
interface vlan 226
  ip address 10.90.15.27 255.255.255.0
  no shutdown

ip route 0.0.0.0 0.0.0.0 10.90.15.1
```

CHAPTER 6

Displaying SSL Information and Statistics

This chapter describes how to use the available **show** commands to display SSL-related information, such as the certificate and key pair files loaded on the ACE. The **show** commands display information associated with the context from which you execute the command. Each command described in this chapter also includes an explanation of the command output.

While the **show** commands are Exec mode commands, you can execute a **show** command from any configuration mode by using the **do** command. The following examples show how to execute the **show running-config** command from either Exec mode or configuration mode.

From Exec mode, enter:

```
host1/Admin# show running-config
```

From configuration mode, enter:

```
host1/Admin(config)# do show running-config
```

This chapter contains the following major sections:

Displaying CSR Parameter Set Configurations

To display the CSR parameter set summary and detailed reports, use the **show crypto csr-params** command in Exec mode.

The syntax of this command is as follows:

show crypto csr-params {*params_set* | **all**}

The arguments and keywords are:

- *params_set*— argument is a specific CSR parameter set. Enter an unquoted alphanumeric string with a maximum of 64 characters. The ACE displays the detailed report for the specified CSR parameter set. The detailed report contains the distinguished name attributes of the CSR parameter set.
- To display the summary report that lists all the CSR parameter sets for the current context, enter the command without specifying a CSR parameter set.

For example, to display the CSR parameter set summary report, enter:

```
host1/Admin# show crypto csr-params all
```

The following example shows how to display the detailed report for the MYCSRCONFIG CSR parameter set:

```
host1/Admin# show crypto csr-params MTCSRCONFIG
```

Table 6-1 describes the fields in the **show crypto csr-params** command output.

Table 6-1 Field Descriptions for the show crypto csr-params config_name Command

Field	Description
Country-name	Country where the certificate owner resides.
State	State where the certificate owner resides.
Locality	Locality where the certificate owner resides.
Org-name	Name of the organization (certificate owner or subject).
Org-unit	Name of unit within the organization.
Common-name	Common-name (domain name or individual hostname of the SSL site).
Serial number	Serial number.
Email	E-mail address.

Displaying the List of Certificate and Key Pair Files

To display a list of all available certificate and key pair files, use the **show crypto files** command in Exec mode.

For example, to display the list of certificate and key pair files, enter:

```
host1/Admin# show crypto files
```

Table 6-2 describes the fields in the **show crypto files** command output.

Table 6-2 Field Descriptions for the show crypto files Command

Field	Description
Filename	Name of the file that contains the certificate or key pair.
File Size	Size of the file.
File Type	Format of the file: PEM, DER, or PKCS12.

Table 6-2 Field Descriptions for the show crypto files Command (continued)

Field	Description
Exportable	Indicates whether you can export the file from the ACE using the **crypto export** command: • **Yes**—You can export the file to an FTP, SFTP, or TFP server (see the "Exporting Certificate and Key Pair Files" section in Chapter 2, "Managing Certificates and Keys"). • **No**—You cannot export the file as it is protected.
Key/Cert	Indicates whether the file contains a certificate (CERT), a key pair (KEY), or both (BOTH).

Displaying Certificate Information

To display the certificate summary and detailed reports, use the **show crypto certificate** command in Exec mode.

The syntax of this command is as follows:

show crypto certificate {*filename* | **all**}

The keywords and arguments are as follows:

- *filename*—Name of a specific certificate file. Enter an unquoted alphanumeric string with a maximum of 40 characters. The ACE displays the certificate detailed report for the specified file. If the certificate file contains a chain, the ACE displays only the bottom level certificate (the signers are not displayed).
- **all**—Displays the certificate summary report that lists all the certificate files for the current context.

For example, to display the certificate summary report, enter:

```
host1/Admin# show crypto certificate all
```

Table 6-3 describes the fields in the **show crypto certificate all** command output.

Table 6-3 Field Descriptions for the show crypto certificate all Command

Field	Description
Certificate file	Name of the certificate file.
Subject	Distinguished name of the organization that owns the certificate and possesses the private key.
Issuer	Distinguished name of the Certificate Association (CA) that issued the certificate.
Not Before	Starting time period, before which the certificate is not considered valid.
Not After	Ending time period, after which the certificate is not considered valid.
CA Cert	Certificate of the CA that signed the certificate.

The following example shows how to display the detailed report for the MYCERT.PEM certificate file:

```
host1/Admin# show crypto certificate MYCERT.PEM
```

Table 6-4 describes the fields in the **show crypto certificate** *filename* command output.

Table 6-4 Field Descriptions for the show crypto certificate filename Command

Field	Description
Certificate	Name of the certificate file.
Data	
Version	Version of the X.509 standard. The certificate complies with this version of the standard.
Serial Number	Serial number associated with the certificate.
Signature Algorithm	Digital signature algorithm used for the encryption of information with a public/private key pair.

Table 6-4 Field Descriptions for the show crypto certificate filename Command (continued)

Field	Description
Issuer	Distinguished name of the CA that issued the certificate.
Validity	
Not Before	Starting time period, before which the certificate is not considered valid.
Not After	Ending time period, after which the certificate is not considered valid.
Subject	Distinguished name of the organization that owns the certificate and possesses the private key.
Subject Public Key Info	
Public Key Algorithm	Name of the key exchange algorithm used to generate the public key (for example, RSA).
RSA Public Key	Number of bits in the key to define the size of the RSA key pair used to secure web transactions.
Modulus	Actual public key on which the certificate was built.
Exponent	One of the base numbers used to generate the key.
X509v3 Extensions	Array of X509v3 extensions added to the certificate.
X509v3 Basic Constraints	Indicates whether the subject may act as a CA, with the certified public key being used to verify certificate signatures. If so, a certification path length constraint may also be specified.
Netscape Comment	Comment that may be displayed when the certificate is viewed.
X509v3 Subject Key Identifier	Public key to be certified. It enables distinct keys used by the same subject to be differentiated (for example, as key updating occurs).
X509v3 Authority Key Identifier	Public key to be used to verify the signature on this certificate or CRL. It enables distinct keys used by the same CA to be distinguished (for example, as key updating occurs).

Table 6-4 Field Descriptions for the show crypto certificate filename Command (continued)

Field	Description
Signature Algorithm	Name of the algorithm used for digital signatures (but not for key exchanges).
Hex Numbers	Actual signature of the certificate. The client can regenerate this signature using the specified algorithm to make sure that the certificate data has not been changed.

Displaying CRL Information

To display a list of certificate revocation lists (CRLs) or definitions for a specified CRL in a context, use the **show crypto crl** command in Exec mode. The syntax of this command is as follows:

show crypto crl {*crl_name* [**detail**] | **all** | **best-effort**}

The keywords and arguments are as follows:

- *crl_name*—Name of a specific CRL configured in the context. Enter an unquoted alphanumeric string. The ACE displays the definitions for the specified CRL.
- **detail**—(Optional) Displays detailed statistics for the downloading of the CRL including failure counters.
- **all**—Displays a lists of all CRLs configured in the context.
- **best-effort**—Displays summarized information for all best-effort CRLs on the ACE (a maximum of 16 CRLs).

For example, to display a list of all CRLs, enter:

```
host1/Admin# show crypto crl all
```

To display the definitions for a specific CRL, for example CRL1, enter:

```
host1/Admin# show crypto crl CRL1
```

Table 6-5 describes the fields in the **show crypto crl** *crl_name* command output.

Table 6-5 Field Descriptions for the show crypto crl Command

Field	Description
URL	URL where the ACE downloads the CRL.
Last Downloaded	Last time the ACE downloaded the CRL. If the CRL is configured on an SSL-proxy service on a policy map that is not active or the service is not associated with a policy map, the field displays the "not downloaded yet" message.
Total Number of Download Attempts	Number of times the ACE attempted to download the CRL.
Failed Download Attempts	Numbers of times the ACE failed to download the CRL.
Total Number of Download Attempts for Real CRL Data	Number of times the ACE attempted to download a specified CRL (not including "best effort" attempts).
Failed Download Attempts for Real CRL Data	Number of times the ACE failed to download a specified CRL (not including "best effort" attempts).
Successful Loads (**detail** option)	Number of times that the ACE successfully loaded the CRL.
Failed Loads (**detail** option)	Number of times that the ACE could not load the CRL because of a failure.
Hours since Last Load (**detail** option)	Number of hours that elapsed since the ACE last successfully downloaded the CRL. If no successful download has occurred, this field displays NA, not applicable.
No IP Addr Resolutions (**detail** option)	Number of times the DNS resolution for the server host address of CRL the failed.
Host Timeouts (**detail** option)	Number of download retries to the CRL that had timed out.
Next Update Invalid (**detail** option)	Number of times that the next update field of the CRL was invalid.

Table 6-5 Field Descriptions for the show crypto crl Command

Field	Description
Next Update Expired (**detail** option)	Number of times that the next update field of the CRL was expired.
Bad Signature (**detail** option)	Number of times that the signature mismatch for the CRL was detected, with respect to the CA certificate configured for signature verification of the CRL.
CRL Found-Failed to load (**detail** option)	Number of times that the ACE could not load the CRL because of the maximum size limitation of 10MB on ACE or the formatting of the CRL was not recognized. The ACE recognizes only DER and PEM encoded CRLs.
File Not Found (**detail** option)	Number of times that the server responded that the CRL file was not found at the server.
Memory Outage failures (**detail** option)	Number of times that the ACE failed to download the CRL because it temporarily could not provide memory to store the CRL data.
Cache Limit failures (**detail** option)	Number of times that the ACE could not load the CRL because the CRL cache was exhausted.
Conn Failures (**detail** option)	Number of times that the ACE failed to download the CRL because it could not establish a connection with the server or no server entity was listening on the destination system.
Internal Failures (**detail** option)	Number of internal failures in the ACE that hampered downloading the CRL, for example, internal communication failures between components responsible for the downloading the CRL.
Not Eligible for download (**detail** option)	Number of times that the CRL was found ineligible for downloading because the following conditions: • The downloading of the same CRL is in progress. • The CRL has already been loaded successfully earlier and has not expired yet.

Table 6-5 Field Descriptions for the show crypto crl Command

Field	Description
HTTP Read Failures (**detail** option)	Number of times that the ACE encountered an error when downloading the CRL because it could not read data on the connection established with server.
HTTP Write failures (**detail** option)	Number of times that the ACE encountered an error when downloading the CRL because it could not write the CRL download request from the connection established with the server.

For example, to display summarized information for all best-effort CRLs, enter:

```
host1/Admin# show crypto crl best-effort
```

Table 6-6 describes the fields in the **show crypto crl best-effort** command output.

Table 6-6 Field Descriptions for the show crypto crl best-effort Command

Field	Description
Best Effort CRL	Identifier to distinguish each best-effort CRL present at this time. At another time, the identifier can vary for the same CRL.
CRL Distribution Point	URL of the CDP. The ACE displays the first 255 characters of the URL.
CRL Downloaded	Whether the CRL is downloaded on the ACE module, Yes or No.
CRL Issuer Name	Name of the CRL issuer. The ACE displays the first 255 characters of the name.
Last Update	Contents of the Last Update field extracted from the CRL. The ACE displays the first 64 characters in the field
Next Update	Contents of the Next Update field extracted from the CRL. The ACE displays the first 64 characters in the field.

If no best-effort CRL exists on the ACE module, the ACE module displays the following message:

```
No best effort crl present in the system
```

Note To view whether the ACE rejects client certificates when the CRL in use is expired, use the **show parameter-map** command.

Displaying CDP Error Statistics

CRL Distribution Points (CDPs) indicate the location of the CRL in the form of a URL. CDP parsing in the certificate occurs only when best effort CRL is in use. To display statistics for discrepancies in CDPs for the certificates, use the **show crypto cdp-errors** command.

For example, to display the CDP statistics, enter:

```
host1/Admin# show crypto cdp-errors
```

Table 6-7 describes the fields in the **show crypto cdp-errors** command output.

Table 6-7 Field Descriptions for the show crypto cdp-errors Command

Field	Description
Incomplete	Number of times that the CDPs are missing information required to download the CRLs, for example, host, file name or base information.
Malformed	Number of times that the CDPs are malformed with erroneous information, for example, specifying an incorrect attribute or base information. This counter also includes CDPs with URL lengths exceeding the ACE limit of 255 characters; a truncated URL could point to the wrong CRL.
Unrecognized Transports	Number of times that the ACE does not recognize or support the transport mechanism in the CDP for the CRL.

Table 6-7 Field Descriptions for the show crypto cdp-errors Command (continued)

Field	Description
Missing from cert	Number of times that the CDPs are missing from the certificate.
Best Effort CDP Errors Ignored	Number of times that the ACE ignored CDP errors in the presented certificates, and thereby allowed the SSL connection. This field is related to the **cdp-errors ignore** command in parameter map SSL configuration mode.

Displaying RSA Key Pair Information

To display the key pair file summary and detailed reports, use the **show crypto key** command in Exec mode.

The syntax of this command is as follows:

show crypto key {*filename* | **all**}

The keywords and arguments are as follows:

- *filename*—Name of a specific key pair file. Enter an unquoted alphanumeric string with a maximum of 40 characters. The ACE displays the key pair detailed report for the specified file.
- **all**—Displays the key pair summary report that lists all of the available key pair files.

For example, to display the key pair summary report, enter:

```
host1/Admin# show crypto all
```

Table 6-8 describes the fields in the **show crypto key** command output.

Table 6-8 Field Descriptions for the show crypto key Command

Field	Description
Filename	Name of the key pair file that contains the RSA key pair.
Bit Size	Size of the file.
Type	Type of key exchange algorithm, such as RSA.

The following example shows how to display the detailed report for the public and private keys contained in the MYKEYS.PEM key pair file:

```
host1/Admin# show crypto key MYKEYS.PEM
1024-bit RSA keypair
```

Table 6-9 describes the fields in the **show crypto key** *filename* command output.

Table 6-9 Field Descriptions for the show crypto key filename Command

Field	Description
Key Size	Size (in bits) of the RSA key pair.
Modulus	Hex value of the public key. The private key modulus is not shown for security purposes.

Displaying Certificate Chain Group Information

To display the chain group file summary and detailed reports, use the **show crypto chaingroup** command in Exec mode.

The syntax of this command is as follows:

show crypto chaingroup {*filename* | **all**}

The keywords and arguments are as follows:

- *filename*—Name of a specific chain group file. Enter an unquoted alphanumeric string with a maximum of 64 characters. The ACE displays the chain group detailed report for the specified file. The detailed report contains a list of the certificates configured for the chain group.
- **all**—Displays the chain group summary report that lists each of the available chain group files. The summary report also lists the certificates configured for each chain group.

For example, to display the chain group summary report, enter:

```
host1/Admin# show crypto chaingroup all
```

The following example shows how to display the detailed report of the certificates configured for the MYCERTGROUP chain group:

```
host1/Admin# show crypto chaingroup MYCERTGROUP
```

Table 6-10 describes the fields in the **show crypto chaingroup** command output.

Table 6-10 Field Descriptions for the show crypto chaingroup Command

Field	Description
Certificate	Certificate filename.
Subject	Distinguished name of the organization that owns the certificate and possesses the private key.
Issuer	Distinguished name of the CA that issued the certificate.

Displaying Client Authentication Group Information

To display a list of certificates for each authentication group or the certificates in a specified client authentication group including the Subject and Issuer information for each certificate, use the **show crypto authgroup** command in Exec mode.

The syntax of this command is as follows:

show crypto authgroup {*group_name* | **all**}

The keywords and arguments are as follows:

- *group_name*—Name of a specific authentication group file. Enter an unquoted alphanumeric string with a maximum of 64 characters.
- **all**—Displays the list of certificates for each authentication groups.

For example, to display the list of certificates for each authentication group, enter:

```
host1/Admin# show crypto authgroup all
```

To display each certificate for the AUTH-CERT1 group including the Subject and Issuer information for each certificate, enter:

```
host1/Admin# show crypto authgroup AUTH-CERT1
```

Table 6-11 describes the fields in the **show crypto authgroup** *group_name* command output.

Table 6-11 Field Descriptions for the show crypto authgroup group_name Command

Field	Description
Certificate	Certificate filename.
Subject	Distinguished name of the organization that owns the certificate and possesses the private key.
Issuer	Distinguished name of the CA that issued the certificate.

Displaying Cached TLS and SSL Session Entries

To display the number of cached TLS and SSL client and server session entries in the current context, use the **show crypto session** command in Exec mode.

The syntax of this command is as follows:

show crypto session

For example, enter:

```
host1/Admin# show crypto session
```

Displaying SSL Parameter Map Settings

To display the settings in your SSL parameter map, use the **show parameter-map** command in Exec mode. The syntax of this command is as follows:

show parameter-map *name*

The *name* argument specifies the name of an existing SSL parameter map. Enter the SSL parameter name as an unquoted text string with a maximum of 64 alphanumeric characters.

For example:

```
host1/Admin# show parameter-map SSL_PARAMMAP
```

Table 6-12 describes the fields in the **show parameter-map** command output relating to the HTTP headers that provide the server with SSL session information. For information about the other fields that display with this command, see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

Table 6-12 Field Descriptions for the show parameter-map Command

Field	Description
Parameter-map	Name of the SSL parameter map
Type	SSL
Description	Previously entered text description of the SSL parameter map
version	Version of SSL or TLS
close-protocol	Status of the **close-protocol** command: none
expired-crl	Status of the **expired-crl** command: allow or reject
cdp-errors	Status of the **cdp-errors** command: allow or reject
authentication-failure any	Status of the **authenticiation-failure any** command: ignore

Table 6-12 Field Descriptions for the show parameter-map Command (continued)

Field	Description
session-cache timeout	Status of the **session-cache timeout** command: enabled or disabled
queue-delay timeout	Status of the **queue-delay timeout** command: enabled or disabled
rehandshake	Status of the **rehandshake enabled** command: enabled or disabled
purpose-check	Status of the **purpose-check** command: enabled or disabled

Displaying Front-End and Back-End SSL Statistics

To display the front-end and back-end SSL statistics for the current context, use the **show stats crypto** command in Exec mode. This command displays alert, authentication, cipher, header insertion, redirect, and termination statistics. To clear these statistics, see the "Clearing SSL and TLS Statistics" section.

To display the back-end SSL statistics, the syntax of this command is as follows:

show stats crypto client [alert | authentication | cipher | termination]

To display the front-end SSL statistics, the syntax of this command is as follows:

show stats crypto server [alert | authentication | cipher | insert | redirect | termination]

The keywords are as follows:

- **client**—Displays the back-end SSL statistics. Without any options, all statistics are displayed.
- **server**—Displays the front-end SSL statistics. Without any options, all statistics are displayed.
- **alert**—(Optional) Displays the statistics for the received and sent alert messages.
- **authentication**—(Optional) Displays authentication statistics.

- **cipher**—(Optional) Displays the cipher statistics.
- **insert**—(Optional) With the **server** keyword, this option displays header insertion statistics.
- **redirect**—(Optional) With the **server** keyword, this option displays the SSL redirect statistics.
- **termination**—(Optional) Displays the SSL termination statistics.

For example, to display the back-end statistics, enter:

```
host1/Admin# show stats crypto client
```

To display the front-end statistics, enter:

```
host1/Admin# show stats crypto server
```

Table 6-13 describes the fields in the **show stats crypto** command output. For an explanation of how the HTTP header insertion counters work, see the "Information about SSL HTTP Header Insertion and Truncated Counters" section.

Table 6-13 Field Descriptions for the show stats crypto Command

Field	Description
Crypto client/server termination statistics:	
SSLv3/TLSv1 negotiated protocol	Number of the times that the protocol is used when negotiating the connection.
SSLv3 full handshakes	Number of SSLv3 handshakes completed without errors.
SSLv3 resumed handshakes	Number of SSLv3 handshakes resumed when using a session ID.
SSLv3 handshakes	Number of SSLv3 handshakes when using a session ID.
TLSv1 full handshakes	Number of TLSv1 handshakes completed without errors.
TLSv1 resumed handshakes	Number of TLSv1 handshakes resumed when using a session ID.
TLSv1 handshakes	Number of TLSv1 handshakes when using a session ID.

Table 6-13 Field Descriptions for the show stats crypto Command (continued)

Field	Description
SSLv3 handshake failures	Number of SSLv3 handshake failures when using a session ID.
SSLv3 failures during data phase	Number of SSLv3 data exchange failures when using a session ID.
TLSv1 handshake failures	Number of TLSv1 handshake failures when using a session ID.
TLSv1 failures during data phase	Number of TLSv1 data exchange failures when using a session ID.
Handshake Timeouts	Number of times that the handshake timed out.
total transactions	Total number of all SSL transactions.
SSLv3 active connections	Number of SSLv3 active connections.
SSLv3 connections in handshake phase	Number of SSLv3 connections in the handshake phase.
SSLv3 conns in renegotiation phase	Number of SSLv3 connections in the renegotiation (rehandshake) phase.
SSLv3 connections in data phase	Number of SSLv3 connections in the data exchange phase of the session.
TLSv1 active connections	Number of TLSv1 active connections.
TLSv1 connections in handshake phase	Number of TLSv1 connections in the handshake phase.
TLSv1 conns in renegotiation phase	Number of TLSv1 connections in the renegotiation (rehandshake) phase.
TLSv1 connections in data phase	Number of TLSv1 connections in the data exchange phase of the session.
Crypto client/server alert statistics:	
SSL alert... rcvd/sent	Number of times that the standard SSL alert messages are received or sent.
Crypto client/server authentication statistics:	

Table 6-13 Field Descriptions for the show stats crypto Command (continued)

Field	Description
Total SSL client authentications	Number of authenticated client connections. This field increments only when displaying server statistics.
Failed SSL client authentications	Number of client connections that failed authentication. This field increments only when displaying server statistics.
SSL authentication cache hits	Number of times that an authenticated client reconnects and a cache entry is found. This field increments only when displaying server statistics.
SSL static CRL lookups	Number of lookups against a statically defined CRL.
SSL best effort CRL lookups	Number of lookups using the best effort.
SSL CRL lookup cache hits	Number of CRL lookups where the cache result was used.
SSL revoked certificates	Number of revoked certificates encountered.
Total SSL server authentications	Number of server certificate authentications that the ACE attempted to perform. This field increments only when displaying client statistics.
Failed SSL server authentications	Number of server certificate authentications that failed. This field increments only when displaying client statistics.
Crypto client/server cipher statistics:	
Cipher sslv3/tlsv1...	Number of times that the cipher suite is used in the connection.
Crypto client/server redirect statistics:	
Redirects due to cert not yet valid	Number of redirects because the certificate is not valid yet.
Redirects due to cert expired	Number of redirects because the certificate has expired.

Table 6-13 Field Descriptions for the show stats crypto Command (continued)

Field	Description
Redirects due to unknown issuer cert	Number of redirects because the ACE cannot retrieve issuer certificate.
Redirects due to cert revoked	Number of redirects because the certificate is revoked.
Redirects due to no client cert	Number of redirects because the client did not send a client certificate.
Redirects due to no CRL available	Number of redirects because a CRL was not available.
Redirects due to expired CRL	Number of redirects because the CRL has expired.
Redirects due to bad cert signature	Number of redirects because the certificate has a bad signature.
Redirects due to other cert error	Number of redirects caused by certificate errors that do not apply to the other redirect fields.
Crypto client/server header insert statistics:	
Session headers extracted	Number of HTTP headers that contain SSL-negotiated session parameter information that the ACE successfully added to the HTTP header information build[1].
Session headers failed	Number of HTTP headers that contain SSL-negotiated session parameter information that the ACE could not add to the HTTP header information build[1].
Server cert headers extracted	Number of HTTP headers that contain SSL server certificate information that the ACE successfully added to the HTTP header information build[1].
Server cert headers failed	Number of HTTP headers that contain SSL server certificate information that the ACE could not add to the HTTP header information build[1].
Client cert headers extracted	Number of HTTP headers that contain SSL client certificate information that the ACE successfully added to the HTTP header information build[1].

Table 6-13 Field Descriptions for the show stats crypto Command (continued)

Field	Description
Client cert headers failed	Number of HTTP headers that contain SSL client certificate information that the ACE could not add to the HTTP header information build[1].
Headers truncated	Number of HTTP headers that contain the SSL negotiated session parameter, server certificate, or client certificate information that the ACE truncated because the combined header information exceeded 512 bytes[1].
Headers insert buffer limit hit	Number of times that the buffer has reached its 512 byte limit and is not available to perform header insertion. This field increments when no part of a header is inserted because of lack of buffer space.

1. For more information, see the "Information about SSL HTTP Header Insertion and Truncated Counters" section.

Information about SSL HTTP Header Insertion and Truncated Counters

When you configure the ACE for SSL HTTP header insertion, the ACE creates a build of the HTTP header information during the SSL handshake with the client. This information is based on the SSL negotiated session parameters, client certificate parameters, or server certificate parameters that you specify in the action list. When the ACE receives the session's first HTTP request, it performs the HTTP header insert operation and inserts the HTTP header build.

While the ACE is creating the HTTP header build, it uses the following counters to track the success rate of the information being inserted:

- "(header type) headers extracted" counters—The ACE increments the corresponding header type counter (session, server certificate, or client certificate) by the number of headers that it can successfully add to the information being built for the HTTP header insertion operation.

- "(header type) headers failed" counters—The ACE increments the corresponding header type counter (session, server certificate, or client certificate) by the number of headers that it is unable to add to the information being built for the HTTP header insertion operation. The ACE is unable to insert a header because it encounters either an internal error (such as not being able to allocate memory) or an error when parsing a certificate field (for example, the certificate has an invalid date specified date field).
- Headers truncated—The ACE increments this counter every time it truncates a header because the combined header information exceeds 512 bytes.

The ACE creates only one build of the header information per session, which means that it inserts the same build even when you configure the ACE to insert the information into all the HTTP requests that it receives during the session. Because the same build is used for all session HTTP requests, the counters increment during the build process only and not every time the ACE performs the HTTP header insertion operation. For information about the counters that track the success rate of the HTTP header insertion operation, see the "Displaying HTTP Header Insertion Statistics" section.

Note It is possible for the ACE to extract the header information during the SSL handshake but not insert the information into the HTTP request. This situation can occur if the SSL handshake fails after the ACE extracts the header information but before it receives the first GET. When this situation occurs, the SSL counters increment but the HTTP counters do not increment.

Displaying HTTP Header Insertion Statistics

You can display HTTP statistics, including information relating to the HTTP headers that contain SSL session information, by using the **show stats http** command in Exec mode. The syntax of this command is as follows:

show stats http

Table 6-14 describes the fields in the **show stats http** command output relating to the HTTP headers that provide the server with SSL session information. For information about the other fields that display with this command, see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

Table 6-14 Field Descriptions for the show stats http Command

Field	Description
SSL headers inserted	Number of times that the ACE successfully performed the HTTP header insert operation by inserting all of the HTTP headers that contain SSL session, client certificate, and server certificate information defined in the corresponding action list into the HTTP request.
SSL header insert errors	Number of times that the ACE failed to perform the HTTP header insert operation completely because it could not insert the HTTP headers that contain the SSL session information defined in the corresponding action list.
SSL spoof headers deleted	Number of times that the ACE deleted an HTTP header from the HTTP request that it received over the client connection. To prevent HTTP header spoofing, the ACE deletes any incoming HTTP headers that contain SSL session information that matches any of the headers that it has to insert.

Clearing SSL and TLS Statistics

You can clear the SSL and TLS statistics displayed by the **show stats crypto** command for the current context by using the **clear stats crypto** command in Exec mode. The syntax for this command is as follows:

clear stats crypto [**client** | **server** [**alert** | **authentication** | **cipher** | **termination**]]

The options are as follow:

- **client**—(Optional) Clears the complete TLS and SSL client statistics for the current context.
- **server**—(Optional) Clears the complete TLS and SSL server statistics for the current context.

- **alert**—(Optional) Clears the back-end SSL alert statistics.
- **authentication**—(Optional) Clears the back-end SSL authentication statistics.
- **cipher**—(Optional) Clears the back-end SSL cipher statistics.
- **termination**—(Optional) Clears the back-end SSL termination statistics.

If you do not enter the **client** or **server** option, the ACE clears both the client and server statistics.

For example, to clear all TLS and SSL statistics, enter the following:

```
host1/Admin# show stats crypto
```

INDEX

A

C

D

E

H

I

K

M

P

Q

R

S

T

U

V