Cisco Application Control Engine Module Server Load-Balancing Configuration Guide

Software Version A2(3.0)
October 2009

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-20818-01

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Cisco Application Control Engine Module Server Load-Balancing Configuration Guide

CONTENTS

CHAPTER 2

CHAPTER 4

Preface

This guide provides instructions for implementing server load balancing (SLB) on the Cisco Application Control Engine (ACE) module in a Catalyst 6500 series switch or a Cisco 7600 series router, hereinafter referred to as the switch or router, respectively.

You can configure the ACE by using the following interfaces:

- The command-line interface (CLI), a line-oriented user interface that provides commands for configuring, managing, and monitoring the ACE.
- Device Manager graphic user interface (GUI), a Web browser-based GUI interface that provides a graphical user interface for configuring, managing, and monitoring the ACE.

This preface contains the following major sections:

- Audience
- How to Use This Guide
- Related Documentation
- Symbols and Conventions
- Obtaining Documentation, Obtaining Support, and Security Guidelines

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 as follows:

Chapter	Description
Chapter 1, Overview	Describes SLB as implemented in the ACE. This chapter includes a procedure that describes how to configure the ACE for load balancing only.
Chapter 2, Configuring Real Servers and Server Farms	Describes real servers, server farms, and load-balancing methods and how to configure them for SLB.
Chapter 3, Configuring Traffic Policies for Server Load Balancing	Describes how to configure class maps to filter interesting SLB traffic and configure policy maps to perform actions on that traffic. Also describes SLB parameter maps and applying policies to interfaces.
Chapter 4, Configuring Health Monitoring	Describes how to configure health probes (keepalives) to monitor the health and status of real servers.
Chapter 5, Configuring Stickiness	Describes how to configure stickiness (connection persistence) to ensure that a client remains stuck to the same server for the duration of a session.
Chapter 6, Configuring Firewall Load Balancing	Describes how to configure firewall load balancing (FWLB) to load balance traffic from the Internet through a firewall to a data center or intranet.
Appendix A, Using TCL Scripts with the ACE	Describes how to upload and execute Toolkit Command Language (TCL) scripts on the ACE.

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 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)
Cisco Application Control Engine Module SSL Configuration Guide	Describes how to configure the following Secure Sockets Layer (SSL) features on the ACE: • SSL certificates and keys • SSL initiation • SSL termination • End-to-end SSL
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.

Document Title	Description
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) Module Troubleshooting Guide, Release A2(x)	Describes the procedures and methodology in wiki format to troubleshoot the most common problems that you may encounter during the operation of your ACE.

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.

Convention	Description
[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.

Notes use the following conventions:

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

Cautions use the following conventions:

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 CLI syntax formatting, 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

CHAPTER 1

Overview

This chapter describes server load balancing (SLB) as implemented in the Cisco Application Control Engine (ACE) module. It contains the following major sections:

- Server Load-Balancing Overview
- Load-Balancing Predictors
- Real Servers and Server Farms
- Configuring Traffic Classifications and Policies
- Connection Limits and Rate Limiting
- Operating the ACE Strictly as a Load Balancer
- Where to Go Next

Server Load-Balancing Overview

Server load balancing (SLB) is the process of deciding to which server a load-balancing device should send a client request for service. For example, a client request may consist of a HyperText Transport Protocol (HTTP) GET for a web page or a File Transfer Protocol (FTP) GET to download a file. The job of the load balancer is to select the server that can successfully fulfill the client request and do so in the shortest amount of time without overloading either the server or the server farm as a whole.

The ACE supports the load balancing of the following protocols:

- Generic protocols
- HTTP
- Remote Authentication Dial-In User Service (RADIUS)
- Reliable Datagram Protocol (RDP)
- Real-Time Streaming Protocol (RTSP)
- Session Initiation Protocol (SIP)

Depending on the load-balancing algorithm—or *predictor*—that you configure, the ACE performs a series of checks and calculations to determine which server can best service each client request. The ACE bases server selection on several factors including the source or destination address, cookies, URLs, HTTP headers, or the server with the fewest connections with respect to load.

Load-Balancing Predictors

The ACE uses the following predictors to select the best server to fulfill a client request:

- Application response—Selects the server with the lowest average response time for the specified response-time measurement based on the current connection count and server weight (if configured).
- Hash address—Selects the server using a hash value based on either the source or destination IP address or both. Use these predictors for firewall load balancing (FWLB). For more information about FWLB, see Chapter 6, Configuring Firewall Load Balancing.
- Hash content—Selects the server using a hash value based on a content string in the Trusted Third Parties (TTP) packet body.
- Hash cookie—Selects the server using a hash value based on a cookie name.
- Hash header—Selects the server using a hash value based on the HTTP header name.
- Hash URL—Selects the server using a hash value based on the requested URL. You can specify a beginning pattern and an ending pattern to match in the URL. Use this predictor method to load balance cache servers.

- Least bandwidth—Selects the server that processed the least amount of network traffic based on the average bandwidth that the server used over a specified number of samples.
- Least connections—Selects the server with the fewest number of active connections based on server weight. For the least-connections predictor, you can configure a slow-start mechanism to avoid sending a high rate of new connections to servers that you have just put into service.
- Least loaded—Selects the server with the lowest load based on information obtained from Simple Network Management Protocol (SNMP) probes. To use this predictor, you must associate an SNMP probe with it.
- Round-robin—Selects the next server in the list of real servers based on the server weight (weighted round-robin). Servers with a higher weight value receive a higher percentage of the connections. This is the default predictor.

Note The hash predictor methods do not recognize the weight value that you configure for real servers. The ACE uses the weight that you assign to real servers only in the least-connections, application-response, and round-robin predictor methods.

For more information about load-balancing predictors, see Chapter 2, Configuring Real Servers and Server Farms.

Real Servers and Server Farms

This section briefly describes real servers and server farms and how they are implemented on the ACE. It contains the following topics:

- Real Servers
- Server Farms
- Health Monitoring

Real Servers

To provide services to clients, you configure *real servers* (the actual physical servers) on the ACE. Real servers provide client services such as HTTP or XML content, hosting websites, FTP file uploads or downloads, redirection for web pages that have moved to another location, and so on. The ACE also allows you to configure backup servers in case a server is taken out of service for any reason.

After you create and name a real server on the ACE, you can configure several parameters, including connection limits, health probes, and weight. You can assign a weight to each real server based on its relative importance to other servers in the server farm. The ACE uses the server weight value for the weighted round-robin and the least-connections load-balancing predictors. For a listing and brief description of the ACE predictors, see the "Load-Balancing Predictors" section. For more detailed information about the ACE load-balancing predictors and server farms, see Chapter 2, Configuring Real Servers and Server Farms.

Server Farms

Typically, in data centers, servers are organized into related groups called *server farms*. Servers within server farms often contain identical content (referred to as mirrored content) so that if one server becomes inoperative, another server can take its place immediately. Also, mirrored content allows several servers to share the load of increased demand during important local or international events, for example, the Olympic Games. This sudden large demand for content is called a *flash crowd*.

After you create and name a server farm, you can add existing real servers to it and configure other server-farm parameters, such as the load-balancing predictor, server weight, backup server, health probe, and so on. For a description of the ACE predictors, see the "Load-Balancing Predictors" section. For more detailed information about the ACE load-balancing predictors and server farms, see Chapter 2, Configuring Real Servers and Server Farms.

Health Monitoring

You can instruct the ACE to check the health of servers and server farms by configuring health probes (sometimes referred to as *keepalives*). After you create a probe, you assign it to a real server or a server farm. A probe can be one of many types, including TCP, ICMP, Telnet, HTTP, and so on. You can also configure scripted probes using the TCL scripting language.

The ACE sends out probes periodically to determine the status of a server, verifies the server response, and checks for other network problems that may prevent a client from reaching a server. Based on the server response, the ACE can place the server in or out of service, and, based on the status of the servers in the server farm, can make reliable load-balancing decisions. For more information about out-of-band health monitoring, see Chapter 4, Configuring Health Monitoring.

Configuring Traffic Classifications and Policies

The ACE uses several configuration elements to classify (filter) interesting traffic and then to perform various actions on that traffic before making the load-balancing decision. These filtering elements and subsequent actions form the basis of a traffic policy for SLB. This section contains the following topics:

- Filtering Traffic with ACLs
- Classifying Layer 3 and Layer 4 Traffic
- Classifying Layer 7 Traffic
- Configuring a Parameter Map
- Creating Traffic Policies
- Applying Traffic Policies to an Interface Using a Service Policy

Filtering Traffic with ACLs

To permit or deny traffic to or from a specific IP address or an entire network, you can configure an access control list (ACL). ACLs provide a measure of security for the ACE and the data center by allowing access only to traffic that you explicitly authorize on a specific interface or on all interfaces. An ACL consists of a series of permit or deny entries with special criteria for the source address, destination address, protocol, port, and so on. All ACLs contain an implicit deny statement, so you must include an explicit permit entry to allow traffic to and through the ACE. For more information about ACLs, see the *Cisco Application Control Engine Module Security Configuration Guide*.

Classifying Layer 3 and Layer 4 Traffic

To classify Layer 3 and Layer 4 network traffic, you configure class maps and specify match criteria according to your application requirements. When a traffic flow matches certain match criteria, the ACE applies the actions specified in the policy map with which the class map is associated. A policy map acts on traffic ingressing the interface to which the policy map is applied through a service policy (globally to all VLAN interfaces in a context or to a single VLAN interface).

Class maps that operate at Layer 3 and Layer 4 for SLB typically use virtual IP (VIP) addresses as matching criteria. For details about Layer 3 and Layer 4 class maps for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Classifying Layer 7 Traffic

In addition to Layer 3 and Layer 4 class maps, you can also configure Layer 7 class maps for advanced load-balancing matching criteria, such as HTTP cookie, header, and URL settings. After you configure a Layer 7 class map, you associate it with a Layer 7 policy map. You cannot apply a Layer 7 policy map to an interface directly (see the "Creating Traffic Policies" section). For details about Layer 7 class maps for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Configuring a Parameter Map

A parameter map combines related HTTP or RTSP actions for use in a Layer 3 and Layer 4 policy map. Parameter maps provide a means of performing actions on traffic that ingresses an ACE interface based on certain criteria, such as HTTP header and cookie settings, server connection reuse, the action to take when an HTTP header, cookie, or URL exceeds a configured maximum length, and so on. After you configure a parameter map, you associate it with a Layer 3 and Layer 4 policy map. For details about configuring an HTTP or RTSP load-balancing parameter map, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Creating Traffic Policies

The ACE uses policy maps to combine class maps and parameter maps into traffic policies and to perform certain configured actions on the traffic that matches the specified criteria in the policies. Policy maps operate at Layer 3 and Layer 4, as well as Layer 7. Because the ACE considers a Layer 7 policy map a child policy, you must associate each Layer 7 policy map with a Layer 3 and Layer 4 policy map before you can apply the traffic policy to an interface using a service policy. For more information about configuring SLB traffic policies, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Applying Traffic Policies to an Interface Using a Service Policy

To apply a traffic policy to one or more interfaces, you use a service policy. You can use a service policy on an individual interface or globally on all interfaces in a context in the input direction only. When you use a service policy on an interface, you apply and activate a Layer 3 and Layer 4 policy map with all its class-map, parameter-map, and Layer 7 policy-map associations and match criteria. For more information about using a service policy to apply a traffic policy to an interface, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Connection Limits and Rate Limiting

To help protect system resources, the ACE allows you to limit the following items:

- Maximum number of connections
- Connection rate (connections per second applied to new connections destined to a real server)
- Bandwidth rate (bytes per second applied to network traffic between the ACE and a real server in both directions)

For more information, see Chapter 2, Configuring Real Servers and Server Farms and the *Cisco Application Control Engine Module Security Configuration Guide*.

Operating the ACE Strictly as a Load Balancer

You can operate your ACE strictly as an SLB device. If you want to use SLB only, you must configure certain parameters and disable some of the ACE security features. By default, the ACE performs TCP/IP normalization checks and ICMP security checks on traffic that enters the ACE interfaces. Using the following configuration will also allow asymmetric routing as required by your network application.

The major configuration items are as follows:

- Configuring a global permit-all ACL and applying it to all interfaces in a context to open all ports
- Disabling TCP/IP normalization
- Disabling ICMP security checks
- Configuring SLB

To operate the ACE for SLB only, perform the following steps:

Step 1 Configure a global permit-all ACL and apply it to all interfaces in a context. This action will open all ports in the current context.

```
host1/Admin(config)# access-list ACL1 extended permit ip any any
host1/Admin(config)# access-group input ACL1
```

Step 2 Disable the default TCP/IP normalization checks on each interface where you want to configure a VIP using a load-balancing service policy. For details about TCP normalization, see the *Cisco Application Control Engine Module Security Configuration Guide*.

```
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# no normalization
```

⚠ **Caution** Disabling TCP normalization may expose your ACE and your data center to potential security risks. TCP normalization helps protect the ACE and the data center from attackers by enforcing strict security policies that are designed to examine traffic for malformed or malicious segments.

Step 3 Disable the default ICMP security checks on each interface where you want to configure a VIP using a load-balancing service policy. For details about the ICMP security checks, see the *Cisco Application Control Engine Module Security Configuration Guide*.

```
host1/Admin(config-if)# no icmp-guard
```

⚠ **Caution** Disabling the ACE ICMP security checks may expose your ACE and your data center to potential security risks. In addition, after you enter the **no icmp-guard** command, the ACE no longer performs Network Address Translations (NATs) on the ICMP header and payload in error packets, which potentially can reveal real host IP addresses to attackers.

Step 4 Configure SLB. For details, see the remaining chapters in this guide.

Where to Go Next

To start configuring SLB on your ACE, proceed to Chapter 2, Configuring Real Servers and Server Farms.

CHAPTER 2

Configuring Real Servers and Server Farms

This chapter describes the functions of real servers and server farms in load balancing and how to configure them on the ACE module. It contains the following major sections:

- Configuring Real Servers
- Configuring a Server Farm
- Displaying Real Server Configurations and Statistics
- Clearing Real Server Statistics and Connections
- Displaying Server Farm Configurations and Statistics
- Clearing Server Farm Statistics
- Where to Go Next

This chapter also provides information on the Asymmetric Server Normalization feature, as described in the "Configuring Asymmetric Server Normalization" section.

Configuring Real Servers

This section describes real servers and how to configure them. It contains the following topics:

Real Server Overview

Real servers are dedicated physical servers that you typically configure in groups called server farms. These servers provide services to clients, such as HTTP or XML content, streaming media (video or audio), TFTP or FTP uploads and downloads, and so on. You identify real servers with names and characterize them with IP addresses, connection limits, and weight values.

The ACE uses traffic classification maps (class maps) within policy maps to filter out interesting traffic and to apply specific actions to that traffic based on the SLB configuration. You use class maps to configure a virtual server address and definition. The load-balancing predictor algorithms (for example, round-robin, least connections, and so on) determine the servers to which the ACE sends connection requests. For information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Real Server Configuration Quick Start

Table 2-1 provides a quick overview of the steps required to configure real servers. 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 2-1.

Table 2-1 Real Server 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. Change to, or directly log in to, the correct context if necessary.

   ```
   host1/Admin# changeto C1
   host1/C1#
   ```

   The rest of the examples in this table use the Admin context, unless otherwise specified. For details about creating contexts, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

2. Enter configuration mode.

   ```
   host/Admin# config
   Enter configuration commands, one per line. End with CNTL/Z
   host1/Admin(config)#
   ```

3. Configure a real server and enter real server configuration mode.

   ```
   host1/Admin(config)# rserver SERVER1
   host1/Admin(config-rserver-host)#
   ```

4. (Recommended) Enter a description of the real server.

   ```
   host1/Admin(config-rserver-host)# description accounting
   department server
   ```

Table 2-1 Real Server Configuration Quick Start (continued)

Task and Command Example
5. Configure an IP address for the real server in dotted-decimal notation. `host1/Admin(config-rserver-host)# ip address 192.168.12.15`
6. Assign one or more existing probes to the real server for health monitoring. `host1/Admin(config-rserver-host)# probe PROBE1`
7. To prevent the real server from becoming overloaded, configure connection limits. `host1/Admin(config-rserver-host)# conn-limit max 2000000 min 1500000`
8. If you plan to use the weighted round-robin or least connections predictor method, configure a weight for the real server. `host1/Admin(config-rserver-host)# weight 50`

9. Place the real server in service.
`host1/Admin(config-rserver-host)# inservice`
`host1/Admin(config-rserver-host)# Ctrl-Z`

10. Use the following command to display the real server configuration. Make any modifications to your configuration as necessary, then reenter the command to verify your configuration changes. `host1/Admin# show running-config rserver`
11. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating a Real Server

You can configure a real server and enter real server configuration mode by using the **rserver** command in configuration mode. You can create a maximum of 16,383 real servers. The syntax of this command is as follows:

rserver [**host** | **redirect**] *name*

The keywords, arguments, and options for this command are as follows:

- **host**—(Optional, default) Specifies a typical real server that provides content and services to clients.

- **redirect**—(Optional) Specifies a real server used to redirect traffic to a new location as specified in the *relocn-string* argument of the **webhost-redirection** command. See the "Configuring a Real Server Relocation String" section.
- *name*—Identifier for the real server. Enter an unquoted text string with no spaces and maximum of 64 alphanumeric characters.

Note You can associate a real sever of type **host** only with a server farm of type **host**. You can associate a real server of type **redirect** only with a server farm of type **redirect**.

For example, to create a real server of type host, enter:

```
host1/Admin(config)# rserver server1
host1/Admin(config-rserver-host)#
```

To remove the real server of type host from the configuration, enter:

```
host1/Admin(config)# no rserver server1
```

To create a real server of type redirect, enter:

```
host1/Admin(config)# rserver redirect server2
host1/Admin(config-rserver-redir)#
```

To remove the real server of type redirect from the configuration, enter:

```
host1/Admin(config)# no rserver redirect server2
```

Note The sections that follow apply to both real server types unless otherwise indicated.

Configuring a Real Server Description

You can configure a description for a real server by using the **description** command in either real server host, real server redirect, or serverfarm host real server configuration mode. The syntax of this command is as follows:

description *string*

For the *string* argument, enter an alphanumeric text string and a maximum of 240 characters, including quotation marks (“ “) and spaces. In serverfarm host real server configuration mode, if the text string includes spaces, enclose the string in quotes.

For example, enter:

```
host1/Admin(config)# rserver server1
host1/Admin(config-rserver-host)# description accounting server
```

To remove the real-server description from the configuration, enter:

```
host1/Admin(config-rserver-host)# no description
```

Configuring a Real Server IP Address

You can configure an IP address so that the ACE can access a real server of type **host**. You can configure an IP address by using the **ip address** command in real server host configuration mode. The syntax of this command is as follows:

ip address *ip_address*

For the *ip_address* argument, enter an IPv4 address in dotted-decimal notation (for example, 192.168.12.15). The IP address must be unique within the current context.

For example, to specify an address enter:

```
host1/Admin(config)# rserver server1
host1/Admin(config-rserver-host)# ip address 192.168.12.15
```

To remove the real server IP address from the configuration, enter:

```
host1/Admin(config-rserver-host)# no ip address
```

Configuring Real Server Health Monitoring

To check the health and availability of a real server, the ACE periodically sends a probe to the real server. Depending on the server response, the ACE determines whether to include the server in its load-balancing decision. For more information about probes, see Chapter 4, Configuring Health Monitoring.

You can assign one or more existing probes to a real server by using the **probe** command in real server host configuration mode. This command applies only to real servers of type **host**. The syntax of this command is as follows:

probe *name*

For the *name* argument, enter the name of an existing probe. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# rserver server1
host1/Admin(config-rserver-host)# probe probe1
```

To remove a real server probe from the configuration, enter:

```
host1/Admin(config-rserver-host)# no probe probe1
```

Configuring AND Logic for Real Server Probes

By default, real servers with multiple probes configured on them have an OR logic associated with them. This means that if one of the real server probes fails, the real server fails and enters the PROBE-FAILED state. To configure a real server to remain in the OPERATIONAL state unless all probes associated with it fail (AND logic), use the **fail-on-all** command in real server host configuration mode. This command is applicable to all probe types. For more information about probes, see Chapter 4, Configuring Health Monitoring.

Note You can configure the **fail-on-all** command also on server farms and real servers in server farms. See the "Configuring AND Logic for Server Farm Probes" section and the "Configuring AND Logic for Real Server Probes in a Server Farm" section.

The syntax of this command is as follows:

fail-on-all

For example, to configure the SERVER1 real server to remain in the OPERATIONAL state unless all associated probes fail, enter:

```
host1/Admin(config)# rserver SERVER1
host1/Admin(config-rserver-host)# ip address 192.168.12.15
host1/Admin(config-rserver-host)# probe HTTP_PROBE
host1/Admin(config-rserver-host)# probe ICMP_PROBE
host1/Admin(config-rserver-host)# fail-on-all
```

In this example, if HTTP_PROBE fails, the SERVER1 real server remains in the OPERATIONAL state. If both probes fail, the real server fails and enters the PROBE-FAILED state.

To remove the AND probe logic from the real server and return the behavior to the default of OR logic, enter:

```
host1/Admin(config-rserver-host)# no fail-on-all
```

Configuring Real Server Connection Limits

To prevent a real server from being overburdened and to conserve system resources, you can limit the maximum number of active connections to the server. You can set the maximum and minimum connection thresholds by using the **conn-limit** command in either real server host or real server redirect configuration mode. The syntax of this command is as follows:

conn-limit max *maxconns* **min** *minconns*

The keywords and arguments are as follows:

- **max** *maxconns*—Specifies the maximum allowable number of active connections to a real server. When the number of connections exceeds the *maxconns* threshold value, the ACE stops sending connections to the real server and assigns the real server a state of MAXCONNS until the number of connections falls below the configured *minconns* value. Enter an integer from 2 to 4000000. The default is 4000000.
- **min** *minconns*—Specifies the minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold. Enter an integer from 2 to 4000000. The default is 4000000. The *minconns* value must be less than or equal to the *maxconns* value.

Because the ACE has two network processors (NPs), the *maxconns* value for a real server is divided as equally as possible between the two NPs. If you configure an odd value for *maxconns*, one of the NPs will have a *maxconns* value that is one more than the other. With very small values for *maxconns*, a connection may be denied even though one of the NPs has the capacity to handle the connection.

Consider the scenario where you configure a value of 3 for the *maxconns* argument. One NP will have a value of 2 and the other NP will have a value of 1. If both NPs have reached their connection limits for that server, the next connection request for that server will be denied and the Out-of-rotation Count field of the **show serverfarm detail** command will increment by 1. Now, suppose that a connection is closed on the NP with the maxconns value of 2. If the next connection request to the ACE hits the NP with the maxconns value of 1 and it has already reached its limit, the ACE denies the connection, even though the other NP has the capacity to service the connection.

If you change the *minconns* value while there are live connections to a real server, the server could enter the MAXCONNS state without actually achieving the *maxconns* value in terms of the number of connections to it. Consider the following scenario where you configure a maxconns value of 10 and a minconns value of 5. Again, the ACE divides the values as equally as possible between the two NPs. In this case, both NPs would have a *maxconns* value of 5, while NP1 would have a *minconns* value of 3 and NP2 would have a *minconns* value of 2.

Suppose that the real server now has 10 live connections to it. Both NPs and the server have reached the MAXCONNS state. If four connections to the real server are closed leaving six live connections, both NPs (and, hence, the real server) would still be in the MAXCONNS state with three connections each. Remember, for an NP to come out of the MAXCONNS state, the number of connections to it must be less than the *minconns* value.

If you change the server's *minconns* value to 7, NP1 would enter the OPERATIONAL state because the number of connections (three) is one less than the *minconns* value of 4. NP2 would still be in the MAXCONNS state (*minconns* value = number of connections = 3). NP1 could process two more connections for a total of only eight connections to the real server, which is two less than the server's *maxconns* value of 10.

You can also specify minimum and maximum connections for a real server in a server farm configuration. The total of the minimum and maximum connections that you configure for a server in a server farm cannot exceed the minimum and maximum connections you set globally in real server configuration mode. For details about configuring real server maximum connections in a server farm configuration, see the "Configuring Connection Limits for a Real Server in a Server Farm" section.

For example, enter:

```
host1/Admin(config)# rserver server1
host1/Admin(config-rserver-host)# conn-limit max 5000 min 4000
```

To reset the real-server maximum connection limit and minimum connection limit to the default value of 4000000, enter:

```
host1/Admin(config-rserver-host)# no conn-limit
```

Configuring Real Server Rate Limiting

In addition to preserving system resources by limiting the total number of active connections to a real server (see the "Configuring Real Server Connection Limits" section), the ACE allows you to limit the connection rate and the bandwidth rate of a real server. The connection rate is the number of connections per second received by the ACE and applied only to the new connections destined to a real server. The bandwidth rate is the number of bytes per second applied to the network traffic exchanged between the ACE and the real server in both directions.

For a real server that is associated with more than one server farm, the ACE uses the aggregated connection rate or bandwidth rate to determine if the real server has exceeded its rate limits. If the connection rate or the bandwidth rate of incoming traffic destined for a particular server exceeds the configured rate limits of the server, the ACE blocks any further traffic destined to that real server until the connection rate or bandwidth rate drops below the configured limit. Also, the ACE removes the blocked real server from future load-balancing decisions and considers only those real servers in the server farm that have a current connection rate or bandwidth rate less than or equal to the configured limit. By default, the ACE does not limit the connection rate or the bandwidth rate of real servers.

You can also limit the connection rate and the bandwidth rate of the following:

- Real server in a server farm. For details, see the "Configuring Connection Limits for a Real Server in a Server Farm" section.

- Virtual server in a connection parameter map. For details, see the *Cisco Application Control Engine Module Security Configuration Guide*.

To limit the connection rate or the bandwidth rate of a real server, use the **rate-limit** command in real server host configuration mode. The syntax of this command is as follows:

rate-limit {**connection** *number1* | **bandwidth** *number2*}

The keywords and arguments are as follows:

- **connection** *number1*—Specifies the real server connection-rate limit in connections per second. Enter an integer from 2 to 350000. There is no default value.
- **bandwidth** *number2*—Specifies the real server bandwidth-rate limit in bytes per second. Enter an integer from 2 to 300000000. There is no default value.

Note The ACE applies rate limiting vales across both network processors (NPs). For example, if you configure a rate-limiting value of 500, the ACE applies a rate-limit value of 250 to each NP.

For example, to limit the connection rate of a real server at the aggregate level to 100,000 connections per second, enter:

```
host1/Admin(config)# rserver host SERVER1
host1/Admin(config-rserver-host)# rate-limit connection 100000
```

To return the behavior of the ACE to the default of not limiting the real-server connection rate, enter:

```
host1/Admin(config-rserver-host)# no rate-limit connection 100000
```

For example, to limit the real-server bandwidth rate to 5000000 bytes per second, enter:

```
host1/Admin(config)# rserver host SERVER1
host1/Admin(config-rserver-host)# rate-limit bandwidth 5000000
```

To return the behavior of the ACE to the default of not limiting the real-server bandwidth, enter:

```
host1/Admin(config-rserver-host)# no rate-limit bandwidth 5000000
```

Configuring a Real Server Relocation String

You can configure a real server to redirect client requests to a location specified by a relocation string or a port number. You can configure a relocation string for a real server that you configured as a redirection server (type **redirect**) by using the **webhost-redirection** command in real server redirect configuration mode. The syntax of this command is as follows:

webhost-redirection *relocation_string* [**301** | **302**]

The keywords and arguments are as follows:

- *relocation_string*—URL string used to redirect requests to another server. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. The relocation string supports the following special characters:
  - **%h**—Inserts the hostname from the request Host header
  - **%p**—Inserts the URL path string from the request

  Note To insert a question mark (?) in the relocation string, press **Ctrl-v** before you type the question mark.

- [**301** | **302**]—Specifies the redirection status code returned to a client. The codes indicate the following:
  - **301**—The requested resource has been moved permanently. For future references to this resource, the client should use one of the returned URLs.
  - **302**—(Default) The requested resource has been found but has been moved temporarily to another location. For future references to this resource, the client should continue to use the request URI because the resource may be moved to other locations occasionally.

  For more information about redirection status codes, see RFC 2616.

For example, enter:

```
host1/Admin(config)# rserver redirect SERVER1
host1/Admin(config-rserver-redir)# webhost-redirection
http://%h/redirectbusy.cgi?path=%p 302
```

Assume the following client GET request:

```
GET /helloworld.html HTTP/1.1
Host: www.cisco.com
```

The redirect response would appear as follows:

```
HTTP/1.1 302 Found
Location: http://www.cisco.com/redirectbusy.cgi?path=/helloworld.html
```

To remove the relocation string from the configuration, enter:

```
host1/Admin(config-rserver-redir)# no webhost-redirection
```

Configuring a Real Server Weight

You can configure the connection capacity of a real server of type **host** or **redirect** in relation to the other servers in a server farm by using the **weight** command in real server host configuration mode. The ACE uses the weight value that you specify for a server in the weighted round-robin and least-connections load-balancing predictors. Servers with a higher configured weight value have a higher priority with respect to connections than servers with a lower weight. For example, a server with a weight of 5 would receive five connections for every one connection for a server with a weight of 1.

Note For the least-connections predictor, server weights take effect only when there are open connections to the real servers. When there are no sustained connections to any of the real servers, the least-connections predictor behaves like the round-robin predictor.

To specify different weight values for a real server of type **host** in a server farm, you can assign multiple IP addresses to the server. You can also use the same IP address of a real server with different port numbers.

The syntax of this command is as follows:

weight *number*

The *number* argument is the weight value assigned to a real server in a server farm. Enter an integer from 1 to 100. The default is 8.

For example, enter:

```
host1/Admin(config-rserver-host)# weight 50
```

To reset the configured weight of a real server to the default value of 8, enter:

```
host1/Admin(config-rserver-host)# no weight
```

Placing a Real Server in Service

Before you can start sending connections to a real server, you must place the server in service. You can place a real server in service by using the **inservice** command in either real server host or real server redirect configuration mode. The syntax of this command is as follows:

inservice

Note Before you can place a real server that you have created in service, you must configure a minimum of an IP address for a server of type **host** or a webhost relocation string for a server of type **redirect**.

For example, to place a real server in service, enter:

```
host1/Admin(config-rserver-host)# inservice
```

Managing Real Servers

If a primary real server fails, the ACE takes that server out of service and no longer includes it in load-balancing decisions. If you configured a backup server for the real server that failed, the ACE redirects the primary real server connections to the backup server. For information about configuring a backup server, see the "Configuring a Backup Server for a Real Server" section.

The ACE can take a real server out of service for the following reasons:

- Probe failure
- ARP timeout
- Entering the **no inservice** command (see the "Gracefully Shutting Down a Server" section

- Entering the **inservice standby** command (see the "Gracefully Shutting Down a Server with Sticky Connections" section)

For servers with sticky connections, the ACE removes those sticky entries from the sticky database when it takes the server out of service. For more information about stickiness, see Chapter 5, Configuring Stickiness.

Use the **failaction purge** command to instruct the ACE to purge Layer 3 and Layer 4 connections if a real server fails. The default behavior of the ACE is to do nothing with existing connections if a real server fails. You can also enter the **failaction reassign** command, which instructs the ACE to send existing connections to a backup server (if configured) in case the primary server fails. For details about the **failaction** command, see the "Configuring the ACE Action when a Server Fails" section. For Layer 7 connections, you can instruct the ACE to rebalance each HTTP request by entering the **persistence-rebalance** command. For details about the **persistence-rebalance** command, see the "Configuring HTTP Persistence Rebalance" section in Chapter 3, Configuring Traffic Policies for Server Load Balancing.

You can also manually take a primary real server out of service gracefully using either the **no inservice** command or the **inservice standby** command. Both commands provide graceful shutdown of a server. Use the **no inservice** command when you do not have stickiness configured and you need to take a server out of service for maintenance or a software upgrade. The **no inservice** command instructs the ACE to do the following:

- Tear down existing non-TCP connections to the server
- Allow existing TCP connections to complete
- Disallow any new connections to the server

With sticky configured, use the **inservice standby** command to gracefully take a primary real server out of service. The **inservice standby** command instructs the ACE to do the following:

- Tear down existing non-TCP connections to the server
- Allow current TCP connections to complete
- Allow new sticky connections for existing server connections that match entries in the sticky database
- Load balance all new connections (other than the matching sticky connections mentioned above) to the other servers in the server farm
- Eventually take the server out of service

Gracefully Shutting Down a Server

You can shut down a real server gracefully by using the **no inservice** command in either real server host or real server redirect configuration mode. This command causes the ACE to reset all non-TCP connections to the server. For TCP connections, existing flows are allowed to complete before the ACE takes the real server out of service. No new connections are allowed.

Note The ACE resets all Secure Sockets Layer (SSL) connections to a particular real server when you enter the **no inservice** command for that server.

The syntax of this command is as follows:

no inservice

For example, to gracefully shut down a real server (for example, for maintenance or software upgrades), enter:

```
host1/Admin(config-rserver-host)# no inservice
```

Examples of Real Server Configurations

The following examples illustrate a running configuration that creates multiple real servers. These configurations show how to specify a host as the real server and how to specify a real server to redirect traffic to a different location. The real server configurations appear in bold in the following two examples:

- Real Server that Hosts Content
- Real Server that Redirects Client Requests

Real Server that Hosts Content

The following example creates three real servers, places them in service, and adds each real server to a server farm.

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

rserver host SERVER1
  ip address 192.168.252.245
  inservice
```

```
rserver host SERVER2
  ip address 192.168.252.246
  inservice
rserver host SERVER3
  ip address 192.168.252.247
  inservice
serverfarm host SFARM1
  probe HTTP_PROBE
  predictor roundrobin
  rserver SERVER1
     weight 10
     inservice
  rserver SERVER2
     weight 20
     inservice
  rserver SERVER3
     weight 30
     inservice
```

Real Server that Redirects Client Requests

The following example specifies a series of real servers that redirect all incoming connections that match URLs ending in /redirect-100k.html, /redirect-10k.html, and /redirect-1k.html to the appropriate server farm.

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

rserver redirect SERVER1
  webhost-redirection http://192.168.120.132/redirect-100k.html 301
  inservice
rserver redirect SERVER2
  webhost-redirection http://192.168.120.133/redirect-10k.html 301
  inservice
rserver redirect SERVER3
  webhost-redirection http://192.168.120.134/redirect-1k.html 301
  inservice
serverfarm redirect SFARM1
    rserver SERVER1
    inservice
serverfarm redirect SFARM2
    rserver SERVER2
    inservice
serverfarm redirect SFARM3
    rserver SERVER3
    inservice
```

Configuring a Server Farm

This section describes server farms and how to configure them. It contains the following topics:

- Server Farm Overview
- Server Farm Configuration Quick Start
- Creating a Server Farm
- Configuring a Description of a Server Farm
- Configuring the ACE Action when a Server Fails
- Associating Multiple Health Probes with a Server Farm
- Configuring AND Logic for Server Farm Probes
- Configuring the Server Farm Predictor Method
- Configuring Server Farm HTTP Return Code Checking
- Configuring a Partial Server Farm Failover
- Associating a Real Server with a Server Farm
- Configuring Additional Real Server Attributes in a Server Farm
- Configuring a Backup Server Farm
- Specifying No NAT
- Configuring Asymmetric Server Normalization
- Example of a Server Farm Configuration

Server Farm Overview

Server farms are groups of networked real servers that contain the same content and that typically reside in the same physical location in a data center. Web sites often comprise groups of servers configured in a server farm. Load-balancing software distributes client requests for content or services among the real servers based on the configured policy and traffic classification, server availability and load, and other factors. If one server goes down, another server can take its place and continue to provide the same content to the clients who requested it.

Server Farm Configuration Quick Start

Table 2-2 provides a quick overview of the steps required to configure server farms. 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 2-2.

Table 2-2 Server Farm 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. Change to, or directly log in to, the correct context if necessary.
`host1/Admin#` **`changeto C1`**
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.
`host/Admin#` **`config`**
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

3. Configure a server farm.
`host1/Admin(config)#` **`serverfarm SF1`**
`host1/Admin(config-sfarm-host)#`

4. Associate one or more health probes of the same or different protocols with the server farm. Reenter the command as necessary to associate additional probes with the server farm. `host1/Admin(config-sfarm-host)#` **`probe PROBE1`**
5. Configure the server-farm predictor (load-balancing) method. `host1/Admin(config-sfarm-host)#` **`predictor leastconns`**
6. Configure HTTP return code checking for a server farm. `host1/Admin(config-sfarm-host)#` **`retcode 200 500 check count`**

Table 2-2 Server Farm Configuration Quick Start (continued)

Task and Command Example
7. (Optional) If you do not want the ACE to use NAT to translate the VIP to the server IP address, enter: `host1/Admin(config-sfarm-host)#` **`transparent`**

8. Associate an existing real server with the server farm and enter server farm host real server configuration mode.
`host1/Admin(config-sfarm-host)#` **`rserver SERVER1 3000`**
`host1/Admin(config-sfarm-host-rs)#`

9. (Optional) Configure a weight for the real server associated with the server farm. The ACE uses this weight value in the weighted round-robin and least-connections predictor methods. If you do not configure a weight, the ACE uses the global weight configured for the real server in real server configuration mode. `host1/Admin(config-sfarm-host-rs)#` **`weight 50`**
10. Configure a backup server in case the real server becomes unavailable. The backup server can function as a sorry server. `host1/Admin(config-sfarm-host-rs)#` **`backup rserver SERVER2 4000`**
11. Configure one or more probes of the same or different protocols to monitor the health of the real server in the server farm. `host1/Admin(config-sfarm-host-rs)#` **`probe PROBE1_TCP`**
12. Configure connection limits for the real server in a server farm. `host1/Admin(config-sfarm-host-rs)#` **`conn-limit max 5000 min 4000`**

13. Place the real server in service.
`host1/Admin(config-sfarm-host-rs)#` **`inservice`**
`host1/Admin(config-sfarm-host-rs)#` **`Ctrl-z`**

14. (Recommended) Use the following command to display the server farm configuration. Make any modifications to your configuration as necessary, then reenter the command to verify your configuration changes. `host1/Admin#` **`show running-config serverfarm`**
15. (Optional) Save your configuration changes to flash memory. `host1/Admin#` **`copy running-config startup-config`**

Creating a Server Farm

You can create a new server farm or modify an existing server farm and enter the server-farm configuration mode by using the **serverfarm** command in configuration mode. You can configure a maximum of 16,384 server farms on each ACE.

The syntax of this command is as follows:

serverfarm [**host** | **redirect**] *name*

The keywords, arguments, and options are as follows:

- **host**—(Optional, default) Specifies a typical server farm that consists of real servers that provide content and services to clients and accesses server farm host configuration mode.
- **redirect**—(Optional) Specifies that the server farm consists only of real servers that redirect client requests to alternate locations specified by the relocation string in the real server configuration and accesses server farm redirect configuration mode. See the "Configuring a Real Server Relocation String" section. You can use a redirect server farm as a sorry server farm by specifying it as a backup server farm of type **redirect**. For details, see the "Configuring a Sorry Server Farm" section in Chapter 3, Configuring Traffic Policies for Server Load Balancing.
- *name*—Unique identifier of the server farm. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a server farm of type **host** called SF1, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)#
```

For example, to create a server farm of type **redirect** called SF2, enter:

```
host1/Admin(config)# serverfarm redirect SF2
host1/Admin(config-sfarm-redirect)#
```

After you create a server farm, you can configure the other server farm attributes and add real servers to it as described in the following sections:

- Configuring a Description of a Server Farm
- Configuring the ACE Action when a Server Fails
- Associating Multiple Health Probes with a Server Farm

- Configuring AND Logic for Server Farm Probes
- Configuring the Server Farm Predictor Method
- Configuring Server Farm HTTP Return Code Checking
- Configuring a Partial Server Farm Failover
- Associating a Real Server with a Server Farm
- Specifying No NAT
- Configuring Asymmetric Server Normalization

Configuring a Description of a Server Farm

You can provide a text description of a server farm by using the **description** command in either server farm host or server farm redirect configuration mode. The syntax of this command is as follows:

description *text*

For the *text* argument, enter a server farm description with a maximum of 240 alphanumeric characters.

For example, to enter a description of a server farm, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# description CURRENT EVENTS ARCHIVE
```

To remove the description of a server farm, enter:

```
host1/Admin(config-sfarm-host)# no description
```

Configuring the ACE Action when a Server Fails

You can set the action that the ACE takes with respect to connections if any real server fails in a server farm. You can configure the failure action by using the **failaction** command in either server farm host or server farm redirect configuration mode.

The syntax of this command is as follows:

failaction {**purge** | **reassign** [**across-interface**]}

The keywords and options are as follows:

- **purge**—Instructs the ACE to remove all connections to a real server if that real server in the server farm fails after you enter this command. The module sends a reset (RST) to both the client and the server that failed.
- **reassign**—Instructs the ACE to reassign the existing server connections to the backup real server (if configured) on the same VLAN interface if the real server fails after you enter this command. If a backup real server has not been configured for the failing server, this keyword has no effect and leaves the existing connections untouched in the failing real server.

  Follow these configuration requirements and restrictions when you use the **reassign** keyword:

  - Enable the **transparent** command (see the "Specifying No NAT"section) to instruct the ACE not to use NAT to translate the ACE VIP address to the server IP address. The **failaction reassign** command is intended for use in stateful firewall load balancing (FWLB) on your ACE, where the destination IP address for the connection coming in to the ACE is for the end-point real server, and the ACE reassigns the connection so that it is transmitted through a different next hop. See Chapter 6, Configuring Firewall Load Balancing for details.
  - Configure the **mac-sticky** command on all server-side interfaces to ensure that packets that are going to and coming from the same server in a flow will traverse the same firewalls or stateful devices.
  - Configure the **predictor hash address** command under firewall server farms.
- **across-interface**—(Optional) Instructs the ACE to reassign all connections from the failed real server to a backup real server on a different VLAN that is commonly referred to as a bypass VLAN. By default, this feature is disabled. See Figure 2-1.

  Follow these configuration requirements and restrictions when you use the **across-interface** option:

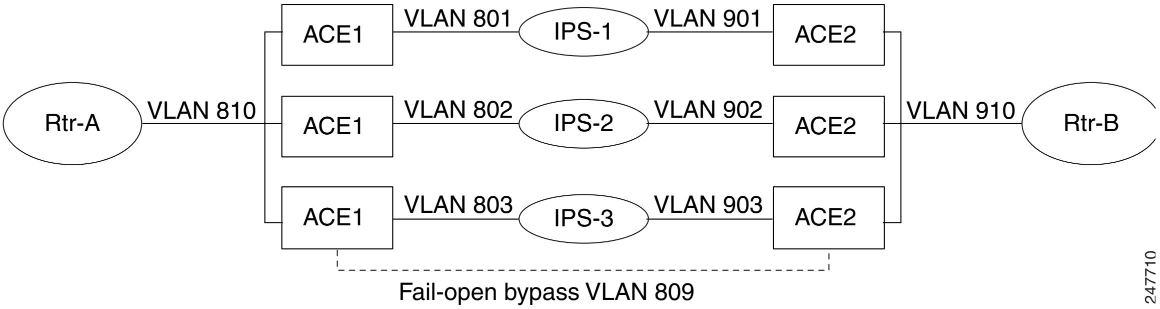
  - You must configure identical policies on the primary interface and the backup-server interface. The backup interface must have the same feature configurations as the primary interface.

- If you configure a policy on the backup-server interface that is different from the policies on the primary-server interface, that policy will be effective only for new connections. The reassigned connection will always have only the primary-server interface policies.
- Interface-specific features (for example, NAT, application protocol inspection, outbound ACLs, or SYN cookie) are not supported.
- You cannot reassign connections to the failed real server after it comes back up. This restriction also applies to same-VLAN backup servers.
- You must connect real servers directly to the ACE. This requirement also applies to same-VLAN backup servers.
- You must disable sequence number randomization on the firewall.
- Probe configurations should be similar on both ACEs and the interval values should be low. For example, if you configure a high interval value on ACE1 and a low interval value on ACE2, the reassigned connections may become stuck because of the probe configuration mismatch. ACE2 with the low interval value will detect the primary server failure first and will reassign all its incoming connections to the backup-server interface VLAN. ACE1 with the high interval value may not detect the failure before the primary server comes back up and will still point to the primary server.

To minimize packet loss, we recommend the following probe parameter values on both ACEs:

- Interval: 2
- Faildetect: 2
- Passdetect interval: 2
- Passdetect count: 5

Figure 2-1 Example of a Bypass VLAN Topology



In the absence of the **failaction** command, the default behavior of the ACE is to take a failed real server out of load-balancing rotation for new connections and to allow existing connections to complete. In this case, the ACE does not send the connections to a backup real server in the server farm if one is configured. You must configure the **failaction reassign** command to cause that behavior.

The backup server farm behavior is not affected by either the presence or the absence of the **failaction** command. If all real servers in the server farm fail, then the backup server farm (if configured) automatically takes over the failed server farm connections.

Note To clear connections to real servers that have failed before you entered the **failaction** command, use the **clear conn** command. For details about the **clear conn** command, see the "Clearing Real Server Connections" section.

For example, to remove connections to a real server in a server farm that fail after you enter this command, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# failaction purge
```

For example, to specify that the ACE reassign the existing real server connections to the backup real server on a different VLAN interface if the real server fails, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# failaction reassign across-interface
host1/Admin(config-sfarm-host)# transparent
```

For example, to specify that the ACE reassign the existing real server connections to the backup real server on the same VLAN interface if the real server fails, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# failaction reassign
host1/Admin(config-sfarm-host)# transparent
```

To reset the behavior of the ACE to the default of taking no action if a real server fails, enter:

```
host1/Admin(config-sfarm-host)# no failaction
```

Example of a Bypass VLAN Configuration

The following configurations show how to configure two ACEs and the intrusion protection system (IPS) firewall located between them for a bypass VLAN for use with the **failaction reassign across-interface** command as shown in Figure 2-1.

ACE1 Configuration

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

probe icmp PING
  interval 2
  faildetect 2
  passdetect interval 2
  passdetect count 2
  receive 1

parameter-map type connection TCP
  set timeout inactivity 86400
parameter-map type connection UDP
  set timeout inactivity 60
parameter-map type connection ICMP
  set timeout inactivity 30

rserver host REAL.BYPASS_V809
  ip address 192.168.9.11
  inservice
```

```
rserver host REAL.311_V901
  ip address 192.168.1.11
  inservice
rserver host REAL.312_V902
  ip address 192.168.2.11
  inservice
rserver host REAL.313_V903
  ip address 192.168.3.11
  inser

serverfarm host SF
  transparent
  failaction reassign across-interface
  predictor hash address 255.255.255.255
  probe PING
  rserver REAL.BYPASS_V809
    inservice standby
  rserver REAL.311_V901
    backup-rserver REAL.BYPASS_V809
    inservice
  rserver REAL.312_V902
    backup-rserver REAL.BYPASS_V809
    inservice
  rserver REAL.313_V903
    backup-rserver REAL.BYPASS_V809
    inservice

class-map match-any NET
  2 match virtual-address 0.0.0.0 0.0.0.0 any
class-map match-any ANY
  2 match any
class-map match-any ANY-TCP
  2 match port tcp any
class-map match-any ANY-UDP
  2 match port udp any
class-map type management match-any ICMP
  description Ping Interfaces
  10 match protocol icmp any
class-map type management match-any SYSTEM.MANAGE.ADMINISTRATION
  description Administrative access traffic
  10 match protocol ssh any
  30 match protocol icmp any

policy-map type management first-match SYSTEM.MANAGE.ICMP
  class ICMP
    permit
policy-map type management first-match SYSTEM.MANAGE.ADMINISTRATION
  class SYSTEM.MANAGE.ADMINISTRATION
    permit
```

```
policy-map type loadbalance first-match SYSTEM.FORWARD
  class class-default
    forward
policy-map type loadbalance first-match VIP
  class class-default
    serverfarm SF
policy-map multi-match GLOBAL.NORMALIZE
  class ANY-TCP
    connection advanced-options TCP
  class ANY-UDP
    connection advanced-options UDP
  class class-default
    connection advanced-options ICMP

policy-map multi-match Vlan801.loadbalance
  class NET
    loadbalance vip inservice
    loadbalance policy SYSTEM.FORWARD
policy-map multi-match VLAN802.LOADBALANCE
  class NET
    loadbalance vip inservice
    loadbalance policy SYSTEM.FORWARD
policy-map multi-match VLAN803.LOADBALANCE
  class NET
    loadbalance vip inservice
    loadbalance policy SYSTEM.FORWARD
policy-map multi-match VLAN910.LOADBALANCE
  class NET
    loadbalance vip inservice
    loadbalance policy VIP
policy-map multi-match VLAN809.LOADBALANCE
  class NET
    loadbalance vip inservice
    loadbalance policy SYSTEM.FORWARD

service-policy input GLOBAL.NORMALIZE

interface vlan 801
  ip address 192.168.1.1 255.255.255.0
  mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input VLAN801.LOADBALANCE
  no shutdown
```

```
interface vlan 802
  ip address 192.168.2.1 255.255.255.0
  mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input VLAN802.LOADBALANCE
  no shutdown
interface vlan 803
  ip address 192.168.3.1 255.255.255.0
  mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input VLAN803.LOADBALANCE
  no shutdown
interface vlan 910
  ip address 192.168.8.1 255.255.255.0
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ADMINISTRATION
  service-policy input VLAN808.LOADBALANCE
  no shutdown
interface vlan 809
  ip address 192.168.9.1 255.255.255.0
  mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input VLAN809.LOADBALANCE
  no shutdown

  ip route 0.0.0.0 0.0.0.0 192.168.0.0
```

ACE2 Configuration

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

probe icmp PING
  interval 2
  faildetect 2
  passdetect interval 2
  passdetect count 5
  receive 1
```

```
parameter-map type connection TCP
  set timeout inactivity 86400
parameter-map type connection UDP
  set timeout inactivity 60
parameter-map type connection ICMP
  set timeout inactivity 30

rserver host REAL.BYPASS_V809
  ip address 192.168.9.1
  inservice
rserver host REAL.TP-INC-311_V801
  ip address 192.168.1.1
  inservice
rserver host REAL.TP-INC-312_V802
  ip address 192.168.2.1
  inservice
rserver host REAL.TP-INC-313_V803
  ip address 192.168.3.1
  inservice

serverfarm host SF
  transparent
  failaction reassign across-interface
  predictor hash address 255.255.255.255
  probe PING
  rserver REAL.BYPASS_V809
    inservice standby
  rserver REAL.TP-INC-311_V801
    backup-rserver REAL.BYPASS_V809
    inservice
  rserver REAL.TP-INC-312_V802
    backup-rserver REAL.BYPASS_V809
    inservice
  rserver REAL.TP-INC-313_V803
    backup-rserver REAL.BYPASS_V809
    inservice

class-map match-any NET
  2 match virtual-address 0.0.0.0 0.0.0.0 any
class-map match-any ANY
  2 match any
class-map match-any ANY-TCP
  2 match port tcp any
class-map match-any ANY-UDP
  2 match port udp any
class-map type management match-any ICMP
  description Ping Interfaces
  10 match protocol icmp any
```

```
class-map type management match-any SYSTEM.MANAGE.ADMINISTRATION
  description Administrative access traffic
  10 match protocol ssh any
  30 match protocol icmp any

policy-map type management first-match SYSTEM.MANAGE.ICMP
  class ICMP
    permit
policy-map type management first-match SYSTEM.MANAGE.ADMINISTRATION
  class SYSTEM.MANAGE.ADMINISTRATION
    permit

policy-map type loadbalance first-match SYSTEM.FORWARD
  class class-default
    forward
policy-map type loadbalance first-match VIP
  class class-default
    serverfarm SF

policy-map multi-match GLOBAL.NORMALIZE
  class ANY-TCP
    connection advanced-options TCP
  class ANY-UDP
    connection advanced-options UDP
  class class-default
    connection advanced-options ICMP

policy-map multi-match VLAN810.LOADBALANCE
  class NET
    loadbalance vip inservice
    loadbalance policy VIP

service-policy input GLOBAL.normalize

rserver host RS1
 ip address 192.168.10.111
 ins

rserver host RS2
 ip address 192.168.10.112
 ins

serverfarm host SF1
 rserver RS1
 ins
 rserver RS2
 ins
```

```
policy-map type loadbalance first-match LB
 class class-default
  serverfarm SF1

policy-map multi-match L4
 class NET
 loadbalance vip inservice
 loadbalance vip icmp
 loadbalance policy LB

interface vlan 809
  ip address 192.168.9.11 255.255.255.0
   mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input l4
  no shutdown
interface vlan 810
  ip address 192.168.10.1 255.255.255.0
   mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ADMINISTRATION
  service-policy input VLAN810.LOADBALANCE
  no shutdown
interface vlan 901
  ip address 192.168.1.11 255.255.255.0
  mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input L4
  no shutdown
interface vlan 902
  ip address 192.168.2.11 255.255.255.0
  mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input L4
  no shutdown
```

```
interface vlan 903
  ip address 192.168.3.11 255.255.255.0
  mac-sticky enable
  mac-address autogenerate
  access-group input ANY
  access-group output ANY
  service-policy input SYSTEM.MANAGE.ICMP
  service-policy input L4
  no shutdown

ip route 0.0.0.0 0.0.0.0 192.168.0.0
```

Firewall (IPS) Configuration

```
interface Vlan801

interface Vlan802

interface Vlan803

interface Vlan901

interface Vlan902

interface Vlan903

context 1
  member default
  allocate-interface Vlan801
  allocate-interface Vlan901
  config-url disk:/1.cfg

context 2
  member default
  allocate-interface Vlan802
  allocate-interface Vlan902
  config-url disk:/2.cfg

context 3
  member default
  allocate-interface Vlan803
  allocate-interface Vlan903
  config-url disk:/3.cfg
```

Context 1 Configuration

```
firewall transparent

interface Vlan801
 nameif OUT
 bridge-group 1
 security-level 0

interface Vlan901
 nameif IN
 bridge-group 1
 security-level 100

interface BVI1
 ip address 192.168.1.10 255.255.255.0

access-list IP extended permit ip any any
access-group IP in interface OUT
access-group IP out interface OUT
access-group IP in interface IN
access-group IP out interface IN

class-map BYPASS
 match access-list IP
policy-map TCPBYPASS
 class BYPASS
  set connection advanced-options TCP-STATE-BYPASS

service-policy TCPBYPASS interface out
service-policy TCPBYPASS interface in
```

Configure contexts 2 and 3 in a similar manner.

Associating Multiple Health Probes with a Server Farm

You can associate multiple health probes of the same or different protocols with each server farm. Each server farm supports the following probe types:

- DNS
- ECHO (TCP and UDP)
- Finger
- FTP
- HTTP
- HTTPS
- ICMP
- IMAP
- POP/POP3
- RADIUS
- RTSP
- Scripted
- SIP
- SMTP
- SNMP
- TCP
- Telnet
- UDP

By default, the real servers that you configure in a server farm inherit the probes that you configure directly on that server farm. Multiple probes that you configure on a server farm have an OR logic associated with them. If one of the probes configured on the server farm fails, all real servers configured in the server farm fail and enter the PROBE-FAILED state. You can also configure AND logic for server farm probes. For details, see the "Configuring AND Logic for Server Farm Probes" section.

You can associate a probe with a server farm by using the **probe** command in server farm host configuration mode only. The syntax of this command is as follows:

probe *name*

For the *name* argument, enter the name of an existing probe as a text string with no spaces and a maximum of 64 alphanumeric characters. For information about creating and configuring probes, see Chapter 4, Configuring Health Monitoring.

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# probe PROBE1
```

To remove the probe from the server farm configuration, enter:

```
host1/Admin(config-sfarm-host)# no probe PROBE1
```

Configuring AND Logic for Server Farm Probes

By default, real servers that you configure in a server farm inherit the probes that you configure directly on that server farm. When you configure multiple probes on a server farm, the real servers in the server farm use an OR logic with respect to the probes. This means that if one of the probes configured on the server farm fails, all the real servers in that server farm fail and enter the PROBE-FAILED state. For information about creating and configuring probes, see Chapter 4, Configuring Health Monitoring.

To configure the real servers in a server farm to use AND logic with respect to multiple server farm probes, use the **fail-on-all** command in server farm host configuration mode. With AND logic, if one server farm probe fails, the real servers in the server farm remain in the OPERATIONAL state. If all the probes associated with the server farm fail, then all the real servers in that server farm fail and enter the PROBE-FAILED state. This command applies to all probe types. You can also configure AND logic for probes that you configure directly on real servers in a server farm. For more information, see the "Configuring AND Logic for Real Server Probes in a Server Farm" section.

The syntax of this command is as follows:

fail-on-all

For example, to configure the SERVER1 and SERVER2 real servers in the SFARM1 server farm to remain in the OPERATIONAL state unless all server farm probes fail, enter:

```
host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# probe HTTP_PROBE
host1/Admin(config-sfarm-host)# probe ICMP_PROBE
host1/Admin(config-sfarm-host)# fail-on-all
host1/Admin(config-sfarm-host)# rserver server1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver SERVER2
host1/Admin(config-sfarm-host-rs)# inservice
```

If either HTTP_PROBE or ICMP_PROBE fails, SERVER1 and SERVER2 remain in the OPERATIONAL state. If both probes fail, both real servers fail and enter the PROBE-FAILED state.

To remove the AND probe logic from the server farm, enter:

```
host1/Admin(config-sfarm-host)# no fail-on-all
```

Configuring the Server Farm Predictor Method

The load-balancing (predictor) method that you configure determines how the ACE selects a real server in a server farm to service a client request. You can configure the load-balancing method by using the **predictor** command in either server farm host or server farm redirect configuration mode. The syntax of this command is as follows:

predictor {**hash** {**address** [**destination** | **source**] [*netmask*]} | {**content** [**offset** *number1*] [**length** *number2*] [**begin-pattern** *expression1*] [**end-pattern** *expression2*]} | {**cookie** *name1*} | {**header** *name2*} | {**layer4-payload** [**begin-pattern** *expression3*] [**end-pattern** *expression4*] [**length** *number3*] [**offset** *number4*]} | {**url** [**begin-pattern** *expression5*] [**end-pattern** *expression6*]}} | {**least-bandwidth** [**assess-time** *seconds*] [**samples** *number5*]} | {**leastconns** [**slowstart** *time*]} | {**least-loaded probe** *name3*} | {**response** {**app-req-to-resp** | **syn-to-close** | **syn-to-synack**}[**samples** *number7*]} | {**roundrobin**}

Note The hash predictor methods do not recognize the weight value that you configure for real servers. The ACE uses the weight that you assign to real servers only in the least-connections, application-response, and round-robin predictor methods.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- Configuring the Hash Address Predictor
- Configuring the Hash Content Predictor
- Configuring the Hash Cookie Predictor
- Configuring the Hash Header Predictor
- Configuring the Hash Layer 4 Payload Predictor
- Configuring the Hash URL Predictor
- Configuring the Least-Bandwidth Predictor Method
- Configuring the Least-Connections Predictor
- Configuring the Least-Loaded Predictor
- Configuring the Application Response Predictor
- Configuring the Round-Robin Predictor

Configuring the Hash Address Predictor

To instruct the ACE to select a real server using a hash value based on the source or destination IP address, use the **predictor hash address** command in server farm host or redirect configuration mode. Use this command when you configure firewall load balancing (FWLB). For more information about FWLB, see Chapter 6, Configuring Firewall Load Balancing. The syntax of this command is as follows:

predictor hash address [**destination** | **source**] [*netmask*]

The keywords and options are as follows:

- **source**—(Optional) Selects the server using a hash value based on the source IP address.
- **destination**—(Optional) Selects the server using a hash value based on the destination IP address.
- *netmask*—(Optional) Bits in the IP address to use for the hash. If not specified, the default mask is 255.255.255.255.

For example, to configure the ACE to load balance client requests based on a hash value of the source address, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash address source
255.255.255.0
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Content Predictor

To instruct the ACE to select a real server using a hash value based on a content string of the HTTP packet body, use the **predictor hash content** command in server farm host or redirect configuration mode. The syntax of this command is as follows:

predictor hash content [**offset** *number1*] [**length** *number2*] [**begin-pattern** *expression1*] [**end-pattern** *expression2*]

The keywords and options are as follows:

- **offset** *number1*—(Optional) Specifies the portion of the content that the ACE uses to stick the client on a particular server by indicating the bytes to ignore starting with the first byte of the payload. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the content.
- **length** *number2*—(Optional) Specifies the length of the portion of the content (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is infinity.

  Note You cannot specify both the **length** and the **end-pattern** options in the same **content** command.

  The offset and length can vary from 0 to 1000 bytes. If the payload is longer than the offset but shorter than the offset plus the length of the payload, the ACE sticks the connection based on that portion of the payload starting with the byte after the offset value and ending with the byte specified by the offset plus the length. The total of the offset and the length cannot exceed 1000.
- **begin-pattern** *expression1*—(Optional) Specifies the beginning pattern of the content string and the pattern string to match before hashing. If you do not specify a beginning pattern, the ACE starts parsing the HTTP body immediately following the offset byte. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.)

  Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). The ACE supports the use of regular expressions for matching string expressions. See Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing, for a list of the supported characters that you can use for matching string expressions.

Note When matching data strings, note that 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 (?).

- **end-pattern** *expression2*—(Optional) Specifies the pattern that marks the end of hashing. If you do not specify either a length or an end pattern, the ACE continues to parse the data until it reaches the end of the field or the end of the packet, or reaches the maximum body parse length. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks ("). The ACE supports the use of regular expressions for matching string expressions. See Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing, for a list of the supported characters that you can use for matching string expressions.

Note You cannot specify both the **length** and the **end-pattern** options in the same **predictor hash content** command.

For example, to configure the ACE to load balance client requests based on a hash value of the content string of the HTTP packet body, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash content offset 25
length 32 begin-pattern abc123*
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Cookie Predictor

To instruct the ACE to select a real server using the hash value of a cookie name or based on the name in the cookie name of the URL query string, use the **predictor hash cookie** command in server farm host or server farm redirect configuration mode. The syntax of this command is as follows:

predictor hash cookie [**secondary**] *name*

The option and argument are as follows:

- secondary—(Optional) Selects the server by using the hash value based on the specified name in the cookie name in the URL query string, not the cookie header. If you do not include this option, the ACE selects a real server using the hash value of the cookie name.
- *name*—Cookie name. Enter a cookie name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

When configuring the **secondary** option, consider the following request example:

GET /index.html?TEST=test123
Cookie: TEST=456

If you configure the **predictor hash cookie secondary TEST** command, it selects the server using the hash value based on test123. If you configure the **predictor hash cookie TEST** command, it selects the server using the hash value based on test456.

The **secondary** option allows the ACE to correctly load balance in cases when the query string identifies the actual resource, instead of the URL. In the following example, if the ACE hashes on the URL, it would load balance on the same real server:

http://youtube.com/watch?v=C16mk4OfcuM
http://youtube.com/watch?v=cJ3jPzs2NLk

For example, to configure the ACE to load balance client requests based on a hash value of the cookie name *corvette*, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash cookie corvette
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Header Predictor

To instruct the ACE to select a real server using the hash of an HTTP header value, use the **predictor hash header** command in server farm host or redirect configuration mode. The syntax of this command is as follows:

predictor hash header *name*

For the *name* argument, enter a header name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters, or enter one of the following standard HTTP headers:

- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Authorization
- Cache-Control
- Connection
- Content-MD5
- Expect
- From
- Host
- If-Match
- Pragma
- Referrer
- Transfer-Encoding
- User-Agent
- Via

For example, to configure the ACE to load balance client requests based on the hash of the Host header value, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash header Host
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Layer 4 Payload Predictor

To instruct the ACE to select a real server based on a hash value of a Layer 4 generic protocol payload, use the **predictor hash layer4-payload** command in server farm host or redirect configuration mode. Use this predictor to load balance packets from protocols that are not explicitly supported by the ACE. The syntax is as follows:

predictor hash layer4-payload [**begin-pattern** *expression3*] [**end-pattern** *expression4*] [**length** *number3*] [**offset** *number4*]

The keywords, options, and arguments are as follows:

- **begin-pattern** *expression3*—(Optional) Specifies the beginning pattern of the Layer 4 payload and the pattern string to match before hashing. If you do not specify a beginning pattern, the ACE starts parsing the HTTP body immediately following the offset byte. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). The ACE supports the use of regular expressions for matching string expressions. See Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing for a list of the supported characters that you can use for matching string expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use the “[]” 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 (?).

- **end-pattern** *expression4*—(Optional) Specifies the pattern that marks the end of hashing. If you do not specify either a length or an end pattern, the ACE continues to parse the data until it reaches the end of the field or the end of the packet, or until it reaches the maximum body parse length. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.)

  Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks ("). The ACE supports the use of regular expressions for matching string expressions. See Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing for a list of the supported characters that you can use for matching string expressions.

  Note You cannot specify both the **length** and the **end-pattern** options in the same **hash layer4-payload** command.

- **length** *number3*—(Optional) Specifies the length of the portion of the payload (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is the entire payload.

  Note You cannot specify both the **length** and the **end-pattern** options in the same **hash layer4-payload** command.

  The offset and length can vary from 0 to 1000 bytes. If the payload is longer than the offset but shorter than the offset plus the length of the payload, the ACE sticks the connection based on that portion of the payload starting with the byte after the offset value and ending with the byte specified by the offset plus the length. The total of the offset and the length cannot exceed 1000.

- **offset** *number4*—(Optional) Specifies the portion of the payload that the ACE uses to load balance the client request to a particular server by indicating the bytes to ignore starting with the first byte of the payload. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the payload.

For example, to configure the ACE to load balance client requests based on a hash value of a Layer 4 frame payload, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash layer4-payload
begin-pattern abc123* length 250 offset 25
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash URL Predictor

To instruct the ACE to select a real server using a hash value based on the requested URL, use the **predictor hash url** command in server farm host or redirect configuration mode. Use this predictor method to load balance cache servers. Cache servers perform better with the URL hash method because you can divide the contents of the caches evenly if the traffic is random enough. In a redundant configuration, the cache servers continue to work even if the active ACE switches over to the standby ACE. For information about configuring redundancy, see the *Cisco Application Control Engine Module Administration Guide*.

The syntax of this command is as follows:

predictor hash url [**begin-pattern** *expression1*] [**end-pattern** *expression2*]

The keywords, options, and arguments are as follows:

- **begin-pattern** *expression1*—(Optional) Specifies the beginning pattern of the URL and the pattern string to match before hashing. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. If you want to match a URL that contains spaces, you must use "\x20" (without the quotation marks) for each space character.

- **end-pattern** *expression2*—(Optional) Specifies the pattern that marks the end of hashing. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. If you want to match a URL that contains spaces, you must use "\x20" (without the quotation marks) for each space character.

For example, to configure the ACE to load balance client requests based on a hash value of a URL in the client HTTP GET request, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash url end-pattern
.cisco.com
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Least-Bandwidth Predictor Method

To instruct the ACE to select the real server that processed the least amount of network traffic based on the average bandwidth that the server used over a specified number of samples, use the **predictor least-bandwidth** command in server farm host or redirect configuration mode. Use this predictor for heavy traffic use, such as downloading a video clip. This predictor is considered adaptive because the ACE continuously provides feedback to the load-balancing algorithm based on the behavior of the real server.

The ACE measures traffic statistics between itself and the real servers in a server farm in both directions, calculates the bandwidth for each server, and updates the bandwidth for each server every second. It then averages the bandwidth for each server over the number of configured samples. From the averaged bandwidth results, the ACE creates an ordered list of real servers and selects the server that used the least amount of bandwidth. The ACE continues to use the ordered list until the next bandwidth assessment period begins. Then the ACE updates the ordered list with the new values obtained from the most recent assess-time interval and the process starts over again.

The syntax of this command is as follows:

predictor least-bandwidth [**assess-time** *seconds*] [**samples** *number5*]

The keywords and arguments are as follows:

- **assess-time** *seconds*—(Optional) Specifies the frequency with which the ACE reevaluates the server load based on the traffic bandwidth generated by the server and updates the ordered list of servers. Enter an integer from 1 to 10 seconds. The default is 2 seconds.
- **samples** *number5*—(Optional) Specifies the maximum number of samples over which the ACE averages the bandwidth measurements and calculates the final load values for each server. Enter an integer from 1 to 16. Each value must be a power of 2, so the valid values are as follows: 1, 2, 4, 8, and 16. The default is 8.

Note Because the ACE uses the same ordered list of servers during any one assess-time period, it load balances all new connections during that time period to only one server (the one with the lowest average bandwidth use). We recommend that you use this predictor for long-lived and steady traffic, such as downloading video. Applying this predictor to random or bursty short-lived traffic may cause unpredictable load-balancing results.

For example, to configure the ACE to select a real server based on the amount of bandwidth that the server used over a specified number of samples, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor least-bandwidth samples 4
assess-time 6
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Least-Connections Predictor

To instruct the ACE to select the server with the fewest number of connections based on the server weight, use the **predictor leastconns** command in server farm host or redirect configuration mode. Use this predictor for processing light user requests (for example, browsing simple static web pages). For information about setting the server weight, see the "Configuring a Real Server Weight" section and the "Configuring the Weight of a Real Server in a Server Farm" section.

Note Server weights take effect only when there are open connections to the servers. When there are no sustained connections to any of the servers, the least-connections predictor method behaves the same way as the round-robin method.

The syntax of this command is as follows:

predictor leastconns [**slowstart** *time*]

The optional **slowstart** *time* keyword and argument specify that the connections to the real server be in a slow-start mode. The ACE calculates a slow-start time stamp from the current time plus the *time* value that you specify. The ACE uses the time stamp as a fail-safe mechanism in case a real server stays in slow-start mode for a prolonged period of time (for example, with long-lived flows). Enter an integer from 1 to 65535 seconds. By default, **slowstart** is disabled.

Use the slow-start mechanism to avoid sending a high rate of new connections to servers that you have recently put into service. When a new real server enters slow-start mode, the ACE calculates and assigns an artificially high metric weight value to the new server and sends a small number of connections to the new server initially. The remaining connections go to the existing servers based on their weight and current connections. The real server (including the new server) with the lowest calculation (weight × current connections) receives the next connection request for the server farm.

Each time that the ACE assigns a new connection to the new real server, the ACE checks the validity of the time stamp. If the time stamp has expired, the ACE takes the server out of slow-start mode. As the new server connections are finished (closed), the ACE decrements both the new server connection count and the metric weight value by the number of closed connections.

A real server exits slow-start mode when one of the following events occurs:

- Slow-start time stamp expires
- New real server metric weight reaches a value of 0

Note It is possible for a new real server to remain in slow-start mode for a period that is longer than the *time* value you configure. This may happen because the ACE checks the time stamp only when it assigns a new connection to the real server.

The ACE places real servers in slow-start mode only if they have no existing connections to them. For example, if you have two real servers (RSERVER1 and RSERVER2) in a server farm that have the same number of active connections to them, slow start is configured for 180 seconds, and you add a new real server (RSERVER3) to the server farm, the ACE places the new real server in slow-start mode and sends a small number of new connections to it. The ACE equally divides the remainder of the new connections between RSERVER1 and RSERVER2. At this point, if RSERVER1 goes into any failed state (for example, PROBE-FAILED or ARP-FAILED) and then later it is placed back in service within 180 seconds of RSERVER3's addition to the server farm, the ACE does not put RSERVER1 in slow-start mode because of the connections to RSERVER1 that existed before the out-of-service event.

Instead, the ACE sends most or all of the new connections to RSERVER1 in an effort to make the connection count the same as RSERVER2 and ignores RSERVER3, which is still in slow-start mode. When the connection count for RSERVER1 and RSERVER2 are similar, then the ACE resumes sending a small number of connections to RSERVER3 for the duration of the slows-tart timer.

If RSERVER1 returns to the OPERATIONAL state more than 180 seconds after you added RSERVER3 to the server farm, then RSERVER3 will be out of slow-start mode. In this case, the ACE sends most or all of the new connections to RSERVER1 in an effort to make the connection count the same as RSERVER2 or RSERVER3, whichever has the least number of connections.

To ensure that a failed server with existing connections enters slow-start mode when it is placed back in service, configure **failaction purge** for the real server in a server farm. This command removes all existing connections to the real server if it fails. For details about the **failaction purge** command, see the section "Configuring the ACE Action when a Server Fails".

For example, to configure the ACE to select the real server using the **leastconns** predictor and the slow-start mechanism, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor leastconns slowstart 3600
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Least-Loaded Predictor

To instruct the ACE to select the server with the lowest load, use the **predictor least-loaded** command in server farm host or redirect configuration mode. With this predictor, the ACE uses SNMP probes to query the real servers for load parameter values (for example, CPU utilization or memory utilization). This predictor is considered adaptive because the ACE continuously provides feedback to the load-balancing algorithm based on the behavior of the real server.

To use this predictor, you must associate an SNMP probe with it. The ACE queries user-specified OIDs periodically based on a configurable time interval. The ACE uses the retrieved SNMP load value to determine the server with the lowest load. For details about configuring SNMP probes, see Chapter 4, Configuring Health Monitoring.

The syntax of this predictor command is as follows:

predictor least-loaded probe *name*

The *name* argument specifies the identifier of the existing SNMP probe that you want the ACE to use to query the server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to configure the ACE to select the real server with the lowest load based on feedback from an SNMP probe called PROBE_SNMP, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor least-loaded probe
PROBE_SNMP
host1/Admin(config-sfarm-host-predictor)#
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

You can influence the per-server load calculations that the ACE performs by using one or more of the following options:

- Autoadjust (see the “Using Autoadjust”, “Configuring Autoadjust for Average Load”, and “Turning Off Autoadjust” sections)

- Static real server weight (see the "Configuring a Real Server Weight" and the "Configuring the Weight of a Real Server in a Server Farm" sections)
- Current connection count (see the "Configuring the Current Connection Count" section)
- SNMP probe OID parameters (for details about configuring SNMP probes, see Chapter 4, Configuring Health Monitoring)

Using Autoadjust

Whenever a server's load reaches zero, by default, the ACE uses the autoadjust feature to assign a maximum load value of 16000 to that server to prevent it from being flooded with new incoming connections. The ACE periodically adjusts this load value based on feedback from the server's SNMP probe and other configured options.

Using the least-loaded predictor with the configured server weight and the current connection count option enabled, the ACE calculates the final load of a real server as follows:

final load = *weighted load* × *static weight* × *current connection count*

where:

- *weighted load* is the load reported by the SNMP probe
- *static weight* is the configured weight of the real server
- *current connection count* is the total number of active connections to the real server

The ACE recalculates the final load whenever the connection count changes, provided that the **weight connection** command is configured. If the **weight connection** command is not configured, the ACE updates the final load when the next load update arrives from the SNMP probe. For details about the **weight connection** command, see the "Configuring the Current Connection Count" section.

Configuring Autoadjust for Average Load

To instruct the ACE to apply the average load of the server farm to a real server whose load reaches zero, use the **autoadjust average** command in server farm host or redirect predictor configuration mode. The average load is the running average of the load values across all real servers in the server farm. The syntax of this command is as follows:

autoadjust average

For example, to instruct the ACE to apply the average load of the server farm to a real server whose load value reaches zero, enter:

```
host1/Admin(config-sfarm-host-predictor)# autoadjust average
```

To reset the behavior of the ACE to the default of applying the maximum load value of 16000 to a real server whose load is zero, enter:

```
host1/Admin(config-sfarm-host-predictor)# no autoadjust average
```

Turning Off Autoadjust

There may be times when you want the ACE to send all new connections to a real server whose load is zero. In this case, use the **autoadjust off** command in server farm host predictor configuration mode. The syntax of this command is as follows:

autoadjust off

This command overrides the default behavior of the ACE of setting the load value for a server with a load of zero to 16000. When you configure this command, the ACE sends all new connections to the server that has a load of zero until the next load update arrives from the SNMP probe for this server.

If two servers have the same lowest load (either zero or nonzero), the ACE load balances the connections between the two servers in a round-robin manner.

For example, to turn off the autoadjust feature for all servers in a server farm so that servers with a load of zero receive all new connections, enter:

```
host1/Admin(config-sfarm-host-predictor)# autoadjust off
```

To return the behavior of the ACE to the default of assigning the maximum load value of 16000 to zero-load servers, enter:

```
host1/Admin(config-sfarm-host-predictor)# no autoadjust off
```

Configuring the Current Connection Count

To instruct the ACE to use the current connection count in the final load calculation for a real server, use the **weight connection** command in server farm host or redirect predictor configuration mode. The syntax of this command is:

weight connection

When you configure this option, the ACE includes the current connection count in the total load calculation for each real server in a server farm.

For example, to include the current connection count in the final load calculation for a server, enter:

```
host1/Admin(config-sfarm-host)# predictor least-loaded probe
PROBE_SNMP
host1/Admin(config-sfarm-host-predictor)# weight connection
```

To reset the behavior of the ACE to the default of excluding the current connection count from the load calculation, enter the following command:

```
host1/Admin(config-sfarm-host-predictor)# no weight connection
```

Configuring the Application Response Predictor

To instruct the ACE to select the server with the lowest average response time for the specified response-time measurement based on the current connection count and server weight (if configured), use the **predictor response** command in server farm host or redirect configuration mode. This predictor is considered adaptive because the ACE continuously provides feedback to the load-balancing algorithm based on the behavior of the real server.

To select the appropriate server, the ACE measures the absolute response time for each server in the server farm and averages the result over a specified number of samples (if configured). With the default **weight connection** option configured, the ACE also takes into account the server's average response time and current connection count. This calculation results in a connection distribution that is proportional to the average response time of the server.

The syntax of this command is as follows:

predictor response {**app-req-to-resp** | **syn-to-close** | **syn-to-synack**}[**samples** *number*]

The keywords and arguments are as follows:

- **app-request-to-resp**—Measures the response time from when the ACE sends an HTTP request to a server to the time that the ACE receives a response from the server for that request. The ACE does not allow you to configure this predictor response in a generic load-balancing policy map.
- **syn-to-close**—Measures the response time from when the ACE sends a TCP SYN to a server to the time that the ACE receives a CLOSE from the server.
- **syn-to-synack**—Measures the response time from when the ACE sends a TCP SYN to a server to the time that the ACE receives the SYN-ACK from the server.
- **samples** *number*—(Optional) Specifies the number of samples over which you want to average the results of the response time measurement. Enter an integer from 1 to 16 in powers of 2. Valid values are 1, 2, 4, 8, and 16. The default is 8.

For example, to configure the response predictor to load balance a request based on the response time from when the ACE sends an HTTP request to a server to when the ACE receives a response back from the server and average the results over four samples, enter:

```
host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# predictor response app-req-to-resp
samples 4
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

To configure an additional parameter to take into account the current connection count of the servers in a server farm, use the **weight connection** command in server farm host predictor configuration mode. By default, this command is enabled. The syntax of this command is as follows:

weight connection

For example, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor response app-request-to-resp 
samples 4
host1/Admin(config-sfarm-host-predictor)# weight connection
```

To remove the current connection count from the calculation of the average server response time, enter:

```
host1/Admin(config-sfarm-host-predictor)# no weight connection
```

Configuring the Round-Robin Predictor

To instruct the ACE to select the next real server in the list of real servers based on the server weight (if configured), use the **roundrobin** command in server farm host or redirect configuration mode. If you did not configure a server weight, the ACE selects the next server in the list regardless of the server weight (regular round-robin). If you configured a weight for the server, the ACE selects the next server in the list based on the configured weight of the servers in the server farm (weighted round-robin). For information about setting the server weight, see the "Configuring a Real Server Weight" section and the "Configuring the Weight of a Real Server in a Server Farm" section.

The syntax of this command is as follows:

roundrobin

For example, to select the next server in the list of servers based on the configured weight (if configured), enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor roundrobin
```

To reset the predictor algorithm to the default of **roundrobin**, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring Server Farm HTTP Return Code Checking

You can configure HTTP return-code checking (retcode map) for a server farm by using the **retcode** command in server farm host configuration mode only. You can specify a single return code number or a range of return code numbers. For example, you can instruct the ACE to check for and count the number of occurrences of such return codes as HTTP/1.1 200 OK, HTTP/1.1 100 Continue, or HTTP/1.1 404 Not Found.

The syntax of this command is as follows:

> **retcode** *number1 number2* **check** {**count**
> | {**log** *threshold_number* **reset** *seconds1*}
> | {**remove** *threshold_number* **reset** *seconds1* [**resume-service** *seconds2*]}}

The keywords, arguments, and options are as follows:

- *number1*—Minimum value for an HTTP return code. Enter an integer from 100 to 599. The minimum value must be less than or equal to the maximum value.
- *number2*—Maximum value for an HTTP return code. Enter an integer from 100 to 599. The maximum value must be greater than or equal to the minimum value.
- **check**—Checks for HTTP return codes associated with the server farm.
- **count**—Tracks the total number of return codes received for each return code number that you specify.
- **log**—Specifies a syslog error message when the number of events reaches the threshold specified by the *threshold_number* argument.
- *threshold_number*—Threshold for the number of events that the ACE receives before it performs the **log** or **remove** action.
- **reset** *seconds1*—Specifies the time interval in seconds over which the ACE checks for the return code for the **log** or **remove** action. Enter an integer from 1 to 4294967295.
- **remove**—Specifies a syslog error message when the number of events reaches the threshold specified by the *threshold_number* argument and the ACE removes the server from service.

- **resume-service** *seconds2*—(Optional) Specifies the number of seconds that the ACE waits before it resumes service for the real server automatically after taking the real server out of service because the **remove** option is configured. Enter an integer from 30 to 3600 seconds. The default setting is 300.

The ACE performs the log or remove actions only if the *threshold_number* value for a particular retcode is reached within a specified period of time. The time period is defined from the receipt of a retcode until the next reset time.

For example, if you enter **retcode 404 404 check 100 remove reset 300**, the remove action will not take place unless the ACE counts more than 100 occurrences of HTTP/1.x 404 within 300 seconds since the last time the counter was reset.

You can configure multiple retcode maps on each server farm. You can view hit counts for retcode checking by using the **show serverfarm** command. For details, see the "Displaying Server Farm Statistics" section.

For example, to check for and count the number of return code hits for all return codes from 200 to 500 inclusive, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# retcode 200 500 check count
```

To remove the HTTP return-code map from the configuration, enter:

```
host1/Admin(config-sfarm-host)# no retcode 200 500
```

Configuring a Partial Server Farm Failover

By default, if you have configured a backup server farm and all real servers in the primary server farm go down, the primary server farm fails over to the backup server farm. For details about configuring a backup server farm, see the "Enabling Load Balancing to a Server Farm" section in Chapter 3, Configuring Traffic Policies for Server Load Balancing and Chapter 5, Configuring Stickiness.

A partial server farm failover allows you to specify a failover threshold using the **partial-threshold** command. The first value that you specify for this command is a percentage of the real servers in a server farm that must remain active for the server farm to stay up. Each time that a server is taken out of service (for example, using the CLI, a probe failure, or the retcode threshold is exceeded), the ACE is updated. If the percentage of active real servers in a server farm falls below the specified threshold, the primary server farm fails over to the backup server farm (if configured).

With partial server farm failover configured, the ACE allows current connections on the remaining active servers in the failed primary server farm to complete and redirects any new connection requests to the backup server farm.

To bring the primary server farm back into service, you specify another threshold value for the **back-inservice** keyword. When the number of active servers is greater than the configured value for this keyword, the ACE places the primary server farm back in service.

You can enable partial server farm failover by using the **partial-threshold** command in server farm host configuration mode. The syntax of this command is as follows:

partial-threshold *number1* **back-inservice** *number2*

The keywords and arguments are as follows:

- *number1*—Minimum percentage of real servers in the primary server farm that must remain active for the server farm to stay up. If the percentage of active real servers falls below this threshold, the ACE takes the server farm out of service. Enter an integer from 0 to 99.
- **back-inservice** *number2*—Specifies the percentage of real servers in the primary server farm that must be active again for the ACE to place the server farm back in service. Enter an integer from 0 to 99. The percentage value that you configure for the **back-inservice** keyword must be greater than or equal to the **partial-threshold** *number1* value.

For example, to configure a partial server farm failover, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# partial-threshold 40 back-inservice 60
```

To disable a partial server farm failover and return the ACE behavior to the default of failing over to the backup server (if configured) if all real servers in the server farm go down, enter:

```
host1/Admin(config-sfarm-host)# no partial-threshold
```

Associating a Real Server with a Server Farm

You can associate one or more real servers with a server farm and enter real-server server-farm configuration mode by using the **rserver** command in either server farm host or server farm redirect configuration mode. The real server must already exist. For information about configuring a real server, see the "Configuring Real Servers" section. You can configure a maximum of 16,384 real servers in a server farm.

The syntax of this command is as follows:

rserver *name* [*port*]

The arguments are as follows:

- *name*—Unique identifier of an existing real server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port*—(Optional) Port number used for the real server port address translation (PAT). Enter an integer from 1 to 65535.

If you choose not to assign a port number for the real server association with the server farm, by default, the ACE automatically assigns the same destination port that was used by the inbound connection to the outbound server connection. 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 automatically uses 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 typically define an outbound destination port of 80, 81, or 8080 for the back-end server connection.

For example, to identify real server SERVER1 and specify port 80 for the outgoing connection, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 80
host1/Admin(config-sfarm-host-rs)#
```

To remove the real server association with the server farm, enter:

```
host1/Admin(config-sfarm-host)# no rserver server1 80
```

Configuring Additional Real Server Attributes in a Server Farm

After you have associated a real server with a server farm, you can configure other real server attributes as described in the following topics:

- Configuring the Weight of a Real Server in a Server Farm
- Configuring a Backup Server for a Real Server
- Configuring Health Monitoring for a Real Server in a Server Farm
- Configuring AND Logic for Real Server Probes in a Server Farm
- Configuring Connection Limits for a Real Server in a Server Farm
- Configuring Rate Limiting for a Real Server in a Server Farm
- Placing a Real Server in Service
- Gracefully Shutting Down a Server with Sticky Connections

Configuring the Weight of a Real Server in a Server Farm

The ACE uses a real server weight value with the weighted round-robin and least connections predictor methods. Servers with higher weight values receive a proportionally higher number of connections than servers with lower weight values. If you do not specify a weight in real configuration mode under a server farm, the ACE uses the weight that you configured for the global real server in real server configuration mode. See the "Configuring a Real Server Weight" section.

Note Server weights take effect only when there are open connections to the servers. When there are no sustained connections to any of the servers, the least-connections predictor method behaves like the round-robin method.

You can configure the weight of a real server in a server farm by using the **weight** command in server farm host real server configuration mode. The syntax of this command is as follows:

weight *number*

The *number* argument is the weight value assigned to a real server in a server farm. Enter an integer from 1 to 100. The default is 8.

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# weight 50
```

To reset the configured weight of a real server to the default of 8, enter:

```
host1/Admin(config-sfarm-host-rs)# no weight
```

Configuring a Backup Server for a Real Server

You can designate an existing real server as a backup server for another real server in case that server becomes unavailable. A backup server is sometimes referred to as a sorry server. If the real server becomes unavailable, the ACE automatically redirects client requests to the backup server. You can configure a backup server by using the **backup-rserver** command in server farm host real server configuration mode.

Note If you are configuring a backup server for a real server, be sure to specify the optional **standby** key word when you place the backup server in service. For details, see the "Placing a Real Server in Service" section.

The syntax of this command is as follows:

backup-rserver *name* [*port*]

The arguments are as follows:

- *name*—Name of an existing real server that you want to configure as a backup server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port*—(Optional) Port number used for the backup real server port address translation (PAT). Enter an integer from 0 to 65535.

Note You can chain a maximum of three backup real servers.

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# backup-rserver SERVER2 4000
```

To remove the backup server from the configuration, enter:

```
host1/Admin(config-sfarm-host-rs)# no backup-rserver
```

Configuring Health Monitoring for a Real Server in a Server Farm

You can monitor the health of the real servers in a server farm by configuring one or more health probes (sometimes called keepalives) directly on the real servers in the server farm. You can associate multiple probes of the same or different protocols with each real server in a server farm. Once you configure the probes, the ACE periodically sends the probes to the real servers. If the ACE receives the appropriate responses from the servers, the ACE includes the servers in load-balancing decisions. If not, the ACE marks the servers as PROBE-FAILED, depending on the configured number of retries.

When you configure multiple probes on a real server in a server farm, the probes have an OR logic associated with them. This means that if one of the probes fails, the real server associated with that probe fails and enters the PROBE-FAILED state. You can also configure AND logic for multiple probes on a real server in a server farm. For details, see the "Configuring AND Logic for Real Server Probes in a Server Farm" section.

You can associate a health probe with a real server in a server farm by using the **probe** command in server farm host real server configuration mode. The syntax of this command is as follows:

probe *probe_name*

For the *probe_name* argument, enter the name of an existing probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. For information about creating and configuring probes, see Chapter 4, Configuring Health Monitoring.

For example, to configure a probe on a real server in a server farm, enter:

```
host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# rserver SERVER1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# probe TCP_PROBE
```

To remove the real server health probe from the configuration, enter:

```
host1/Admin(config-sfarm-host-rs)# no probe TCP_PROBE
```

Configuring AND Logic for Real Server Probes in a Server Farm

By default, multiple probes that you configure directly on a real server in a server farm have an OR logic associated with them. This means that, if one of the real server probes fails, the real server fails and enters the PROBE-FAILED state. To configure a real server in a server farm to remain in the OPERATIONAL state unless all probes associated with it fail (AND logic), use the **fail-on-all** command in server farm host real server configuration mode. This command is applicable to all probe types. For more information about creating and configuring probes, see Chapter 4, Configuring Health Monitoring. The syntax of this command is as follows:

fail-on-all

You can selectively configure this command on only certain real servers in the server farm. Any real server that you do not configure with the **fail-on-all** command, maintains its default OR logic with respect to probes.

For example, to configure the SERVER1 real server in SFARM1 to remain in the OPERATIONAL state unless all associated probes fail, enter:

```
host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# rserver SERVER1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# probe HTTP_PROBE
host1/Admin(config-sfarm-host-rs)# probe ICMP_PROBE
host1/Admin(config-sfarm-host-rs)# fail-on-all
```

If either HTTP_PROBE or ICMP_PROBE fails, the SERVER1 real server remains in the OPERATIONAL state. If both probes fail, the real server fails and enters the PROBE-FAILED state.

To remove the AND probe logic from the real server in a server farm, enter:

```
host1/Admin(config-sfarm-host-rs)# no fail-on-all
```

Configuring Connection Limits for a Real Server in a Server Farm

You can prevent a real server in a server farm from becoming overloaded by limiting the number of connections allowed to that server. You can limit the number of connections to a real server by using the **conn-limit** command in server farm real server configuration mode. The syntax of this command is as follows:

conn-limit max *maxconns* **min** *minconns*

The keywords and arguments are as follows:

- **max** *maxconns*—Specifies the maximum number of active connections to a real server. When the number of connections exceeds the *maxconns* threshold value, the ACE stops sending connections to the real server until the number of connections falls below the configured *minconns* value. Enter an integer from 2 to 4000000. The default is 4000000.
- **min** *minconns*—Specifies the minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maxconns threshold. The *minconns* value must be less than or equal to the *maxconns* value. The default is 4000000.

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# conn-limit max 5000 min 4000
```

To remove the connection limit from a real server, enter:

```
host1/Admin(config-sfarm-host-rs)# no conn-limit
```

Configuring Rate Limiting for a Real Server in a Server Farm

In addition to preserving system resources by limiting the total number of active connections to a real server in a server farm (see the "Configuring Real Server Connection Limits" section), the ACE allows you to limit the connection rate and the bandwidth rate of a real server in a server farm. The connection rate is the number of connections per second that is received by the ACE and destined to a particular real server. The bandwidth rate is the number of bytes per second that is received by the ACE and destined for a particular real server.

Once the connection-rate limit or the bandwidth-rate limit of a real server in a server farm is reached, the ACE blocks any further traffic destined to that real server until the connection rate or bandwidth rate drops below the configured limit. The ACE removes the blocked real server from future load-balancing decisions and considers only those real servers that have a current connection rate or bandwidth less than the configured limit. By default, the ACE does not limit the connection rate or the bandwidth rate of real servers in a server farm.

You can also limit the connection rate and the bandwidth rate of the following:

- Real server at the aggregate level. For details, see the "Configuring Real Server Rate Limiting" section.
- Virtual server in a connection parameter map. For details, see the *Cisco Application Control Engine Module Security Configuration Guide*.

Note The connection rate or bandwidth rate limit that you configure on a real server associated with a server farm cannot exceed the aggregate connection or bandwidth rate limit that you configure on a real server outside of a server farm.

You can limit the connection rate or the bandwidth rate of a real server in a server farm by using the **rate-limit** command in server farm host real server configuration mode or server farm redirect real server configuration mode. The syntax of this command is as follows:

rate-limit {**connection** *number1* | **bandwidth** *number2*}

The keywords and arguments are as follows:

- **connection** *number1*—Specifies the real server connection-rate limit in connections per second. Enter an integer from 2 to 4294967295. There is no default value.
- **bandwidth** *number2*—Specifies the real server bandwidth-rate limit in bytes per second. Enter an integer from 2 to 300000000. There is no default value.

For example, to limit the connection rate of a real server to 100000 connections per second, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# rate-limit connection 100000
```

To return the behavior of the ACE to the default of not limiting the real-server connection rate, enter:

```
host1/Admin(config-sfarm-host-rs)# no rate-limit connection 100000
```

For example, to limit the real-server bandwidth rate to 5000000 bytes per second, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# rate-limit bandwidth 5000000
```

To return the behavior of the ACE to the default of not limiting the real-server bandwidth, enter:

```
host1/Admin(config-sfarm-host-rs)# no rate-limit bandwidth 5000000
```

Placing a Real Server in Service

Before you can start sending connections to a real server in a server farm, you must place it in service. You can place a real server in a server farm in service by using the **inservice** command in either server farm host real server or server farm redirect real server configuration mode. The syntax of this command is as follows:

inservice [**standby**]

Use the optional **standby** keyword when you are configuring backup real servers. The **standby** keyword specifies that a backup real server remain inactive unless the primary real server fails. If the primary server fails, the backup server becomes active and starts accepting connections.

Note You can modify the configuration of a real server in a server farm without taking the server out of service.

For example, to place a real server in service and have it remain inactive until the primary real server fails, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# inservice standby
```

To take a backup real server out of the standby state and out of service for maintenance or software upgrades, enter:

```
host1/Admin(config-sfarm-host-rs)# no inservice
```

To take a backup real server out of the standby state only and put it in service so that it can start accepting connections, enter:

```
host1/Admin(config-sfarm-host-rs)# no inservice standby
```

Note The **no inservice standby** command has an effect on a backup real server only if **inservice standby** is configured on that backup real server. The **standby** option applies only to a backup real server.

For example, consider the following configuration:

```
serverfarm SFARM1
 rserver SERVER1
   backup-rserver SERVER2
   inservice
 rserver SERVER2
   backup-rserver SERVER3
   inservice standby
 rserver SERVER3
   inservice standby
```

If you enter **no inservice standby** for SERVER2, the ACE takes SERVER2 out of the standby state and places it in service. SERVER2 will start to receive half of the connections; SERVER1 receives the other half. If SERVER1 fails, SERVER2 also will receive the connections of SERVER1.

Note You can chain a maximum of three backup real servers.

Gracefully Shutting Down a Server with Sticky Connections

In addition to putting a backup real server in service standby, another use of the **inservice standby** command is to provide the graceful shutdown of primary real servers. Use this command to gracefully shut down servers with sticky connections. When you enter this command for a primary real server, the ACE does the following:

- Tears down existing non-TCP connections to the server
- Allows current TCP connections to complete
- Allows new sticky connections for existing server connections that match entries in the sticky database
- Load balances all new connections (other than the matching sticky connections mentioned above) to the other servers in the server farm
- Eventually takes the server out of service

For example, to perform a graceful shutdown on a primary real server with sticky connections in a server farm, enter:

```
host1/Admin(config)# serverfarm sf1
host1/Admin(config-sfarm-host)# rserver rs1
host1/Admin(config-sfarm-host-rs)# inservice standby
```

Configuring a Backup Server Farm

Configure a backup server farm to ensure that, if all the real servers in a server farm go down, the ACE continues to service client requests. You configure a backup server farm as an action in a Layer 7 policy map under a Layer 7 class map using the **serverfarm** *name1* [**backup** *name2*] command. For details about configuring a backup server farm under a policy, see the "Enabling Load Balancing to a Server Farm" section in Chapter 3, Configuring Traffic Policies for Server Load Balancing. You can also configure a backup server farm under a sticky group using the same command. For details about a backup server farm under a sticky group, see the "Backup Server Farm Behavior with Stickiness" section in Chapter 5, Configuring Stickiness.

Specifying No NAT

You can instruct the ACE not to use NAT to translate the VIP address to the server IP address by using the **transparent** command in serverfarm host configuration mode. Use this command in firewall load balancing (FWLB) when you configure the insecure and secure sides of the firewall as a server farm. For details about FWLB, see Chapter 6, Configuring Firewall Load Balancing. The syntax of this command is as follows:

transparent

For example, enter:

```
host1/Admin(config-sfarm-host)# transparent
```

Configuring Asymmetric Server Normalization

Asymmetric Server Normalization (ASN) allows the ACE to load balance an initial request from the client to a real server; however, the server directly responds to the client bypassing the ACE. This behavior allows the acceleration of server to client communications and is transparent to the client. When the ACE operates in ASN, it does not perform any network translation when receiving packets destined to the VIP address. Traffic from the client hits the VIP address and the ACE uses the address as the destination address but rewrites the destination MAC address to the address of the real server.

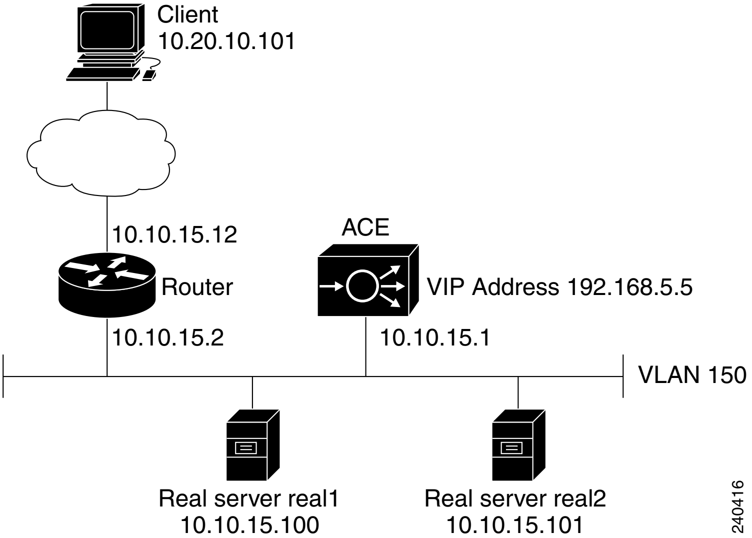
This section contains the following topics:

- ASN Sample Topology
- ASN Configuration Considerations
- Configuring ASN on the ACE

ASN Sample Topology

Figure 2-2 shows an ASN network topology where the VIP address does not belong to the IP subnet of the real servers.

Figure 2-2 ASN Network Topology Example



The topology consists of a client (10.20.10.101), a router, the ACE, and two real servers (10.10.15.100 and 10.10.15.101). Both real servers are grouped in a server farm represented by the VIP address 192.168.5.5. The router and the ACE are on subnet 10.10.15.0/24. When the client connects to 192.168.5.5, its packets hit the router. The preferred method for the router to forward traffic to the VIP address is a static route to 192.168.5.5 through 10.10.15.1. The router forwards the traffic destined to 192.168.5.5 to the MAC address of the ACE. There is no reason for the router to ever use Address Resolution Protocol (ARP) for 192.168.5.5.

ASN Configuration Considerations

When configuring ASN, note the following:

- Real servers must be configured with a virtual interface, sometimes referred to as a loopback interface.
- The virtual interface has an IP address of the VIP address.
- The virtual interface must not respond to ARP requests.
- ASN is a Layer 4-only feature. The load balancer does not participate in both legs of bidirectional client and server communications. Therefore, features that require TCP/SSL termination are not applicable to ASN environments. Features that are incompatible with ASN are as follows:
  - HTTP header parsing
  - HTTP header insert
  - Cookie sticky
  - SYN cookies
  - SSL termination or initiation
  - End-to-end SSL
- Because the ACE sees only the client-server leg of the connection, it cannot detect whether the connection has ended. With TCP being full duplex, the presence of a FIN in one direction does not mean that the ACE should tear down the entire connection. Therefore, connections age out after the connection inactivity timeout. The default connection inactivity timeouts are as follows:
  - ICMP—2 seconds
  - TCP—3600 seconds (1 hour)
  - UDP—120 seconds (2 minutes)

  You can adjust the timeouts by using the **set timeout inactivity** command in parameter map connection configuration mode. To apply the timeout to a specific class of traffic, create a class map and policy map.

The following example creates a connection parameter map that times outs all TCP connections after 20 seconds of inactivity and applies it under a Layer-4 policy map:

```
host1/Admin(config)# parameter-map type connection timeouts
host1/Admin(config-parammap-conn)# set time activity 20
host1/Admin(config-parammap-conn)# exit
host1/Admin(config)# policy-map multi-match LBPOL
host1/Admin(config-pmap)# class vip
host1/Admin(config-pmap-c)# connection advanced-options timeouts
```

The ACE automatically tears down both connection entries after the timer has expired and sends a TCP RST to the client and server.

- Real servers must be Layer-2 adjacent to the load balancer when running in ASN. The reason is that the load balancer performs a destination MAC address rewrite, and the rewritten MAC always belongs to a real server. In the example in Figure 2-2, real servers must be placed on VLAN 150. You cannot insert a routed hop between the ACE and the real servers.
- You must create a virtual interface on the real servers for each VIP address that exists on the load balancers. Depending on the topology, you may need to configure ARP suppression on the servers.

Configuring ASN on the ACE

To configure ASN on the ACE, do the following:

- Configure the server farm as a transparent server farm by using the **transparent** command in serverfarm host configuration mode. When you configure a transparent server farm, the ACE does not perform NAT of the VIP address to a real server address.
- Disable the TCP normalization on the client-side interface by using the **no normalization** command in VLAN interface configuration mode. Disabling normalization turns off ACE statefulness, allowing the ACE to accept client-originated TCP ACKs and data without having seen a full three-way handshake previously. For more information of disabling TCP normalization, see the *Cisco Application Control Engine Module Security Configuration Guide*.

With ASN, the servers respond directly to the client by bypassing the ACE. However, the clients traverse the ACE to reach the servers. The result is traffic asymmetry that is handled by disabling normalization.

The following example is the configuration of the real servers, server farm, and VIP address for the ASN topology in Figure 2-2:

```
access-list inbound line 10 extended permit ip host 10.20.10.101 host
192.168.5.5

rserver host real1
    ip address 10.10.15.100
    inservice
rserver host real2
    ip address 10.10.15.101
    inservice

serverfarm host farm1
    transparent
    rserver real1
        inservice
    rserver real2
        inservice

class-map match-all vip
    2 match virtual-address 192.168.5.5 any

parameter-map type connection timeouts
    set time activity 20

policy-map type loadbalance first-match lbpol
    class class-default
        serverfarm farm1
policy-map multi-match LBPOL
    class vip
        loadbalance vip inservice
        loadbalance policy lbpol
        loadbalance vip icmp-reply active
        connection advanced-options timeouts

interface vlan 150
    description server-side
    ip address 10.10.15.1 255.255.255.0
    no normalization
    access-group input inbound
    service-policy input LBPOL
    no shutdown

ip route 0.0.0.0 0.0.0.0 10.10.15.2
```

When the client connects to the VIP address, the ACE logs a bidirectional connection creation that you can display by using the **show conn** command in Exec mode.

However, the second leg of the connection from the server to the client remains idle and its byte count does not increment.

To periodically check the health status of the servers, you must probe the virtual address, 192.168.5.5 in the previous configuration example. The following example is a simple ICMP probe configuration:

```
probe icmp ICMP1
    ip address 192.168.5.5
serverfarm host farm1
    probe ICMP1
```

Example of a Server Farm Configuration

The following example shows the running configuration that selects the next server in the list of real servers based on the server weight (weighted round-robin). The server farm configuration appears in bold in the example.

In this configuration, the ACE uses a real server weight value with the weighted round-robin predictor method. Real servers with higher weight values receive a proportionally higher number of connections than real servers with lower weight values.

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

rserver host SERVER1
  ip address 192.168.252.245
  inservice
rserver host SERVER2
  ip address 192.168.252.246
  inservice
rserver host SERVER3
  ip address 192.168.252.247
  inservice
rserver host SERVER4
  ip address 192.168.252.248
  inservice
rserver host SERVER5
  ip address 192.168.252.249
  inservice
rserver host SERVER6
  ip address 192.168.252.250
  inservice
```

```
serverfarm host SFARM1
  probe HTTP_PROBE
  predictor roundrobin
  rserver SERVER1
    weight 10
    inservice
  rserver SERVER2
    weight 20
    inservice
   rserver SERVER3
    weight 30
    inservice

serverfarm host SFARM2
  probe HTTP_PROBE
  predictor roundrobin
  rserver SERVER4
    weight 10
    inservice
  rserver SERVER5
    weight 20
    inservice
   rserver SERVER6
    weight 30
    inservice

class-map match-all L4WEB_CLASS
  2 match virtual-address 192.168.120.112 tcp eq www
policy-map type loadbalance first-match L7WEB_POLICY
  class class-default
    serverfarm SFARM1 backup SFARM2
policy-map multi-match L4WEB_POLICY
  class L4WEB_CLASS
    loadbalance vip inservice
    loadbalance policy L7WEB_POLICY
    loadbalance vip icmp-reply active
    nat dynamic 1 VLAN 120

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 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4WEB_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```

Displaying Real Server Configurations and Statistics

This section describes the commands that you can use to display information about the real-server configuration and statistics. It contains the following topics:

- Displaying Real Server Configurations
- Displaying Real Server Statistics
- Displaying Real Server Connections

Displaying Real Server Configurations

You can display information about the real-server configuration by using the **show running-config rserver** command in Exec mode. The syntax of this command is as follows:

show running-config rserver

Displaying Real Server Statistics

You can display summary or detailed statistics for a named real server or for all real servers by using the **show rserver** command in Exec mode. The syntax of this command is as follows:

show rserver [*name*] [**detail**]

The argument and option are as follows:

- *name*—(Optional) Identifier of an existing real server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **detail**—(Optional) Displays detailed statistics for the real server name that you enter or for all real servers.

For example, to display detailed statistics for all configured real servers, enter:

```
host1/Admin# show rserver detail
```

Table 2-3 describes the fields in the **show rserver** command output.

Table 2-3 Field Descriptions for the show rserver Command Output

Field	Description
Summary Real Server Information	
Rserver	Name of the real server.
Type	Type of configured real server: HOST or REDIRECT.
State	Current state of the real server: • INACTIVE—Real server is not associated with a server farm • OPERATIONAL—Real server is in primary mode and is Up • STANDBY—Real server is in backup mode and is Up • OUTOFSERVICE—Real server is no longer in service (for both the primary and the backup real server)
Real	
Serverfarm	Name of the server farm to which the server belongs.
IP Address	IP address of the real server.
Weight	Weight of the real server in the server farm.
State	State of the server farm: OPERATIONAL or OUTOFSERVICE.
Current Connections	Number of active connections to the real server.
Total Connections	Total number of connections to the real server.
Detailed Real Server Information	
Rserver	Name of the real server.
Type	Type of configured real server: HOST or REDIRECT.
State	Current state of the real server: INACTIVE (not associated with a server farm), OPERATIONAL or OUTOFSERVICE.

Table 2-3 Field Descriptions for the show rserver Command Output (continued)

Field	Description
Description	User-entered text description of the real server with a maximum of 240 alphanumeric characters.
Max-conns	Configured maximum allowable number of active connections to a real server.
Min-conns	Configured minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold.
Out-of-rotation-count	Number of times that the real server was not considered for load balancing because the number of connections, connection rate, or bandwidth rate exceeded the configured limits of the server.
Conn-rate-limit	Configured connection rate limit of the real server in connections per second.
Bandwidth-rate-limit	Configured bandwidth rate limit of the real server in bytes per second.
Weight	Configured weight of the real server.
Redirect Str	For redirect servers only, the configured redirect string.
Redirect Code	For redirect servers only, the configured redirect code.
Redirect Port	For redirect servers only, the configured redirect port.
Real	
Serverfarm	Name of the server farm to which the server belongs.
IP Address	IP address of the real server.
Weight	Configured weight of the real server in the server farm.
State	Current state of the real server: OPERATIONAL or OUTOFSERVICE.
Current Connections	Number of active connections to the real server.
Total Connections	Total number of connections to the real server.

Table 2-3 Field Descriptions for the show rserver Command Output (continued)

Field	Description
Max-Conns	Configured maximum allowable number of active connections to a real server.
Min-Conns	Configured minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold.
Conn-Rate-Limit	Configured connection rate limit in connections per second of the real server in the server farm.
Bandwidth-Rate-Limit	Configured bandwidth rate limit in bytes per second of the real server in the server farm.

Table 2-3 Field Descriptions for the show rserver Command Output (continued)

Field	Description
Out-of-Rotation-Count	Number of times that the real server was not considered for load balancing because the number of connections, connection rate, or bandwidth rate exceeded the configured limits of the server.
Total Conn-failures	Total number of connection attempts that failed to establish a connection to the real server. For Layer 4 traffic with normalization on, the count increments if the three-way handshake fails to be established for either of the following reasons: • A RST comes from the client or the server after a SYN-ACK. • The server does not reply to a SYN. The connection times out. For Layer 4 traffic with normalization off, the count does not increment. For L7 traffic (normalization is always on), the count increments if the three-way handshake fails to be established for either of the following reasons: • A RST comes from the server after the front-end connection is established • The server does not reply to a SYN. The connection times out.

Displaying Real Server Connections

You can display active inbound and outbound real server connections by using the **show conn rserver** command in Exec mode. The syntax of this command is as follows:

show conn rserver *name1* [*port_number*][**serverfarm** *name2*] [**detail**]

The arguments and options are as follows:

- *name1*—Identifier of an existing real server whose connections you want to display. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port_number*—(Optional) Port number of the server. Enter an integer from 1 to 65535.
- **serverfarm***name2*—(Optional) Identifier of an existing server farm with which the real server is associated. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **detail**—(Optional) Displays additional information for the connection including idle time, elapsed time, byte count, packet count, and, if applicable, the state of the connection in the reuse pool.

For example, enter:

```
host1/Admin# show conn rserver SERVER1 4000 serverfarm SFARM1
```

Table 2-4 describes the fields in the **show conn rserver** command output.

Table 2-4 Field Descriptions for the show conn rserver Command Output

Field	Description
Conn-ID	Numerical identifier of the connection.
NP	Network processor that is handling the inbound or outbound real server connection.
Dir	Direction of the traffic on the connection: in or out.
Proto	TCP/IP protocol used in the connection.
VLAN	Numerical identifier of the virtual LANs used in the inbound and the outbound connections.
Source	Source IP addresses and ports of the inbound and outbound connections.
Destination	Destination IP addresses and ports of the inbound and outbound connections.
Idle Time	Length of time that this connection has been idle (**detail** option).
Byte Count	Number of bytes that have traversed the connection (**detail** option).

Table 2-4 Field Descriptions for the show conn rserver Command Output (continued)

Field	Description
Elapsed Time	Length of time that has elapsed since the connection was established (**detail** option).
Packet Count	Number of packets that have traversed the connection (**detail** option).
Conn in Reuse Pool	Indication of whether the ACE has placed the connection in the pool for possible reuse (**detail** option). Valid values are TRUE or FALSE.
State	Current state of the connection. Non-TCP connections display as “--”. Possible values for TCP connections are as follows: • INIT—Connection is closed. This is the initial state of a connection. • SYNSEEN—ACE received a SYN packet from a client. • SYNACK—ACE sent a SYNACK packet to a client. • ESTAB—Three-way handshake completed and the connection is established. • CLSFIN—ACE closed the connection with a FIN packet. • CLSRST—ACE closed the connection by resetting it. • CLSTIMEOUT—ACE closed the connection because the connection timed out. • CLOSED—Connection is half closed.

Clearing Real Server Statistics and Connections

This section describes the commands that you use to clear real server statistics and connection information. It contains the following topics:

- Clearing Real Server Statistics
- Clearing Real Server Connections

Clearing Real Server Statistics

You can reset statistics to zero for all instances of a particular real server regardless of the server farms that it is associated with by using the **clear rserver** command in Exec mode. The syntax of this command is as follows:

clear rserver *name*

The *name* argument is the identifier of an existing real server whose statistics you want to clear. Enter a real server name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to reset the statistics of real server SERVER1, enter:

```
host1/Admin# clear rserver SERVER1
```

Note If you have redundancy configured, you need to explicitly clear real-server statistics on both the active and the standby ACEs. Clearing statistics on the active module only leaves the standby module's statistics at the old values.

Clearing Real Server Connections

You can clear real server connections by using the **clear conn rserver** command in Exec mode. The syntax of this command is as follows:

clear conn rserver *name1* [*port*] **serverfarm** *name2*

The arguments, options, and keyword are as follows:

- *name1*—Unique identifier of an existing real server whose connections you want to clear. Enter the real server name as unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.
- **serverfarm** *name2*—Specifies a unique identifier of the server farm with which the real server is associated. Enter the server farm name as an unquoted text string with no spaces and maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin# clear conn rserver SERVER1 4000 SFARM1
```

Displaying Server Farm Configurations and Statistics

This section describes the commands that you can use to display information about the server farm configuration and statistics. It contains the following topics:

- Displaying Server Farm Configurations
- Displaying Server Farm Statistics
- Displaying Server Farm Connections

Displaying Server Farm Configurations

You can display information about the server farm configuration by using the **show running-config serverfarm** command in Exec mode. The syntax of this command is as follows:

show running-config serverfarm

Displaying Server Farm Statistics

You can display summary or detailed server-farm statistics by using the **show serverfarm** command in Exec mode. The syntax of this command is as follows:

show serverfarm [*name* [**retcode**]] [**detail**]

The argument and options are as follows:

- *name*—(Optional) Statistics for the server farm specified in the *name* argument. Enter the name of an existing server farm as an unquoted text string with a maximum of 64 alphanumeric characters.
- **retcode**—(Optional) Displays the HTTP return codes associated with the server farm.
- **detail**—(Optional) Displays detailed statistics for the specified server farm.

For example, enter:

```
host1/Admin# show serverfarm sfarm1 detail
```

Table 2-5 describes the fields in the **show serverfarm detail** command output.

Table 2-5 Field Descriptions for the show serverfarm detail Command Output

Field	Description
Serverfarm	Name of the server farm.
Type	Configured server farm type: HOST or REDIRECT.
Total Rservers	Total number of real servers associated with the server farm.
Active Rservers	Number of real servers that are active in the server farm.
Description	User-entered text description of the server farm with a maximum of 240 alphanumeric characters.
State	Current state of the server farm. Possible values are ACTIVE or INACTIVE.

Table 2-5 Field Descriptions for the show serverfarm detail Command Output (continued)

Field	Description
Predictor and field values	Configured load-balancing method and values for various fields associated with the predictor. Possible predictor values are as follows: • HASH-ADDRSRC • HASH-ADDRDEST • HASH-COOKIE • HASH-HEADER • HASH-HTTP-CONTENT • HASH-LAYER4-PAYLOAD • HASH-URL • LEASTBANDWIDTH • LEASTCONNS • LEASTLOADED • RESPONSE • ROUNDROBIN
Failaction	Action that the ACE takes for connections if a real server fails in a server farm. Possible actions are purge or none.
Back-Inservice	Configured value of the **back-inservice** keyword of the **partial-threshold** command. Specifies the minimum percentage of real servers in the primary server farm that must be active again for the ACE to place the server farm back in service.
Partial-Threshold	Configured value of the **partial-threshold** command. Specifies the minimum percentage of real servers in the primary server farm that must remain active for the server farm to stay up.
Num Times Failover	Number of time that the server farm failed over to the backup server farm.

Table 2-5 Field Descriptions for the show serverfarm detail Command Output (continued)

Field	Description
Num Times Back Inservice	Number of times that the ACE placed the server farm back in service after a failover.
Total Conn-Dropcount	Total number of connections that the ACE discarded because the number of connections exceeded the configured **conn-limit max** value. See the "Configuring Real Server Connection Limits" section.
Real	
Rserver	Name of the real server associated with the server farm.
IP Address:Port	IP address and port of the real server.
Weight	Weight assigned to the real server in the server farm.
State	Current state of the real server. Possible states are OPERATIONAL or OUTOFSERVICE.
Current Connections	Number of active connections to the real server.
Total Connections	Total number of connections to the specified server farm.
Description	User-entered text description of the real server.
Max-conns	Configured maximum allowable number of active connections to a real server.
Min-conns	Configured minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold.
Out-of-rotation-count	Number of times that the real server was not considered for load balancing because the number of connections, connection rate, or bandwidth rate exceeded the configured limits of the server.
Conn-rate-limit	Configured connection rate limit of the real server in connections per second.

Table 2-5 Field Descriptions for the show serverfarm detail Command Output (continued)

Field	Description
Bandwidth-rate-limit	Configured bandwidth rate limit of the real server in bytes per second.
Retcode	Configured HTTP return code.
Average Response Time	For the response predictor, the average response time of the real server in milliseconds.
Connection Failures	Total number of connection attempts that failed to establish a connection to the real server.
Average Bandwidth	For the leastbandwidth predictor, the average bandwidth for the real server based on the bytes transferred between the ACE to real server. The ACE uses this calculated average bandwidth to load balance further traffic for the VIP.

Displaying Server Farm Connections

You can display the current active inbound and outbound connections for all real servers in a server farm by using the **show conn serverfarm** command in Exec mode. The syntax of this command is as follows:

show conn serverfarm *name* [**detail**]

The argument and option are as follows:

- *name*—Name of the server farm whose real-server connections you want to display. Enter an unquoted alphanumeric text string with no spaces and a maximum of 64 alphanumeric characters.
- **detail**—(Optional) Displays detailed connection information for the specified server farm, including the idle time, elapsed time, byte count, packet count, and state of the connection in the reuse pool.

For example, enter:

```
host1/Admin# show conn serverfarm sfarm1
```

Table 2-6 describes the fields in the **show conn serverfarm** command output.

Table 2-6 Field Descriptions for the show conn serverfarm Command Output

Field	Description
Conn-ID	Numerical identifier of the connection.
Dir	Direction of the traffic on the connection: in or out.
Proto	TCP/IP protocol used for this connection.
VLAN	Numerical identifier of the virtual LANs used in the inbound and the outbound connections.
Source	Source IP addresses and ports of the inbound and outbound connections.
Destination	Destination IP addresses and ports of the inbound and outbound connections.
State	Current state of the connection. Non-TCP connections display as "--". Possible values for TCP connections are as follows: • INIT—Connection is closed. This is the initial state of a connection. • SYNSEEN—ACE received a SYN packet from a client. • SYNACK—ACE sent a SYNACK packet to a client. • ESTAB—Three-way handshake completed and the connection is established. • CLSFIN—ACE closed the connection with a FIN packet. • CLSRST—ACE closed the connection by resetting it. • CLSTIMEOUT—ACE closed the connection because the connection timed out. • CLOSED—Connection is half closed.
Idle Time	Length of time that this connection has been idle (**detail** option).

Table 2-6 Field Descriptions for the show conn serverfarm Command Output (continued)

Field	Description
Byte Count	Number of bytes that have traversed the connection (**detail** option).
Elapsed Time	Length of time that has elapsed since the connection was established (**detail** option).
Packet Count	Number of packets that have traversed the connection (**detail** option).
Conn in Reuse Pool	Indication of whether the ACE has placed the connection in the pool for possible reuse (**detail** option). Valid values are TRUE or FALSE.

Clearing Server Farm Statistics

You can reset statistics to zero for all real servers in a specified server farm by using the **clear serverfarm** command in Exec mode. The syntax of this command is as follows:

clear serverfarm *name* [**predictor** | **retcode**]

The argument and option are as follows:

- *name*—Identifier of an existing server farm with real server statistics that you want to reset. Enter a server farm name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **predictor**—Resets the average bandwidth field for each real server in the specified server farm, as displayed by the **show serverfarm** *name* **detail** command.
- **retcode**—Resets all HTTP return code (retcode) statistics for the specified server farm. For information about configuring a server farm to perform HTTP retcode checking, see the "Configuring Server Farm HTTP Return Code Checking" section.

For example, to reset the statistics (including retcode statistics) of all real servers in server farm SF1, enter:

```
host1/Admin# clear serverfarm SF1 retcode
```

Note If you have redundancy configured, you need to explicitly clear server-farm statistics on both the active and the standby ACEs. Clearing statistics on the active module only leaves the standby module's statistics at the old values.

Where to Go Next

Once you are satisfied with your real server and server-farm configurations, configure an SLB traffic policy to filter interesting traffic and load balance it to the servers in your server farm, as described in Chapter 3, Configuring Traffic Policies for Server Load Balancing.

CHAPTER 3

Configuring Traffic Policies for Server Load Balancing

This chapter describes how to configure the ACE module to use classification (class) maps and policy maps to filter and match interesting network traffic based on various criteria and load balance the traffic to real servers in server farms using one of the ACE load-balancing predictor methods.

This chapter contains the following major sections:

- Configuring the ACE to Perform Hashing When the Source and Destination Ports Are Equal
- Configuring RDP Load Balancing
- Configuring RADIUS Load Balancing
- Configuring RTSP Load Balancing
- Configuring SIP Load Balancing
- Example of a Server Load-Balancing Policy Configuration
- Displaying Load-Balancing Configuration Information and Statistics
- Clearing SLB Statistics
- Where to Go Next

Overview of SLB Traffic Policies

You classify inbound network traffic destined to, or passing through, the ACE based on a series of flow match criteria specified by a class map. Each class map defines a traffic classification, which is network traffic that is of interest to you. A policy map defines a series of actions (functions) that you want applied to a set of classified inbound traffic.

ACE traffic policies support the following server load-balancing (SLB) traffic attributes:

- Layer 3 and Layer 4 connection information—Source or destination IP address, source or destination port, virtual IP address, and IP protocol
- Layer 7 protocol information—Hypertext Transfer Protocol (HTTP) cookie, HTTP URL, HTTP header, Remote Authentication Dial-In User Service (RADIUS), Remote Desktop Protocol (RDP), Real-Time Streaming Protocol (RTSP), Session Initiation Protocol (SIP), and Secure Sockets Layer (SSL)

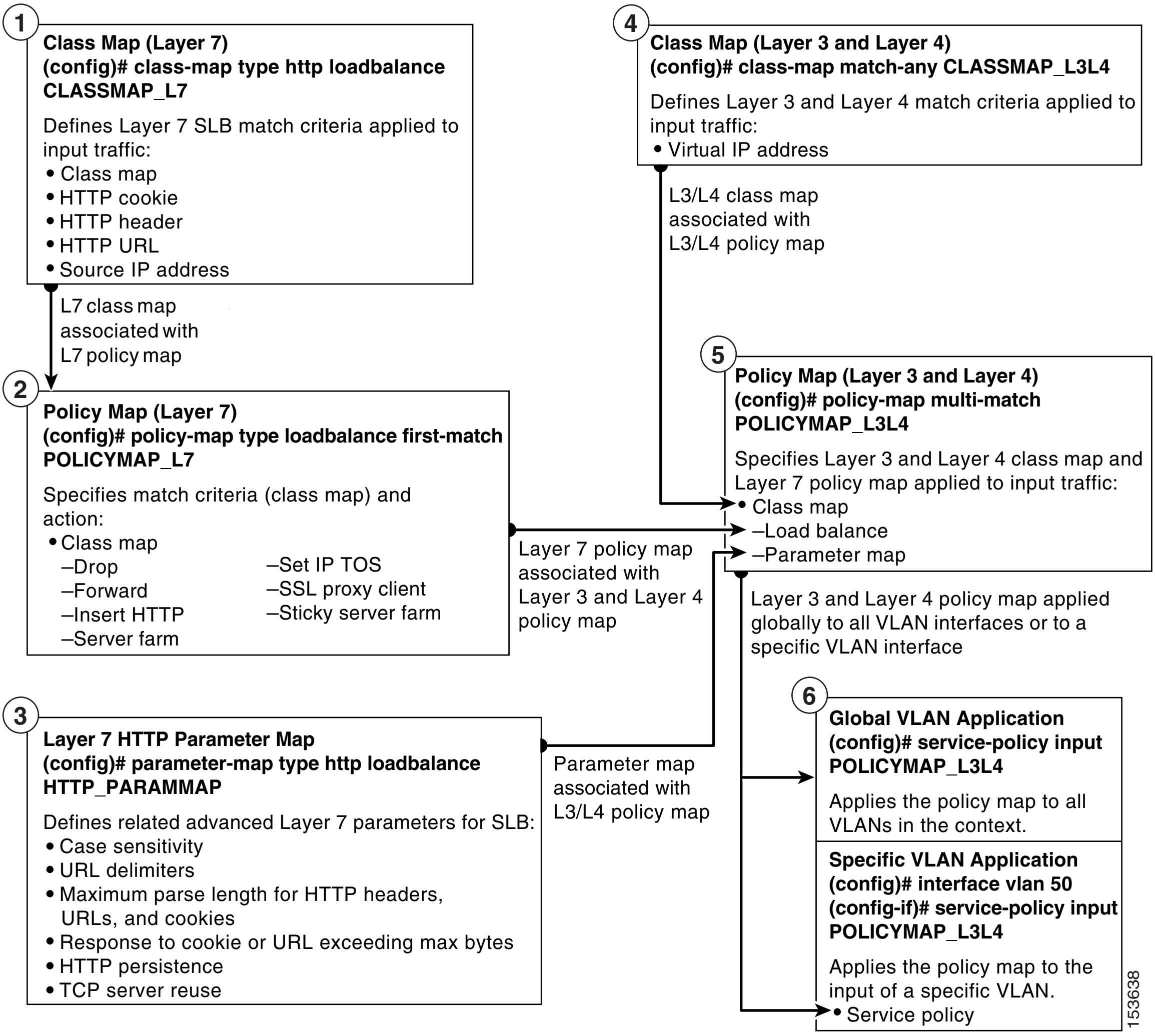
The three steps in the traffic classification process are as follows:

1. Create a class map using the **class-map** command and the associated **match** commands, which comprise a set of match criteria related to Layer 3 and Layer 4 traffic classifications or Layer 7 protocol classifications.

2. Create a policy map using the **policy-map** command, which refers to the class maps and identifies a series of actions to perform based on the traffic match criteria.
3. Activate the policy map by associating it with a specific VLAN interface or globally with all VLAN interfaces using the **service-policy** command to filter the traffic received by the ACE.

Figure 3-1 provides a basic overview of the process required to build and apply the Layer 3, Layer 4, and Layer 7 policies that the ACE uses for SLB. The figure also shows how you associate the various components of the SLB policy configuration with each other.

Figure 3-1 Server Load-Balancing Configuration Flow Diagram



1
Class Map (Layer 7)
(config)# class-map type http loadbalance CLASSMAP_L7
Defines Layer 7 SLB match criteria applied to input traffic:
• Class map
• HTTP cookie
• HTTP header
• HTTP URL
• Source IP address
L7 class map associated with L7 policy map
2
Policy Map (Layer 7)
(config)# policy-map type loadbalance first-match POLICYMAP_L7
Specifies match criteria (class map) and action:
• Class map
–Drop
–Forward
–Insert HTTP
–Server farm
–Set IP TOS
–SSL proxy client
–Sticky server farm
Layer 7 policy map associated with Layer 3 and Layer 4 policy map
3
Layer 7 HTTP Parameter Map
(config)# parameter-map type http loadbalance HTTP_PARAMMAP
Defines related advanced Layer 7 parameters for SLB:
• Case sensitivity
• URL delimiters
• Maximum parse length for HTTP headers, URLs, and cookies
• Response to cookie or URL exceeding max bytes
• HTTP persistence
• TCP server reuse
Parameter map associated with L3/L4 policy map
4
Class Map (Layer 3 and Layer 4)
(config)# class-map match-any CLASSMAP_L3L4
Defines Layer 3 and Layer 4 match criteria applied to input traffic:
• Virtual IP address
L3/L4 class map associated with L3/L4 policy map
5
Policy Map (Layer 3 and Layer 4)
(config)# policy-map multi-match POLICYMAP_L3L4
Specifies Layer 3 and Layer 4 class map and Layer 7 policy map applied to input traffic:
• Class map
–Load balance
–Parameter map
Layer 3 and Layer 4 policy map applied globally to all VLAN interfaces or to a specific VLAN interface
6
Global VLAN Application
(config)# service-policy input POLICYMAP_L3L4
Applies the policy map to all VLANs in the context.
Specific VLAN Application
(config)# interface vlan 50
(config-if)# service-policy input POLICYMAP_L3L4
Applies the policy map to the input of a specific VLAN.
• Service policy
153638

Layer 7 SLB Traffic Policy Configuration Quick Start

Table 3-1 provides a quick overview of the steps required to configure a Layer 7 HTTP class map and a Layer 7 HTTP policy map. You use a similar procedure to configure Layer 7 class maps and policy maps for other supported protocols. Each step includes the CLI command and 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.

Table 3-1 Layer 7 SLB Policy 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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.
`host1/Admin# config`
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

3. Create a Layer 7 class map for SLB. See the "Configuring a Layer 7 Class Map for SLB" section.
`host1/Admin(config)# class-map type http loadbalance match-all L7SLBCLASS`
`host1/Admin(config-cmap-http-lb)#`

Table 3-1 Layer 7 SLB Policy Configuration Quick Start (continued)

Task and Command Example

4. Configure one or more of the following match criteria for the Layer 7 SLB class map:

   - Define HTTP content for load balancing. See the "Defining an HTTP Content Match for Load Balancing" section.

     ```
     host1/Admin(config-cmap-http-lb)# match http content abc*123
     offset 50
     ```

   - Define a cookie for HTTP load balancing. See the "Defining a Cookie for HTTP Load Balancing" section.

     ```
     host1/Admin(config-cmap-http-lb)# match http cookie
     TESTCOOKIE1 cookie-value 123456
     ```

   - Define an HTTP header for load balancing. See the "Defining an HTTP Header for Load Balancing" section.

     ```
     host1/Admin(config-cmap-http-lb)# match http header Host
     header-value .*cisco.com
     ```

   - Define a URL for HTTP load balancing. See the "Defining a URL for HTTP Load Balancing" section.

     ```
     host1/Admin(config-cmap-http-lb)# match http url
     /WHATSNEW/LATEST.*
     ```

   - Define a source IP match statement. See the "Defining Source IP Address Match Criteria" section.

     ```
     host1/Admin(config-cmap-http-lb)# match source-address
     192.168.11.2 255.255.255.0
     ```

Table 3-1 Layer 7 SLB Policy Configuration Quick Start (continued)

Task and Command Example

5. Use the **exit** command to reenter configuration mode.

```
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)#
```

6. Create a Layer 7 policy map for SLB. See the "Configuring a Layer 7 Policy Map for SLB" section.

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

7. Associate the Layer 7 class map that you created in Step 3 with the Layer 7 policy map that you created in Step 6. See the "Associating a Layer 7 Class Map with a Layer 7 Policy Map" section.

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

8. Specify one or more of the following policy-map actions that you want the ACE to take when network traffic matches a class map:

   - Instruct the ACE to discard packets that match a policy map. See the "Discarding Requests" section.

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

   - Instruct the ACE to forward packets that match a policy map without load balancing them. See the "Forwarding Requests Without Load Balancing" section.

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

   - Enable HTTP header insertion. See the "Configuring HTTP Header Insertion" section.

     ```
     host1/Admin(config-pmap-lb-c)# insert-http Host header-value
     www.cisco.com
     ```

   - Enable load balancing to a server farm. See the "Enabling Load Balancing to a Server Farm" section.

     ```
     host1/Admin(config-pmap-lb-c)# serverfarm FARM2 backup FARM3
     ```

Table 3-1 Layer 7 SLB Policy Configuration Quick Start (continued)

Task and Command Example

- Specify the IP differentiated services code point (DSCP) of packets within the traffic class. See the "Configuring a Sticky Server Farm" section.

```
host1/Admin(config-pmap-lb-c)# set ip tos 8
```

- If you are using SSL Initiation (ACE acting as an SSL client), specify an SSL proxy service. See the "Specifying an SSL Proxy Service" section. For more information about SSL, see the *Cisco Application Control Engine Module SSL Configuration Guide*.

```
host1/Admin(config-pmap-lb-c)# ssl-proxy client
PROXY_SERVICE1
```

- To use stickiness (connection persistence), specify a sticky server farm for load balancing. See the "Configuring a Sticky Server Farm" section.

```
host1/Admin(config-pmap-lb-c)# sticky-serverfarm
STICKY_GROUP1
```

9. Before you can use a Layer 7 policy map for load balancing, you must associate it with a Layer 3 and Layer 4 SLB policy map. Create the Layer 3 and Layer 4 class map and policy map, then associate the Layer 7 policy map with the Layer 3 and Layer 4 policy map. Finally, associate the Layer 3 and Layer 4 policy map with an interface. See the following sections:

- Configuring a Layer 3 and Layer 4 Class Map for SLB
- Configuring a Layer 3 and Layer 4 Policy Map for SLB
- Applying a Layer 3 and Layer 4 Policy to an Interface

10. Display your class-map and policy-map configurations and statistics (see the "Displaying Load-Balancing Configuration Information and Statistics" section).

```
host1/Admin# show running-config class-map
host1/Admin# show running-config policy-map
```

11. (Optional) Save your configuration changes to flash memory.

```
host1/Admin# copy running-config startup-config
```

Layer 3 and Layer 4 SLB Traffic Policy Configuration Quick Start

Table 3-2 provides a quick overview of the steps required to configure a Layer 3 and Layer 4 class map and a Layer 3 and Layer 4 policy map. Each step includes the CLI command and 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 3-2.

Table 3-2 Layer 3 and Layer 4 SLB Policy Configuration Quick Start

Task and Command Example

1. If you are operating in multiple contexts, observe the CLI prompt to verify you are operating in the desired context. Change to, or directly log in to, the correct context if necessary.

   ```
   host1/Admin# changeto C1
   host1/C1#
   ```

   For details on creating contexts, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

2. Enter configuration mode.

   ```
   host1/Admin# config
   Enter configuration commands, one per line. End with CNTL/Z
   host1/Admin(config)#
   ```

3. Create a Layer 3 and Layer 4 SLB class map. See the "Configuring a Layer 3 and Layer 4 Class Map for SLB" section.

   ```
   host1/Admin(config)# class-map L4VIPCLASS
   host1/Admin(config-cmap)#
   ```

4. Define a virtual IP (VIP) address match statement. See the "Defining VIP Address Match Criteria" section.

   ```
   host1/Admin(config-cmap)# match virtual-address 192.168.1.10 tcp
   port eq 80
   ```

5. Reenter configuration mode.

   ```
   host1/Admin(config-cmap)# exit
   host1/Admin(config)#
   ```

Table 3-2 Layer 3 and Layer 4 SLB Policy Configuration Quick Start (continued)

Task and Command Example

6. Create a Layer 3 and Layer 4 policy map. See the "Configuring a Layer 3 and Layer 4 Policy Map for SLB" section.
`host1/Admin(config)# policy-map multi-match L4SLBPOLICY`
`host1/Admin(config-pmap)#`

7. Associate the Layer 3 and Layer 4 class map that you created in Step 2 with the policy map you created in Step 4. See the "Associating a Layer 3 and Layer 4 Class Map with a Policy Map" section.
`host1/Admin(config-pmap)# class L4VIPCLASS`
`host1/Admin(config-pmap-c)#`

8. Specify one or more of the following policy-map actions that you want the ACE to take when network traffic matches a class map:
• Enable the ACE to advertise the IP address of a virtual server as the host route. See the "Enabling the Advertising of a Virtual Server IP Address" section.
`host1/Admin(config-pmap-c)# loadbalance vip advertise active`
• Enable a VIP to reply to ICMP ECHO requests. For example, if a user sends an ICMP ECHO request to a VIP, this command instructs the VIP to send an ICMP ECHO-REPLY. See the "Enabling a VIP to Reply to ICMP Requests" section.
`host1/Admin(config-pmap-c)# loadbalance vip icmp-reply`
• Associate a Layer 7 SLB policy map with the Layer 3 and Layer 4 policy map to provide an entry point for Layer 7 classifications. See the "Associating a Layer 7 SLB Policy Map with a Layer 3 and Layer 4 SLB Policy Map" section.
`host1/Admin(config-pmap-c)# loadbalance policy L7SLBPOLICY`
• Enable a VIP for SLB operations.
`host1/Admin(config-pmap-c)# loadbalance vip inservice`

Table 3-2 Layer 3 and Layer 4 SLB Policy Configuration Quick Start (continued)

Task and Command Example

9. Activate a policy map and attach it to an interface. See the "Applying a Layer 3 and Layer 4 Policy to an Interface" section.
`host1/Admin(config)# interface VLAN50`
`host1/Admin(config-if)# service-policy input L4SLBPOLICY`
`host1/Admin(config-if)# Ctrl-z`

10. Display your class-map and policy-map configurations and statistics (see the "Displaying Load-Balancing Configuration Information and Statistics" section).
`host1/Admin# show running-config class-map`
`host1/Admin# show running-config policy-map`
`host1/Admin# show service-policy name [detail]`

11. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Configuring HTTP Header Insertion, Deletion, and Rewrite

This section describes action lists and how to use them to insert, rewrite, and delete HTTP headers. An action list is a named group of actions that you associate with a Layer 7 HTTP class map in a Layer 7 HTTP policy map. You can create an action list to modify an HTTP header by using the **action-list type modify http** command in configuration mode. The syntax of this command is as follows:

action-list type modify http *name*

For the *name* argument, enter a unique name for the action list as an unquoted text string with a a maximum of 64 alphanumeric characters.

For example, enter:

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

To remove the action list from the configuration, enter:

```
host1/Admin(config)# no action-list type modify http
HTTP_MODIFY_ACTLIST
```

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

The following sections describe the HTTP actions that you can put in an action list:

- Configuring HTTP Header Insertion
- Configuring HTTP Header Rewrite
- Configuring HTTP Header Deletion

After you create an action list and associate actions with it, you must associate the action list with a Layer 7 policy map. For details, see the "Associating an Action List with a Layer 7 Policy Map" section.

For information about rewriting an HTTP redirect URL for SSL, see the *Cisco Application Control Engine Module SSL Configuration Guide*.

Configuring HTTP Header Insertion

When the ACE uses Network Address Translation (NAT) to translate the source IP address of a client to a VIP, servers need a way to identify that client for the TCP and IP return traffic. To identify a client whose source IP address has been translated using NAT, you can instruct the ACE to insert a generic header and string value of your choice in the client HTTP request. (For information about NAT, see the *Cisco Application Control Engine Module Security Configuration Guide*.)

Note With either TCP server reuse or persistence rebalance enabled, the ACE inserts a header in every client request. For information about TCP server reuse, see the "Configuring TCP Server Reuse" section. For information about persistence rebalance, see the "Configuring HTTP Persistence Rebalance" section.

You can insert a header name and value in an HTTP request from a client, a response from a server, or both, by using the **header insert** command in action list modify configuration mode. The syntax of this command is as follows:

header insert {**request** | **response** | **both**} *header_name* **header-value** *expression*

The keywords, options, and arguments are as follows:

- **request**—Specifies that the ACE insert an HTTP header only in HTTP request packets from clients.
- **response**—Specifies that the ACE insert an HTTP header only in HTTP response packets from servers.
- **both**—Specifies that the ACE insert an HTTP header in both HTTP request packets and response packets.
- *header_name*—Identifier of an HTTP header. Enter an unquoted text string with a maximum of 255 alphanumeric characters.
- **header-value** *expression*—Specifies the value of the HTTP header that you want to insert in request packets, response packets, or both. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. You can also use the following dynamic replacement strings:
  - **%is**—Inserts the source IP address in the HTTP header
  - **%id**—Inserts the destination IP address in the HTTP header
  - **%ps**—Inserts the source port in the HTTP header
  - **%pd**—Inserts the destination port in the HTTP header

  The ACE supports the use of regular expressions (regexes) for matching data strings. Table 3-3 lists the supported characters that you can use in regular expressions. Use parenthesized expressions for dynamic replacement using %1 and %2 in the replacement pattern.

  Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

Table 3-3 Special Characters for Matching String Expressions

Convention	Description
.	One of any character.
.*	Zero or more of any character.

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

Convention	Description
\.	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 0 to 255.
expr{m}	Match the expression exactly *m* times. The range for *m* is from 0 to 255.
expr{m,}	Match the expression *m* or more times. The range for *m* is from 0 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.
\xST	Stop metacharacter. For information on the use of this metacharacter, see the "Using the "\xST" Metacharacter in Regular Expressions for Layer 4 Generic Data Parsing" section.

For example, to insert a Host: source_ip:source_port in both the client request and the server response headers, enter:

```
host1/Admin(config)# action-list type modify http HTTP_MODIFY_ACTLIST
host1/Admin(config-actlist-mod)# header insert both Host header-value
%is:%ps
```

To associate the action list with a Layer 7 load-balancing policy map, enter:

```
host1/Admin(config)# policy-map type loadbalance http first-match
L7_POLICY
host1/Admin(config-pmap-lb)# class L7_CLASS
host1/Admin(config-pmap-lb-c)# serverfarm sf1
host1/Admin(config-pmap-lb-c)# action HTTP_MODIFY_ACTLIST
```

To remove the **header insert** command from the action list, enter:

```
host1/Admin(config-actlist-mod)# no header insert both Host
header-value %is:%ps
```

Configuring HTTP Header Rewrite

You can rewrite an HTTP header in request packets from a client, response packets from a server, or both, by using the **header rewrite** command in action list modify configuration mode. The syntax of this command is as follows:

header rewrite {**request** | **response** | **both**} *header_name* **header-value** *expression* **replace** *pattern*

The keywords and arguments are as follows:

- **request**—Specifies that the ACE rewrite an HTTP header string only in HTTP request packets from clients
- **response**—Specifies that the ACE rewrite an HTTP header string only in HTTP response packets from servers
- **both**—Specifies that the ACE rewrite an HTTP header string in both HTTP request packets and response packets
- *header_name*—Identifier of an HTTP header. Enter an unquoted text string with a maximum of 255 alphanumeric characters.

- **header-value** *expression*—Specifies the value of the HTTP header that you want to replace in request packets, response packets, or both. Enter a text string from 1 to 255 alphanumeric characters. The ACE supports the use of regexes for matching data strings. See Table 3-3 for a list of the supported characters that you can use in regular expressions. Use parenthesized expressions for dynamic replacement using %1 and %2 in the replacement pattern.

  Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

- **replace** *pattern*—Specifies the pattern string that you want to substitute for the header value regular expression. For dynamic replacement of the first and second parenthesized expressions from the header value, use %1 and %2, respectively.

For example, to replace www.cisco.com with www.cisco.net, enter:

```
host1/Admin(config)# action-list type modify http HTTP_MODIFY_ACTLIST
host1/Admin(config-actlist-mod)# header rewrite request Host 
header-value www\.(cisco)\.com replace www.%1.net
```

To associate the action list with a Layer 7 load-balancing policy map, enter:

```
host1/Admin(config)# policy-map type loadbalance http first-match 
L7_POLICY
host1/Admin(config-pmap-lb)# class L7_CLASS
host1/Admin(config-pmap-lb-c)# serverfarm sf1
host1/Admin(config-pmap-lb-c)# action HTTP_MODIFY_ACTLIST
```

To remove the **header rewrite** command from the action list, enter:

```
host1/Admin(config-actlist-mod)# no header rewrite request Host 
header-value www\.(cisco)\.com replace www.%1.net
```

Configuring HTTP Header Deletion

You can delete an HTTP header in a request from a client, in a response from a server, or both, by using the **header delete** command in action list modify configuration mode. The syntax of this command is as follows:

header delete {**request** | **response** | **both**} *header_name*

The keywords and arguments are as follows:

- **request**—Specifies that the ACE delete the header only in HTTP request packets from clients
- **response**—Specifies that the ACE delete the header only in HTTP response packets from servers
- **both**—Specifies that the ACE delete the header in both HTTP request packets and response packets
- *header_name*—Identifier of an HTTP header that you want to delete. Enter an unquoted text string with a maximum of 255 alphanumeric characters.

For example, to delete the Host header from request packets only, enter:

```
host1/Admin(config)# action-list type modify http HTTP_MODIFY_ACTLIST
host1/Admin(config-actlist-mod)# header delete request Host
```

To associate the action list with a Layer 7 load-balancing policy map, enter:

```
host1/Admin(config)# policy-map type loadbalance http first-match
L7_POLICY
host1/Admin(config-pmap-lb)# class L7_CLASS
host1/Admin(config-pmap-lb-c)# serverfarm sf1
host1/Admin(config-pmap-lb-c)# action HTTP_MODIFY_ACTLIST
```

To remove the **header delete** command from the action list, enter:

```
host1/Admin(config-actlist-mod)# no header delete request Host
```

Configuring a Layer 7 Class Map for Generic TCP and UDP Data Parsing

You can use generic TCP and UDP data parsing to perform regular expression (regex) matches on packets from protocols that the ACE does not explicitly support. Such regex matches can be based on a custom protocol configuration. To accomplish this task, you create a Layer 7 class map for generic TCP or UDP data parsing and then instruct the ACE to perform a policy-map action based on the payload of a TCP stream or UDP packet.

To avoid using a large amount of memory with regular expressions, we recommend the following guidelines when you configure generic data parsing:

- Use only one generic rule per VIP
- Use the same offset for all generic rules on the same VIP
- Use the smallest possible offset that will work for your application
- Avoid deploying Layer 4 payload stickiness (see Chapter 5, Configuring Stickiness) and Layer 4 payload matching simultaneously, when possible

Note The **persistence-rebalance** command is not compatible with generic protocol parsing.

You can create a class map for generic TCP or UDP data parsing by using the **class-map type generic** command in configuration mode. The syntax of this command is as follows:

class-map type generic match-all | match-any *name*

The keywords and arguments are as follows:

- **generic**—Specifies nonprotocol-specific behavior for data parsing
- **match-all | match-any**—(Optional) Determines how the ACE evaluates Layer 3 and Layer 4 network traffic when multiple match criteria exist in a class map. The class map is considered a match if the **match** commands meet one of the following conditions.
  - **match-all**—(Default) Network traffic needs to satisfy all of the match criteria (implicit AND) to match the class map.

  - **match-any**—Network traffic needs to satisfy only one of the match criteria (implicit OR) to match the load-balancing class map.

- *name*—Name assigned to the class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. The name is used for both the class map and to associate the class map with a policy map.

For example, enter:

```
host1/Admin(config)# class-map type generic match-any GENERIC_L7_CLASS
```

To remove the class map from the configuration, enter:

```
host1/Admin(config)# no class-map type generic match-any
GENERIC_L7_CLASS
```

After you create a class map for generic protocol parsing, configure one or more match statements as described in the following sections:

- Defining Layer 4 Payload Match Criteria for Generic Data Parsing
- Using the "\xST" Metacharacter in Regular Expressions for Layer 4 Generic Data Parsing
- Defining Source IP Address Match Criteria

Defining Layer 4 Payload Match Criteria for Generic Data Parsing

Generic data parsing begins at Layer 4 with the TCP or UDP payload, which allows you to match Layer 5 data (in the case of Lightweight Directory Access Protocol (LDAP) or DNS or any Layer 7 header or payload (for example, HTTP). You can define match criteria for Layer 4 payloads by using the **match layer4-payload** command in class-map generic configuration mode. The syntax of this command is as follows:

[*line_number*] **match layer4-payload** [**offset** *number*] | **regex** *expression*

Note You cannot configure more than one **match layer4-payload** command in the same **match-all** class map.

The keywords, options, and arguments are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The sequence numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- **offset** *number*—(Optional) Specifies an absolute offset in the data where the Layer 4 payload expression search string starts. The offset starts at the first byte of the TCP or UDP body. Enter an integer from 0 to 999. The default is 0.
- **regex** *expression*—Specifies the Layer 4 payload expression that is contained within the TCP or UDP entity body. The ACE supports the use of regexes for matching data strings. See Table 3-3 for a list of the supported characters that you can use in regular expressions. Use parenthesized expressions for dynamic replacement using %1 and %2 in the replacement pattern.

  For information on the use of the "\xST" (STop) metacharacter for regular expressions, see the "Using the "\xST" Metacharacter in Regular Expressions for Layer 4 Generic Data Parsing" section.

  Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

For example, to create a class map for generic Layer 4 data parsing, enter:

```
host1/Admin(config)# class-map type generic match-any GENERIC_L7_CLASS
host1/Admin(config-cmap-generic)# 10 match layer4-payload offset 500
regex abc123
```

To remove the match statement from the class map, enter:

```
host1/Admin(config-cmap-generic)# no 10
```

Using the "\xST" Metacharacter in Regular Expressions for Layer 4 Generic Data Parsing

This section describes the use of the "\xST" metacharacter for regular expressions that are used as part of Layer 4 generic data parsing. It includes the following topics:

- Overview
- "\xST" Metacharacter Regex Usage Considerations
- Configuration Examples

Overview

The ACE supports the "\xST" (STop) metacharacter for all regular expressions (regexes) in specific cases that use the maximum parse length to terminate parsing. However, the "\xST" metacharacter is specifically for use by applications that involve the generic data parsing of a Layer 4 payload.

If you intend to use the "\xST" metacharacter for regex matches on packets from protocols, we recommend that you use this metacharacter only for the following protocols in the generic data parsing of a Layer 4 payload:

- SSL session-ID stickiness—To perform sticky hashing on the initial packets in an SSL handshake, allowing the ACE to stick the same client to the same SSL server based on the SSL session ID.
- Financial Information eXchange (FIX) type 'A' Logon message—To define load-balancing criteria while setting up the outbound path of a connection.

The inclusion of the “\xST” metacharacter aids the ACE in properly load-balancing SSL session-ID and FIX packets. Without the “\xST” metacharacter in regexes, certain SSL session-ID and FIX packets may get stuck in the ACE HTTP engine and eventually time out the connection.

“\xST” Metacharacter Regex Usage Considerations

The “\xST” metacharacter has the following usage guidelines related to its inclusion in regex matching:

- If the input matches a regex pattern that includes the “\xST” metacharacter, the regex engine halts upon finding the character directly next to the '\xST' in the regex string (2nd '\x01' in the match statement).
- Only use the “\xST” metacharacter once in the policy. The ACE does not consider any additional input data once it sees the matching pattern ,which may affect other regexes that are configured elsewhere in the policy.
- Only use the “\xST” metacharacter at the end of a regex pattern; not at the beginning. Otherwise, the ACE will display the “Error: Invalid regular expression” error message.
- Do not add the “\xST” metacharacter directly after a * wildcard match. For example, “abc.*\xST” is not be a recommended regex.

Configuration Examples

The following configuration examples show the use of the “\xST” metacharacter in two very specific regexes:

SSL Session-ID Stickiness Configuration Example

```
parameter-map type generic SESSID-PARAM

set max-parse-length 76

sticky layer4-payload SESSID-STICKY
      serverfarm SF1
      response sticky
      layer4-payload offset 43 length 32 begin-pattern
    "(\x20|\x00\xST)"
```

FIX Protocol Configuration Example

```
sticky layer4-payload FIX-STICKY
        serverfarm FIX-SF1
        layer4-payload begin-pattern "\x0149=" end-pattern "\x01"

class-map type generic match-all FIX-CM
    2 match layer4-payload regex ".*\x0110=...\x01\xST"
```

Defining Source IP Address Match Criteria

You can configure the class map to filter traffic based on a client source IP address by using the **match source-address** command in class map generic configuration mode. If this command is the only match criteria in the class map, it is considered to be a Layer 3 and Layer 4 class map.

The syntax of this command is as follows:

[*line_number*] **match source-address** *ip_address* [*netmask*]

The arguments and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- *ip_address—*Source IP address of the client. Enter the IP address in dotted-decimal notation (for example, 192.168.11.2).
- *netmask*—(Optional) Subnet mask of the IP address. Enter the netmask in dotted-decimal notation (for example, 255.255.255.0). The default is 255.255.255.255.

Note You cannot configure more than one **match source-address** command in the same **match-all** class map.

For example, to specify that the class map match on source IP address 192.168.11.2 255.255.255.0, enter:

```
host1/Admin(config)#  class-map type generic match-any
GENERIC_L4_CLASS
host1/Admin(config-cmap-generic)# 50 match source-address 192.168.11.2
255.255.255.0
```

To remove the source IP address match statement from the class map, enter:

```
host1/Admin(config-cmap-generic)# no 50
```

Nesting Layer 7 SLB Class Maps

The nesting of class maps allows you to achieve complex logical expressions for generic parsing.You can identify one generic class map that is to be used as a matching criterion for another generic class map by using the **match class-map** command in class-map generic configuration mode.

Note The ACE restricts the nesting of class maps to two levels to prevent you from including a nested class map under another class map.

The syntax of this command is as follows:

[*line_number*] **match class-map** *map_name*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line.
- *map_name*—Name of an existing generic class map.

The **match class-map** command allows you to combine the use of the **match-any** and **match-all** keywords in the same class map. To combine **match-all** and **match-any** characteristics in a class map, create a class map that uses one **match** command (either **match any** or **match all**), and then use this class map as a match statement in a second class map that uses a different match type.

For example, assume that commands A, B, C, and D represent separate match criteria, and you want generic protocol traffic that matches A, B, or C and D (A or B or [C and D]) to satisfy the class map. Without the use of nested class maps, traffic would either have to match all four match criteria (A and B and C and D) or match any of the match criteria (A or B or C or D) to satisfy the class map. By creating a single class map that uses the **match-all** keyword for match criteria C and D (criteria E), you can then create a new **match-any** class map that uses match criteria A, B, and E. The new traffic class contains your desired classification sequence: A or B or E, which is equivalent to A or B or [C and D].

For example, to combine the characteristics of two class maps, one with **match-any** and one with **match-all** characteristics, into a single class map by using the **match class-map** command, enter:

```
host1/Admin(config)#  class-map type generic match-any
GENERIC_L4_CLASS
host1/Admin(config-cmap-generic)# 50 match source-address 192.168.11.2
255.255.255.0
host1/Admin(config-cmap-generic)# exit

host1/Admin(config)# class-map type generic match-all GENERIC_L4_CLASS
host1/Admin(config-cmap-generic)# 10 match class-map GENERIC_L4_CLASS2
host1/Admin(config-cmap-generic)# 20 match source-address 192.168.11.2
host1/Admin(config-cmap-generic)# exit
```

To remove the nested class map from the generic class map, enter:

```
host1/Admin(config-cmap-generic)# no 10
```

Configuring a Layer 7 Class Map for SLB

A Layer 7 SLB class map contains match criteria that classify specific Layer 7 network traffic. This section describes how to create a class map for Layer 7 SLB based on HTTP cookies, HTTP headers, HTTP URLs, RADIUS attributes, RDP, RTSP headers or URLs, SIP headers, or source IP addresses.

You can create a Layer 7 class map for SLB and enter the class-map configuration mode by using the **class-map type** command in configuration mode. The syntax of this command is as follows:

class-map type {{**http** | **radius** | **rtsp** | **sip**} **loadbalance**} [**match-all** | **match-any**] *map_name*

You can configure multiple **match** commands in a single class map to specify the matching criteria. For example, you can configure a Layer 7 load-balancing class map to define multiple URLs, cookies, and HTTP headers in a group that you then associate with a traffic policy. The **match-all** and **match-any** keywords determine how the ACE evaluates multiple match statement operations when multiple match criteria exist in a class map.

The keywords, arguments, and options are as follows:

- **http**—Specifies a Hypertext Transfer Protocol (HTTP) load-balancing class map. This is the default.
- **radius**—Specifies the Remote Access Dial-In User Service (RADIUS) protocol for load balancing.
- **rtsp**—Specifies the Real-Time Streaming Protocol (RTSP) for load balancing.
- **sip**—Specifies the Session Initiation Protocol (SIP) for load balancing.
- **loadbalance**—Specifies a load-balancing type class map.
- **match-all | match-any**—(Optional) Determines how the ACE evaluates Layer 7 HTTP SLB operations when multiple match criteria exist in a class map. The class map is considered a match if the match commands meet one of the following conditions:
  - **match-all** —(Default) Network traffic needs to satisfy all of the match criteria (implicit AND) to match the Layer 7 load-balancing class map. The **match-all** keyword is applicable only for match statements of different Layer 7 load-balancing types. For example, specifying a **match-all** condition for URL, HTTP header, and URL cookie statements in the same class map is valid. However, specifying a **match-all** condition for multiple HTTP headers or multiple cookies with the same names or multiple URLs in the same class map is invalid.
  - **match-any**—Network traffic needs to satisfy only one of the match criteria (implicit OR) to match the HTTP load-balancing class map. The **match-any** keyword is applicable only for match statements of the same Layer 7 load-balancing type. For example, the ACE does not allow you to specify a **match-any** condition for URL, HTTP header, and URL cookie statements in the same class map but does allow you to specify a **match-any** condition for multiple URLs, multiple HTTP headers, or multiple cookies with different names in the same class map.

- *map_name*—Unique identifier assigned to the class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. The class-map name is used for both the class map and to associate the class map with a policy map.

For example, to create a Layer 7 load-balancing class map named L7SLBCLASS, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)#
```

To remove a Layer 7 load-balancing class map from the configuration, enter:

```
host1/Admin(config)# no class-map type http loadbalance match-any
L7SLBCLASS
```

The following topics describe how to specify match criteria for the Layer 7 class map:

- Configuration Considerations
- Defining an HTTP Content Match for Load Balancing
- Defining a Cookie for HTTP Load Balancing
- Defining an HTTP Header for Load Balancing
- Defining a URL for HTTP Load Balancing
- Defining an Attribute for RADIUS Load Balancing
- Defining a Header for RTSP Load Balancing
- Defining a URL for RTSP Load Balancing
- Defining a Header for SIP Load Balancing
- Defining Source IP Address Match Criteria
- Nesting Layer 7 SLB Class Maps

Configuration Considerations

When you are creating a class map for SLB, note the following restrictions:

- You can associate a maximum of 10 cookie names and header names with each Layer 3 and Layer 4 policy map. You can allocate the number of cookie names and header names in any combination as long as you do not exceed the maximum of 10.
- You can associate a maximum of 1024 instances of the same type of regex with each Layer 3 and Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:
  - Match statements in Layer 7 class maps
  - Inline match statements in Layer 7 policy maps
  - Layer 7 hash predictors for server farms
  - Layer 7 sticky expressions in sticky groups
  - Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists
- The ACE restricts the nesting of class maps to two levels to prevent you from including one nested class map in a different class map.
- The maximum number of class maps for each ACE is 8192.

Defining an HTTP Content Match for Load Balancing

The ACE performs regular expression matching against the received HTTP message body from a particular connection based on a regular expression string in the message body (not the header). To configure the class map to make Layer 7 SLB decisions based on the HTTP content, use the **match http content** command in class-map configuration mode. The syntax of this command is as follows:

[*line_number*] **match http content** *expression* [**offset** *number*]

The arguments and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- *expression*—The regular expression content to match. Enter a string from 1 to 255 alphanumeric characters. The ACE supports the use of regular expressions for matching data strings. See Table 3-3 for a list of the supported characters that you can use in regular expressions.

  Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

- **offset** *number*—(Optional) Specifies the byte at which the ACE begins parsing the message body. Enter an integer from 0 to 999. The default is 0.

For example, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any
L7_HTTP_CLASS
host1/Admin(config-cmap-http-lb)# 10 match http content abc*123 offset
50
```

Defining a Cookie for HTTP Load Balancing

The ACE performs regular expression matching against the received packet data from a particular connection based on the cookie expression. You can configure a maximum of 10 cookie names and header names per class in any combination. You can configure the class map to make Layer 7 SLB decisions based on the name and string of a cookie by using the **match http cookie** command in class-map configuration mode. The syntax of this command is as follows:

[*line_number*] **match http cookie** {*name* | **secondary** *name*} **cookie-value** *expression*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The sequence numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- *name*—Unique cookie name. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

Note If certain characters are used in the cookie name, such as an underscore (_), hyphen (-), period (.), or semicolon (:), replace those characters with the equivalent percent encoding (% HEX HEX) characters. For example, to configure the cookie name Regex_MatchCookie, replace the underscore (_) character with the equivalent %5F percent encoding character and enter the cookie name as **Regex%5FMatchCookie** in the CLI.

- **secondary** *name*—Specifies a cookie in a URL string. You can specify the delimiters for cookies in a URL string using a command in an HTTP parameter map. For more information, see the "Defining URL Delimiters" section.
- **cookie-value** *expression*—Specifies a unique cookie value regular expression. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. Alternatively, you can enter a text string with spaces provided that you enclose the entire string in quotation marks ("). The ACE supports the use of regular expressions for matching string expressions. See Table 3-3 for a list of the supported characters that you can use for matching string expressions.

  Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

For example, to specify that the Layer 7 class map load balance on a cookie with the name of testcookie1, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 100 match http cookie testcookie1
cookie-value 123456
```

To remove an HTTP cookie match statement from the class map, enter:

```
host1/Admin(config-cmap-http-lb)# no 100
```

Defining an HTTP Header for Load Balancing

The ACE performs regular expression matching against the received packet data from a particular connection based on the HTTP header expression. You can configure a maximum of 10 HTTP header names and cookie names per class in any combination. To configure a class map to make Layer 7 SLB decisions based on the name and value of an HTTP header, use the **match http header** command in class-map HTTP load balance configuration mode.

The syntax of this command is as follows:

[*line_number*] **match http header** *name* **header-value** *expression*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The sequence numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- *name*—Name of the field in the HTTP header. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). You can enter any header field name, including a standard HTTP header field name or any user-defined header field name. For a list of standard HTTP header field names, see Table 3-4.

- **header-value** *expression*—Specifies the header value regular expression string to compare against the value in the specified field in the HTTP header. Enter a text string with a maximum of 255 alphanumeric characters. The ACE supports the use of regular expressions for header matching. Expressions are stored in a header map in the form *header-name: expression*. Header expressions allow spaces, provided that the entire string that contains spaces is quoted. If you use a **match-all** class map, all headers in the header map must be matched. See Table 3-3 for a list of the supported characters that you can use in regular expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

Table 3-4 Standard HTTP Header Fields

Field Name	Description
Accept	Semicolon-separated list of representation schemes (content type metainformation values) that will be accepted in the response to the request.
Accept-Charset	Character sets that are acceptable for the response. This field allows clients capable of understanding more comprehensive or special-purpose character sets to signal that capability to a server that can represent documents in those character sets.
Accept-Encoding	Restricts the content encoding that a user will accept from the server.
Accept-Language	ISO code for the language in which the document is written. The language code is an ISO 3316 language code with an optional ISO639 country code to specify a national variant.
Authorization	Specifies that the user agent wants to authenticate itself with a server, usually after receiving a 401 response.

Table 3-4 Standard HTTP Header Fields (continued)

Field Name	Description
Cache-Control	Directives that must be obeyed by all caching mechanisms along the request/response chain. The directives specify behavior intended to prevent caches from adversely interfering with the request or response.
Connection	Allows the sender to specify connection options.
Content-MD5	MD5 digest of the entity-body that provides an end-to-end integrity check. Only a client or an origin server can generate this header field.
Expect	Used by a client to inform the server about what behaviors the client requires.
From	E-mail address of the person that controls the requesting user agent.
Host	Internet host and port number of the resource being requested, as obtained from the original URI given by the user or referring resource. The Host field value must represent the naming authority of the origin server or gateway given by the original URL.
If-Match	Used with a method to make it conditional. A client that has one or more entities previously obtained from the resource can verify that one of those entities is current by including a list of their associated entity tags in the If-Match header field. This feature allows efficient updates of cached information with a minimum amount of transaction overhead. It is also used on updating requests to prevent inadvertent modification of the wrong version of a resource. As a special case, the value "*" matches any current entity of the resource.

Table 3-4 Standard HTTP Header Fields (continued)

Field Name	Description
Pragma	Pragma directives understood by servers to whom the directives are relevant. The syntax is the same as for other multiple-value fields in HTTP, for example, the **accept** field, a comma-separated list of entries, for which the optional parameters are separated by semicolons.
Referer	Address (URI) of the resource from which the URI in the request was obtained.
Transfer-Encoding	What (if any) type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient.
User-Agent	Information about the user agent, for example, a software program originating the request. This information is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents to customize responses to avoid particular user agent limitations.
Via	Used by gateways and proxies to indicate the intermediate protocols and recipients between the user agent and the server on requests and between the origin server and the client on responses.

For example, to specify that the Layer 7 class map load balance on an HTTP header named Host, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 100 match http header Host
header-value .*cisco.com
```

For example, to use regular expressions in a class map to emulate a wildcard search to match the header value expression string, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 10 match http header Host
header-value .*cisco.com
host1/Admin(config-cmap-http-lb)# 20 match http header Host
header-value .*yahoo.com
```

For example, to specify that the Layer 7 class map load balance on an HTTP header named Via, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 200 match http header Via
header-value 192.*
```

To remove HTTP header match criteria from the L7SLBCLASS class map, enter:

```
host1/Admin(config-cmap-http-lb)# no 10
host1/Admin(config-cmap-http-lb)# no 20
```

Defining a URL for HTTP Load Balancing

The ACE performs regular expression matching against the received packet data from a particular connection based on the HTTP URL string. To configure a class map to make Layer 7 SLB decisions based on the URL name and, optionally, the HTTP method, use the **match http url** command in class-map HTTP load balance configuration mode. The syntax of this command is as follows:

[*line_number*] **match http url** *expression* [**method** *name*]

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The *line_number* line numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.

- *expression*—URL, or portion of a URL, to match. Enter a URL string from 1 to 255 alphanumeric characters. The ACE performs matching on whatever URL string appears after the HTTP method, regardless of whether the URL includes the hostname. The ACE supports the use of regular expressions for matching URL strings. See Table 3-3 for a list of the supported characters that you can use in regular expressions.

  Note When matching URLs, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

- **method** *name*—(Optional) Specifies the HTTP method to match. Enter a method name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. The method can either be one of the standard HTTP 1.1 method names (OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, or CONNECT) or a text string that must be matched exactly (for example, CORVETTE).

We recommend that you use the “/.*” regex to match HTTP URLs. If you use the “.*” regex only, the ACE may pass requests that do not conform to RFC syntax (see RFC 2396). Web servers typically respond to such invalid requests with a 400 error. An example of an invalid request is “GET index.html HTTP/1.1”, whereas “GET /index.html HTTP/1.1” is valid. Using the "/.*" regex will match all valid URLs that do not have a host name in the URI, which is rare.

Configuring the "/.*" regex does exclude matches for URIs that begin with “http://”, which, in some scenarios, may not be desirable. However, such requests are not expected to be seen in a production environment because only some web proxies exhibit this behavior and not clients. To match generic requests with a hostname in the URI, include a statement such as **match http url http://.*** with the **match http url /.*** statement in a match-any type class map. This combination of regular expressions in different match statements will rule out a URI that does not include a preceding “/” (forward slash) character.

Using the configured **match http url** *expression* command, the ACE attempts to match a URL in an HTTP request following the first space after the request method. For example, if the request were "GET /index.html HTTP/1.1", the ACE tries to match starting with the "/" character. Therefore, the class-map match statement is conformant with the RFCs that cover HTTP URI syntax. This syntax includes request URIs that start either with "/" or with "*scheme*://*authority*/" (for example, http://www.cisco.com/).

To specify that the Layer 7 class map load balance on a specific URL, enter:

```
host1/Admin(config)# class-map type http loadbalance L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 10 match http url /whatsnew/latest.*
```

To use regular expressions to emulate a wildcard search to match on any .gif or .html file, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 100 match http url .*.gif
host1/Admin(config-cmap-http-lb)# 200 match http url .*.html
```

To remove a URL match statement from the L7SLBCLASS class map, enter **no** and the line number. For example, to remove line 100, enter:

```
host1/Admin(config-cmap-http-lb)# no 100
```

Note If you did not use line numbers to enter the original URL match statement, you can obtain the line number from your running configuration. To display the running configuration, enter **show running-config**.

Defining an Attribute for RADIUS Load Balancing

The ACE performs Layer 7 RADIUS load balancing based on the calling-station-ID or username RADIUS attributes. After you configure a Layer 7 class map (see the "Configuring a Layer 7 Class Map for SLB" section), you can specify Layer 7 RADIUS match criteria by using the **match radius attribute** command in class map RADIUS load balance configuration mode. The syntax of this command is as follows:

match radius attribute {**calling-station-id** | **username**} *expression*

The keywords and arguments are as follows:

- **calling-station-id**—Specifies the unique identifier of the calling station.
- **username**—Specifies the name of the RADIUS user who initiated the connection.
- *expression*—The calling station ID or username to match. Enter a string from 1 to 64 alphanumeric characters. The ACE supports the use of regular expressions for matching strings. See Table 3-3 for a list of the supported characters that you can use in regular expressions.

Note A match-all class map cannot have more than one same type match, while a match-any class map cannot have more than one different type match.

For example, to configure RADIUS match criteria based on the calling station ID, enter:

```
host1/Admin(config)# class-map type radius loadbalance match-any
RADIUS_L7_CLASS
host1/Admin(config-cmap-radius-lb)# 10 match radius attribute
calling-station-id 122*
```

To remove the RADIUS attribute match statement from the RADIUS_L7_CLASS class map, enter **no** and the line number. For example, to remove line 10, enter:

```
host1/Admin(config-cmap-radius-lb)# no 10
```

Defining a Header for RTSP Load Balancing

The ACE performs regular expression matching against the received packet data from a particular connection based on the RTSP header expression. You can configure a maximum of 10 RTSP header names per class.

Note When the ACE receives an RTSP session request, the load-balancing decision is based on the first request message. All subsequent request and response message exchanges are forwarded to the same server. When you configure header match criteria, ensure that the header is included in the first request message by a media player.

You can configure a class map to make Layer 7 SLB decisions based on the name and value of an RTSP header by using the **match rtsp header** command in class-map RTSP load balance configuration mode.

The syntax of this command is as follows:

[*line_number*] **match rtsp header** *name* **header-value** *expression*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The sequence numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- *name*—Name of the field in the RTSP header. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). You can enter any header field name, including a standard RTSP header field name or any user-defined header field name.

Note RTSP is intentionally similar in syntax and operation to HTTP/1.1, so you can use any HTTP header defined in Table 3-4 if the RTSP server supports it. For a complete list of RTSP headers, see RFC 2326.

- **header-value** *expression*—Specifies the header value expression string to compare against the value in the specified field in the RTSP header. Enter a text string with a maximum of 255 alphanumeric characters. The ACE supports the use of regular expressions for header matching. Expressions are stored in a header map in the form *header-name: expression*. Header expressions allow spaces if the entire string that contains spaces is quoted. If you use a **match-all** class map, all headers in the header map must be matched. See Table 3-3 for a list of the supported characters that you can use in regular expressions.

For example, to specify that the Layer 7 class map load balance on an RTSP header named Session, enter:

```
host1/Admin(config)# class-map type rtsp loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-rtsp-lb)# 100 match rtsp header Session
header-value abc123
```

To use regular expressions in a class map to emulate a wildcard search to match the header value expression string, enter:

```
host1/Admin(config)# class-map type rtsp loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-rtsp-lb)# 10 match rtsp header Require
header-value feature1
host1/Admin(config-cmap-rtsp-lb)# 20 match rtsp header Require
header-value feature2
```

To specify that the Layer 7 class map load balance on an RTSP header named Via, enter:

```
host1/Admin(config)# class-map type rtsp loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-rtsp-lb)# 30 match rtsp header Via
header-value 192.*
```

To remove all RTSP header match criteria from the L7SLBCLASS class map, enter:

```
host1/Admin(config-cmap-rtsp-lb)# no 10
host1/Admin(config-cmap-rtsp-lb)# no 20
host1/Admin(config-cmap-rtsp-lb)# no 30
```

Defining a URL for RTSP Load Balancing

The ACE performs regular expression matching against the received packet data from a particular connection based on the RTSP URL string. You can configure a class map to make Layer 7 SLB decisions based on the URL name and optionally, the RTSP method, by using the **match rtsp url** command in class-map RTSP load balance configuration mode. The syntax of this command is as follows:

[*line_number*] **match rtsp url** *expression*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The *line_number* line numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.

- *expression*—URL, or portion of a URL, to match. Enter a URL string from 1 to 255 alphanumeric characters. The ACE performs matching on whatever URL string appears after the RTSP method, regardless of whether the URL includes the hostname. The ACE supports the use of regular expressions for matching URL strings. See Table 3-3 for a list of the supported characters that you can use in regular expressions.

  Note When matching URLs, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

To specify that the Layer 7 class map load balance on a specific URL, enter:

```
host1/Admin(config)# class-map type rtsp loadbalance L7SLBCLASS
host1/Admin(config-cmap-rtsp-lb)# 10 match rtsp url /whatsnew/latest.*
```

To use regular expressions to emulate a wildcard search to match on any .wav or .mpg file, enter:

```
host1/Admin(config)# class-map type rtsp loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-rtsp-lb)# 100 match rtsp url .*.wmv
host1/Admin(config-cmap-rtsp-lb)# 200 match rtsp url .*.mpg
```

To remove a URL match statement from the L7SLBCLASS class map, enter **no** and the line number. For example, to remove line 100, enter:

```
host1/Admin(config-cmap-rtsp-lb)# no 100
```

Note If you did not use line numbers to enter the original URL match statement, you can obtain the line number from your running configuration. To display the running configuration, enter **show running-config**.

Defining a Header for SIP Load Balancing

The ACE performs regular expression matching against the received packet data from a particular connection based on the SIP header expression. You can configure a maximum of nine SIP header field names per class (the ACE always parses Call-ID).

Note When the ACE receives a SIP session, the load-balance decision is based on the first request message. All subsequent request and response message exchanges (with the same Call-ID) are forwarded to the same server. As a result, when configuring header match criteria, ensure that the header is included in the first request message.

You can configure a class map to make Layer 7 SLB decisions based on the name and value of a SIP header by using the **match sip header** command in class-map SIP load balance configuration mode.

The syntax of this command is as follows:

[*line_number*] **match sip header** *name* **header-value** *expression*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The sequence numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- *name*—Name of the field in the SIP header. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). You can enter any header field name, including a standard SIP header field name or any user-defined header field name. For a list of standard SIP header field names, see Table 3-5.

Note SIP is similar to HTTP, so you can use any HTTP header defined in Table 3-4 if the SIP server supports it. For a complete list of SIP headers, see RFC 3261.

- **header-value** *expression*—Specifies the header value expression string to compare against the value in the specified field in the SIP header. Enter a text string with a maximum of 255 alphanumeric characters. The ACE supports the use of regular expressions for header matching. Expressions are stored in a header map in the form *header-name: expression*. Header expressions allow spaces if the entire string that contains the spaces is quoted. If you use a **match-all** class map, all headers in the header map must be matched. See Table 3-3 for a list of the supported characters that you can use in regular expressions.

Table 3-5 Standard SIP Header Fields

Field Name	Description
Call-ID	Unique identifier that groups a series of messages in a call.
Contact	SIP URI that can be used to contact the user agent.
From	Initiator of the SIP request, the source.
To	Desired recipient of the SIP request; the destination.
Via	Transport used for the transaction and where the response should be sent.

For example, to specify that the Layer 7 class map load balance on an SIP header named Session, enter:

```
host1/Admin(config)# class-map type sip loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-sip-lb)# 100 match sip header Session
header-value abc123
```

To use regular expressions in a class map to emulate a wildcard search to match the header value expression string, enter:

```
host1/Admin(config)# class-map type sip loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-sip-lb)# 10 match sip header To header-value
.*@cisco.com
host1/Admin(config-cmap-sip-lb)# 20 match sip header To header-value
.*@linksys.com
```

To specify that the Layer 7 class map load balance on an SIP header named Via, enter:

```
host1/Admin(config)# class-map type sip loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-sip-lb)# 30 match sip header Via header-value
192.*
```

To remove all SIP header match criteria from the L7SLBCLASS class map, enter:

```
host1/Admin(config-cmap-sip-lb)# no 10
host1/Admin(config-cmap-sip-lb)# no 20
host1/Admin(config-cmap-sip-lb)# no 30
```

Defining Source IP Address Match Criteria

You can configure the class map to make Layer 7 SLB decisions based on a client source IP address by using the **match source-address** command in class-map load balance configuration mode. If this command is the only match criteria in the class map, the ACE considers it to be a Layer 3 and Layer 4 class map.

The syntax of this command is as follows:

[*line_number*] **match source-address** *ip_address* [*netmask*]

The arguments and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line. The line numbers do not indicate any priority for the match statements. Enter a unique integer from 1 to 1024.
- *ip_address—*Source IP address of the client. Enter the IP address in dotted-decimal notation (for example, 192.168.11.2).
- *netmask*—(Optional) Subnet mask of the IP address. Enter the netmask in dotted-decimal notation (for example, 255.255.255.0). The default is 255.255.255.255.

For best results, do not configure multiple class maps with overlapping subnets in the **match source-address** statements. For example:

```
class-map type http loadbalance match-any LB_CLASS
  2 match source-address 192.168.40.0 255.255.255.0
  3 match source-address 192.168.41.0 255.255.255.0
class-map type http loadbalance match-all WIDE_SUBNET_CLASS
  2 match source-address 192.168.0.0 255.255.0.0

policy-map type loadbalance http first-match HTTP_POLICY
  class LB_CLASS
    drop
  class WIDE_SUBNET_CLASS
    sticky-serverfarm SF2
```

In this case, the ACE may not match the client source addresses properly. To ensure proper operation of the ACE, configure individual source IP addresses in one of the class maps as a workaround.

For example, to apply the workaround to the WIDE_SUBNET_CLASS class map, configure the following IP adresses:

```
class-map type http loadbalance match-any WIDE_SUBNET
  2 match source-address 192.168.40.0 255.255.255.0
  3 match source-address 192.168.41.0 255.255.255.0
  4 match source-address 192.168.0.0 255.255.0.0
```

For example, to specify that the class map match on source IP address 192.168.11.2 255.255.255.0, enter:

```
host1/Admin(config)# class-map http type loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 50 match source-address 192.168.11.2
255.255.255.0
```

To remove the source IP address match statement from the class map, enter:

```
host1/Admin(config-cmap-http-lb)# no 50
```

Nesting Layer 7 SLB Class Maps

The nesting of class maps allows you to achieve complex logical expressions for Layer 7 SLB. You can identify one Layer 7 SLB class map that is to be used as a matching criterion for another Layer 7 class map by using the **match class-map** command in class-map load balance configuration mode.

Note The ACE restricts the nesting of class maps to two levels to prevent you from including a nested class map under another class map.

The syntax of this command is as follows:

[*line_number*] **match class-map** *map_name*

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you can use for editing or deleting the individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line.
- *map_name*—Name of an existing Layer 7 load-balancing class map.

The **match class-map** command allows you to combine the use of the **match-any** and **match-all** keywords in the same class map. To combine **match-all** and **match-any** characteristics in a class map, create a class map that uses one **match** command (either **match any** or **match all**), and then use this class map as a match statement in a second class map that uses a different match type.

For example, assume that commands A, B, C, and D represent separate match criteria, and you want Layer 7 traffic that matches A, B, or C and D (A or B or [C and D]) to satisfy the class map. Without the use of nested class maps, traffic would either have to match all four match criteria (A and B and C and D) or match any of the match criteria (A or B or C or D) to satisfy the class map. By creating a single class map that uses the **match-all** keyword for match criteria C and D (criteria E), you can then create a new **match-any** class map that uses match criteria A, B, and E. The new traffic class contains your desired classification sequence: A or B or E, which is equivalent to A or B or [C and D].

For example, to combine the characteristics of two class maps, one with **match-any** and one with **match-all** characteristics, into a single class map by using the **match class-map** command, enter:

```
host1/Admin(config)# class-map type http loadbalance match-any CLASS3
host1/Admin(config-cmap-http-lb)# 100 match http url .*.gif
host1/Admin(config-cmap-http-lb)# 200 match http url .*.html
host1/Admin(config-cmap-http-lb)# exit

host1/Admin(config)# class-map type http loadbalance match-all CLASS4
host1/Admin(config-cmap-http-lb)# 10 match class-map CLASS3
host1/Admin(config-cmap-http-lb)# 20 match source-address 192.168.11.2
host1/Admin(config-cmap-http-lb)# exit
```

To remove the nested class map from the Layer 7 class map, enter:

```
host1/Admin(config-cmap-http-lb)# no 10
```

Configuring a Layer 7 Policy Map for SLB

To use a Layer 7 SLB policy map, first create the policy map and define match statements and policy actions. Because Layer 7 policy maps are child policies, you must then 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 on an interface; you can activate only a Layer 3 and Layer 4 policy map on an interface or globally on all interfaces in a context.

For background information about the role of policy maps in the ACE, see the *Cisco Application Control Engine Module Administration Guide*.

Note The ACE treats as a Layer 3 and Layer 4 policy any policy map that has only source IP configured as the match criteria in the class map or inline match (except SIP LB) or the default class configured as the class map, and there are no configured Layer 7 policy actions.

You can create a Layer 7 SLB policy map and enter policy-map configuration mode by using the **policy-map type** command in configuration mode. The syntax of this command is as follows:

policy-map type loadbalance [**generic** | **http** | **radius** | **rdp** | **rtsp** | **sip**]
first-match *map_name*

The keywords and arguments are as follows:

- **loadbalance**—Specifies a policy map that defines Layer 7 SLB decisions.
- **generic**—(Optional) Specifies a generic protocol policy map for load balancing. Use this keyword to provide support for protocols that the ACE does not explicitly support. If you do not configure Layer 4 payload match criteria (see the "Defining Layer 4 Payload Match Criteria for Generic Data Parsing" section) or the UDP fast-age feature (see the "Enabling Per-Packet Load Balancing for UDP Traffic" section), the ACE treats the generic policy as a Layer 3 and Layer 4 policy.

  Note The **persistence-rebalance** command is not compatible with generic protocol parsing.

  When configuring the generic protocol policy map, you can also enable per-packet load balancing on UDP traffic, also known as the UDP fast-age feature. For more information on this feature, see the "Enabling Per-Packet Load Balancing for UDP Traffic" section.
- **http**—(Optional) Specifies the Hypertext Transfer Protocol (HTTP) for load balancing. This is the default.
- **radius**—(Optional) Specifies the Remote Authentication Dial-In User Service (RADIUS) for load balancing.
- **rdp**—(Optional) Specifies the Microsoft Remote Desktop Protocol (RDP) for load balancing.
- **rtsp**—(Optional) Specifies the Real-Time Streaming Protocol (RTSP) for load balancing.
- **sip**—(Optional) Specifies the Session Initiation Protocol (SIP) for load balancing.
- **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 a Layer 7 policy map for SIP load balancing, enter:

```
host1/Admin(config)# policy-map type loadbalance sip first-match
SIP_L7_POLICY
host1/Admin(config-pmap-lb-sip)#
```

To remove a policy map from the ACE, enter:

```
host1/Admin(config)# no policy-map type loadbalance sip first-match
SIP_L7_POLICY
```

This section contains the following topics that describe how to use sequence numbers, define inline match statements, and define policy-map actions:

- Adding a Layer 7 Policy Map Description
- Defining Inline Match Statements in a Layer 7 Policy Map
- Associating a Layer 7 Class Map with a Layer 7 Policy Map
- Specifying Layer 7 SLB Policy Actions
- Associating a Layer 7 Policy Map with a Layer 3 and Layer 4 Policy Map

Adding a Layer 7 Policy Map Description

You can use the **description** command to provide a brief summary about the Layer 7 policy map.

You must access the policy map configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

Use the *text* argument to enter an unquoted text string with a maximum of 240 alphanumeric characters.

For example, to add a description that the policy map is to insert HTTP headers, enter:

```
host1/Admin(config-pmap-lb)# description insert HTTP headers
```

To remove the description from the policy map, enter:

```
host1/Admin(config-pmap-lb)# no description
```

Defining Inline Match Statements in a Layer 7 Policy Map

Layer 7 SLB policy maps allow you to enter a single inline SLB match criteria in the policy map without specifying a traffic class. The inline Layer 7 SLB policy map **match** commands function similarly to the Layer 7 SLB class map **match** commands. However, when you use an inline **match** command, you can specify an action for only a single match statement in the Layer 7 policy.

Note To specify actions for multiple match statements, use a class map as described in the "Configuring a Layer 7 Class Map for Generic TCP and UDP Data Parsing" and "Configuring a Layer 7 Class Map for SLB" sections.

The syntax for an inline **match** command is as follows:

match *name1 match_statement* [**insert-before** *name2*]

The arguments and options are as follows:

- *name1*—Name assigned to the inline **match** command. Enter an unquoted text string with no spaces. The length of the inline match statement name plus the length of the policy map name with which it is associated cannot exceed a total maximum of 64 alphanumeric characters. For example, if the policy map name is L7_POLICY (nine characters), an inline match statement name under this policy cannot exceed 55 alphanumeric characters (64 - 9 = 55).
- *match_statement*—Individual Layer 7 SLB match criteria.
- **insert-before** *name2*—(Optional) Places the current match statement ahead of an existing class map or other 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.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

For information about the inline match statements that you can configure in a Layer 7 SLB policy map, see the "Configuring a Layer 7 Class Map for Generic TCP and UDP Data Parsing" and the "Configuring a Layer 7 Class Map for SLB" sections.

Note The *line_number* argument described in the above-referenced sections is only for use with match statements in class maps. Otherwise, the descriptions of match statements in Layer 7 class maps and inline match statements in Layer 7 policy maps are the same.

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

You can associate an existing Layer 7 class map with a Layer 7 policy map by using the **class** command. The syntax of this command is as follows:

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

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 to the traffic policy. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

- **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.
- **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 none of the specified classifications match the traffic, then the ACE performs the action specified under 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.

For example, to use the **insert-before** option to define the position of a class map in the policy map, enter:

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

To remove a class map from a Layer 7 policy map, enter:

```
host1/Admin(config-pmap-lb)# no class L7SLBCLASS
```

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

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

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 must specify the actions that the ACE should take when network traffic matches a class map or inline **match** command. You can specify the Layer 7 SLB policy actions by using the commands described in the following topics:

- Associating an Action List with a Layer 7 Policy Map
- Discarding Requests
- Forwarding Requests Without Load Balancing
- Configuring HTTP Header Insertion
- Enabling Load Balancing to a Server Farm

- Configuring a Sticky Server Farm
- Specifying the IP Differentiated Services Code Point of Packets
- Specifying an SSL Proxy Service

Associating an Action List with a Layer 7 Policy Map

You use action lists to group several ACE actions (for example, HTTP header insert, rewrite, or delete) together in a named list under a Layer 7 policy map. For details about configuring an action list, see the "Configuring HTTP Header Insertion, Deletion, and Rewrite" section.

You can associate an action list with a Layer 7 policy map by using the **action** command in policy map load-balancing class configuration mode. 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 policy map, enter:

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

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

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

For example, to associate an action list for HTTP header rewrite, enter:

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

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

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

Discarding Requests

You can instruct the ACE to discard packets that match a particular policy map by using the **drop** command in policy map load-balancing class configuration mode. The syntax of this command is as follows:

drop

For example, enter:

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

To reset the behavior of the ACE to the default of accepting packets that match a policy map, enter:

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

Forwarding Requests Without Load Balancing

You can instruct the ACE to forward requests that match a particular policy map without performing load balancing on the request by using the **forward** command in policy map load-balancing class configuration mode. The syntax of this command is as follows:

forward

For example, enter:

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

To reset the ACE to the default of load balancing packets that match a policy map, enter:

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

Configuring HTTP Header Insertion

When the ACE uses Network Address Translation (NAT) to translate the source IP address of a client to a VIP, servers need a way to identify that client for the TCP and IP return traffic. To identify a client whose source IP address has been translated using NAT, you can instruct the ACE to insert a generic header and string value of your choice in the client HTTP request. (For information about NAT, see the *Cisco Application Control Engine Module Security Configuration Guide*.

Note With either TCP server reuse or persistence rebalance enabled, the ACE inserts a header in every client request. For information about TCP server reuse, see the "Configuring TCP Server Reuse" section. For information about persistence rebalance, see the "Configuring HTTP Persistence Rebalance" section.

You can insert a generic header and value in an HTTP request by using the **insert-http** command in policy map load-balancing class configuration mode. You can enter multiple **insert-http** commands for each class. The syntax of this command is as follows:

insert-http *name* **header-value** *expression*

The keywords and arguments are as follows:

- *name*—Name of the HTTP header to insert in the client HTTP request. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. You can specify any custom header name that you want, subject to the maximum character length. You can also enter any of the predefined header names in Table 3-4, regardless of whether that header name already exists in the client request header. The ACE does not overwrite any existing header information in the client request.
- **header-value** *expression*—Specifies the header-value expression string to insert in the HTTP header. Enter a text string with a maximum of 512 alphanumeric characters. If you configure more than 512 bytes of data to be inserted into the HTTP header, the ACE does not insert any data in the header.

  You can also specify the following special **header-value** expressions using the following dynamic replacement strings:

  - **%is**—Inserts the source IP address in the HTTP header
  - **%id**—Inserts the destination IP address in the HTTP header
  - **%ps**—Inserts the source port in the HTTP header
  - **%pd**—Inserts the destination port in the HTTP header

Note For Microsoft Outlook Web Access (OWA), specify the field name as HTTP_FRONT_END_HTTPS with a value of ON.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

For example, in an SSL configuration, you could insert a generic field called ClientCert, and the header value could be the client certificate or a portion thereof.

For example, to insert the header name Host with a header value of www.cisco.com in an HTTP client request header, enter:

```
host1/Admin(config)# policy-map type loadbalance first-match
L7SLBPOLICY
host1/Admin(config-pmap-lb)# class L7SLBCLASS
host1/Admin(config-pmap-lb-c)# insert-http Host header-value
www.cisco.com
```

The header name and value will appear in the HTTP header as follows:

```
Host: www.cisco.com
```

To remove the HTTP header name and value from the policy map, enter:

```
host1/Admin(config-pmap-lb-c)# no insert-http Host header-value
www.cisco.com
```

Enabling Load Balancing to a Server Farm

You can load balance a client request for content to a server farm by using the **serverfarm** command in policy map load-balancing class configuration mode. Server farms are groups of networked real servers that contain the same content and that typically reside in the same physical location.

The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Unique identifier of the server farm. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **backup** *name2*—(Optional) Designates an existing host (with valid content) or a redirect (sorry) server farm as a backup server farm in case all the real servers in the primary server farm become unavailable. You can configure one backup server farm for each existing primary server farm. When at least one server in the primary server farm becomes available again, the ACE sends all new connections back to the primary server farm. The ACE allows existing connections to the backup server farm to complete. You can fine-tune the conditions under which the primary server farm fails over and returns to service by configuring a partial server farm failover. For details about partial server farm failover, see the "Configuring a Partial Server Farm Failover" section in Chapter 2, Configuring Real Servers and Server Farms. Enter the name of an existing server farm that you want to specify as a backup server farm as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

  Note If all servers in the server farm fail and you did not configure a backup server farm, the ACE sends a reset (RST) to a client in response to a connection request.

- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

The following example specifies the **serverfarm** command as an action in a Layer 7 load-balancing policy map:

```
host1/Admin(config)# policy-map type loadbalance first-match
L7SLBPOLICY
host1/Admin(config-pmap-lb)# class L7SLBCLASS
host1/Admin(config-pmap-lb-c)# serverfarm SFARM1 backup SFARM2
```

To remove the server-farm action from the Layer 7 load-balancing policy map, enter:

```
host1/Admin(config-pmap-lb-c)# no serverfarm FARM2
```

Configuring a Sorry Server Farm

When the primary server farm is unavailable, you can instruct the ACE to send client requests to a sorry server farm. A sorry server is a redirect server in a backup server farm with content stating that the web page, resource, or service that a client requested is temporarily unavailable. When at least one server in the primary server farm returns to service, the ACE directs clients back to the primary server farm. You can fine-tune the conditions under which the primary server farm fails over and returns to service by configuring a partial server farm failover. For details about partial server farm failover, see the "Configuring a Partial Server Farm Failover" section in Chapter 2, Configuring Real Servers and Server Farms.

To configure a sorry server, use the **serverfarm** command in policy map class configuration mode as described in the "Enabling Load Balancing to a Server Farm" section and configure an existing redirect server farm as the backup server farm. If you specifically want client connections to return to the same server in the primary server farm that they were connected to before the primary went down based on a client source IP addresses, configure the **predictor hash address source command** as the load-balancing method for the primary server farm. If the primary server farm is down and you want all requests from a particular subnet to be redirected to a particular site and requests from a different subnet to be redirected to a different site, use the **predictor hash address source** command as the load-balancing method for the sorry server farm. For equal load balancing across the sites in either server farm, use the **roundrobin** predictor method. Otherwise, you can configure any supported predictor method that works for your application on either server farm. For information about configuring the server farm predictor, see Chapter 2, Configuring Real Servers and Server Farms.

For example, to configure a primary server farm and a sorry server farm, enter the following commands:

```
host1/Admin(config)# rserver SERVER1
host1/Admin(config-rserver-host)# ip address 192.168.12.4
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
host1/Admin(config)# rserver SERVER2
host1/Admin(config-rserver-host)# ip address 192.168.12.5
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
host1/Admin(config)# rserver redirect SERVER3
host1/Admin(config-rserver-redir)# webhost-redirection www.cisco.com
301
host1/Admin(config-rserver-redir)# inservice
host1/Admin(config-rserver-redir)# exit
host1/Admin(config)# rserver redirect SERVER4
host1/Admin(config-rserver-redir)# webhost-redirection www.cisco.com
301
host1/Admin(config-rserver-redir)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# predictor roundrobin
host1/Admin(config-sfarm-host)# rserver SERVER1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver SERVER2
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# exit

host1/Admin(config)# serverfarm redirect SFARM2
host1/Admin(config-sfarm-redirect)# predictor roundrobin
host1/Admin(config-sfarm-redirect)# rserver SERVER3
host1/Admin(config-sfarm-redirect-rs)# inservice
host1/Admin(config-sfarm-redirect-rs)# exit
host1/Admin(config-sfarm-redirect)# rserver SERVER4
host1/Admin(config-sfarm-redirect-rs)# inservice
host1/Admin(config-sfarm-redirect-rs)# exit
host1/Admin(config-sfarm-redirect)# exit

host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# 100 match http header Host
header-value .*cisco.com
host1/Admin(config-cmap-http-lb)# exit
```

```
host1/Admin(config)# policy-map type loadbalance first-match
L7SLBPOLICY
host1/Admin(config-pmap-lb)# class L7SLBCLASS
host1/Admin(config-pmap-lb-c)# serverfarm SFARM1 backup SFARM2
```

Configuring a Sticky Server Farm

You can specify that requests matching a Layer 7 policy map be load balanced to a sticky server farm by using the **sticky-serverfarm** command in policy map load-balancing class configuration mode. The syntax of this command is as follows:

sticky-serverfarm *name*

The *name* argument is the identifier of an existing sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. For information about sticky groups, see Chapter 5, Configuring Stickiness.

For example, enter:

```
host1/Admin(config-pmap-lb-c)# sticky-serverfarm STICKY_GROUP1
```

To remove the sticky server farm from the policy map, enter:

```
host1/Admin(config-pmap-lb-c)# no sticky-serverfarm STICKY_GROUP1
```

Specifying the IP Differentiated Services Code Point of Packets

You can specify the IP differentiated services code point (DSCP) of packets in a policy map by using the **set ip tos** command in policy map load-balancing class configuration mode. This command marks a packet by setting the IP DSCP bit in the Type of Service (ToS) byte. Once the IP DSCP bit is set, other Quality of Service (QoS) services can operate on the bit settings.

The syntax of this command is as follows:

set ip tos *value*

The *value* argument is the IP DSCP value. Enter an integer from 0 to 255. The default is to not modify the ToS field.

The following example specifies the **set ip tos** command as a QoS action in the Layer 7 load-balancing policy map. All packets that satisfy the match criteria of L7SLBCLASS are marked with the IP DSCP value of 8. How packets marked with the IP DSCP value of 8 are treated is determined by the network configuration.

```
host1/Admin(config)# policy-map type loadbalance first-match
L7SLBPOLICY
host1/Admin(config-pmap)# class L7SLBCLASS
host1/Admin(config-pmap-lb-c)# set ip tos 8
```

To reset the ACE to the default of not modifying the ToS byte value, enter:

```
host1/Admin(config-pmap-lb-c)# no set ip tos 8
```

Specifying an SSL Proxy Service

The ACE uses an SSL proxy service in a Layer 7 policy map to load balance outbound SSL initiation requests to SSL servers. The ACE acts as an SSL client sending an encrypted request to an SSL server. For more information about SSL initiation, see the *Cisco Application Control Engine Module SSL Configuration Guide*.

To specify an SSL proxy service in a policy map, use 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 service. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config-pmap-lb-c)# ssl-proxy client PROXY_SERVICE1
```

To remove the SSL proxy service from the policy map, enter:

```
host1/Admin(config-pmap-lb-c)# no ssl-proxy client PROXY_SERVICE1
```

Associating a Layer 7 Policy Map with a Layer 3 and Layer 4 Policy Map

You can associate a Layer 7 SLB policy with a Layer 3 and Layer 4 SLB policy by using the **service-policy type loadbalance** command in policy-map class configuration mode. For details, see the "Associating a Layer 7 SLB Policy Map with a Layer 3 and Layer 4 SLB Policy Map" section.

Configuring a Generic Protocol Parameter Map

You can use a parameter map to combine related actions for generic protocol parsing. You reference this parameter map in the policy map by using the **appl-parameter generic advanced-options** command. See the "Associating a Generic, HTTP, or RTSP Parameter Map with a Layer 3 and Layer 4 Policy Map" section.

You can configure generic protocol actions for SLB connections by using the **parameter-map type generic** command in configuration mode. The syntax of this command is as follows:

parameter-map type generic *name*

The *name* argument is the identifier assigned to the parameter map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# parameter-map type generic GEN_PARAMETER_MAP
```

To remove a generic parameter map from the configuration, enter:

```
host1/Admin(config)# no parameter-map type generic GEN_PARAMETER_MAP
```

The following topics describe how to use the commands to define the generic protocol parameter map:

- Disabling Case-Sensitivity Matching for Generic Protocols
- Setting the Maximum Number of Bytes to Parse for Generic Protocols

Disabling Case-Sensitivity Matching for Generic Protocols

By default, the ACE CLI is case sensitive. To disable case-sensitivity matching for generic protocols only, use the **case-insensitive** command in generic parameter-map configuration mode. With case-insensitive matching enabled, uppercase and lowercase letters are considered the same.

The syntax of this command is as follows:

case-insensitive

For example, to disable case sensitivity, enter:

```
host1/Admin(config-parammap-generi)# case-insensitive
```

To reenable case-sensitive matching after it has been disabled, enter:

```
host1/Admin(config-parammap-generi)# no case-insensitive
```

Setting the Maximum Number of Bytes to Parse for Generic Protocols

You can set the maximum number of bytes to parse for generic protocols by using the **set max-parse-length** command in generic parameter-map configuration mode. The syntax of this command is as follows:

set max-parse-length *bytes*

The *bytes* argument is the maximum number of bytes to parse. Enter an integer from 1 to 65535. The default is 2048 bytes.

For example, to set the maximum parse length to 8192, enter:

```
host1/Admin(config-parammap-generi)# set max-parse-length 8192
```

To reset the maximum parse length to the default of 2048 bytes, enter:

```
host1/Admin(config-parammap-generi)# no set max-parse-length
```

Configuring an HTTP Parameter Map

You can use a parameter map to combine related HTTP actions for a Layer 3 and Layer 4 policy map. You reference this parameter map in the policy map by using the **appl-parameter http advanced-options** command. See the "Associating a Generic, HTTP, or RTSP Parameter Map with a Layer 3 and Layer 4 Policy Map" section.

You can configure advanced HTTP behavior for SLB connections by using the **parameter-map type http** command in configuration mode. The syntax of this command is as follows:

parameter-map type http *name*

The *name* argument is the identifier assigned to the parameter map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# parameter-map type http HTTP_PARAMETER_MAP
```

To remove an HTTP parameter map from the configuration, enter:

```
host1/Admin(config)# no parameter-map type http HTTP_PARAMETER_MAP
```

The following topics describe how to use the commands to define the advanced HTTP parameter map:

- Disabling Case-Sensitivity Matching for HTTP
- Configuring the ACE to Modify Headers on Every HTTP Request or Response
- Defining URL Delimiters
- Setting the Maximum Number of Bytes to Parse for Content
- Setting the Maximum Number of Bytes to Parse for Cookies, HTTP Headers, and URLs
- Configuring the ACE Behavior when a URL or Cookie Exceeds the Maximum Parse Length
- Configuring HTTP Persistence Rebalance
- Configuring TCP Server Reuse

Disabling Case-Sensitivity Matching for HTTP

By default, the ACE CLI is case sensitive. To disable case-sensitivity matching for HTTP only, use the **case-insensitive** command in HTTP parameter-map configuration mode. With case-insensitive matching enabled, uppercase and lowercase letters are considered the same. When case sensitivity is disabled, it applies to the following:

- HTTP header names and values
- HTTP cookie names and values
- URL strings
- HTTP deep inspection (for details, see the *Cisco Application Control Engine Module Security Configuration Guide*)

The syntax of this command is as follows:

case-insensitive

For example, to disable case sensitivity, enter:

```
host1/Admin(config-parammap-http)# case-insensitive
```

To reenable case-sensitive matching after it has been disabled, enter:

```
host1/Admin(config-parammap-http)# no case-insensitive
```

Configuring the ACE to Modify Headers on Every HTTP Request or Response

By default, when the **persistence-rebalance** command is disabled and you configure the ACE to modify HTTP headers (insert, delete, or rewrite), the ACE performs the operation only on the first HTTP request or response. The **persistence-rebalance** command causes the ACE to perform header modifications on each request or response, but it also instructs the ACE to load balance each new request to a potentially new real server. For more information about the **persistence-rebalance** command, see the "Configuring HTTP Persistence Rebalance" section. For information about header insertion, deletion, and rewrite, see the "Configuring HTTP Header Insertion, Deletion, and Rewrite" section.

To instruct the ACE to modify headers (insert, delete, or rewrite) on every HTTP request or response without the additional effect of performing load balancing on each new HTTP request caused by the **persistence-rebalance** command, use the **header modify per-request** command in parameter map HTTP configuration mode. This command has an effect only when **persistence-rebalance** is disabled. The syntax of this command is:

header modify per-request

The **header modify per-request** command also causes the ACE to perform URL location header rewrite on every HTTP response if the **ssl url rewrite location** command is enabled. For more information about SSL URL rewrite, see the *Cisco Application Control Engine Module SSL Configuration Guide*.

For example, to instruct the ACE to perform header modification on every HTTP request or response, enter the following command:

```
host1/Admin(config-parammap-http)# header modify per-request
```

To return the ACE behavior to the default of modifying headers only on the first HTTP request or response, enter the following command:

```
host1/Admin(config-parammap-http)# no header modify per-request
```

Defining URL Delimiters

You can define a list of ASCII-character delimiters that you can use to separate the cookies in a URL string by using the **set secondary-cookie-delimiters** command in parameter map HTTP configuration mode. The syntax of this command is as follows:

set secondary-cookie-delimiters *text*

The *text* argument identifies the list of delimiters. Enter an unquoted text string with no spaces and a maximum of four characters. The order of the delimiters in the list does not matter. The default list of delimiters is /&#+.

Cookies and their delimiters appear in GET request lines. In the following example of a GET request line, the ampersand (&) that appears between name-value pairs is the secondary cookie delimiter. The question mark (?) begins the URL query and is not configurable.

```
GET /default.cgi?user=me&hello=world&id=2 HTTP/1.1
```

For example, to specify the secondary cookie delimiters as !@#$, enter:

```
host1/Admin(config-parammap-http)# set secondary-cookie-delimiters
!@#$
```

To reset the delimiter list to the default of /&#+, enter:

```
host1/Admin(config-parammap-http)# no set secondary-cookie-delimiters
!@#$
```

Defining the Secondary Cookie Start

You can define the ASCII-character string at the start of a secondary cookie in a URL or ignore any start string of a secondary cookie in the URL and consider the secondary cookie part of the URL by using the **set secondary-cookie-start** command in parameter map HTTP configuration mode. The syntax of this command is as follows:

set secondary-cookie-start {**none** | *text*}

The keyword and argument are as follows:

- **none**—The secondary cookie start is not configured or the ACE ignores any start string of a secondary cookie in the URL and considers the secondary cookie as part of the URL.

  When you configure the **none** keyword to consider the entire URL query string as part of a URL, the commands that rely on the URL query, such as the **match cookie secondary** and **predictor hash cookie secondary** commands, do not work. Do not configure these commands under the same real server.

- *text*—The start string of the secondary cookie. Enter a maximum of two characters. The default start character is ?.

For example, to define the secondary cookie start string, enter:

```
host1/Admin(config-parammap-http)# set secondary-cookie-start ?!
```

To reset the secondary cookie start string to the default setting of ?, enter:

```
host1/Admin(config-parammap-http)# no set secondary-cookie-start
```

Setting the Maximum Number of Bytes to Parse for Content

By default, the maximum number of bytes that the ACE parses to check for content is 4096. If a content string exceeds the default value, the ACE drops the packet and sends a RST (reset) to the client browser. You can increase the number of bytes that the ACE parses using the **set content-maxparse-length** command in HTTP parameter map configuration mode. The syntax of this command is as follows:

set content-maxparse-length *bytes*

The *bytes* argument is the maximum number of bytes to parse for the total length of a content string. Enter an integer from 1 to 65535. The default is 4096 bytes.

Note If you set the *bytes* argument to a value that is greater than 32 KB, you must configure the **set tcp buffer-share** command in a connection parameter map to a value that is greater than the *bytes* value. If you do not, even if you configure the **length-exceed continue** command, the ACE may not completely parse a content string packet that is greater than 32 KB. The reason is that the default value of the **set tcp buffer-share** command buffer size (32 KB) will not accommodate the larger content string size.

For example, to set the content maximum parse length to 8192, enter:

```
host1/Admin(config-parammap-http)# set content-maxparse-length 8192
```

To reset the HTTP content maximum parse length to the default of 4096 bytes, enter:

```
host1/Admin(config-parammap-http)# no set content-maxparse-length
```

Setting the Maximum Number of Bytes to Parse for Cookies, HTTP Headers, and URLs

By default, the maximum number of bytes that the ACE parses to check for a cookie, HTTP header, or URL is 4096. If a cookie, HTTP header, or URL exceeds the default value, the ACE drops the packet and sends a RST (reset) to the client browser. You can increase the number of bytes that the ACE parses using the **set header-maxparse-length** command in HTTP parameter-map configuration mode. The syntax of this command is as follows:

set header-maxparse-length *bytes*

The *bytes* argument is the maximum number of bytes to parse for the total length of all cookies, HTTP headers, and URLs. Enter an integer from 1 to 65535. The default is 4096 bytes.

Note If you set the *bytes* argument to a value that is greater than 32 KB, you must configure the **set tcp buffer-share** command in a connection parameter map to a value that is greater than the *bytes* value. If you do not, even if you configure the **length-exceed continue** command, the ACE may not completely parse a header packet that is greater than 32 KB. The reason is that the default value of the **set tcp buffer-share** buffer size (32 KB) will not accommodate the larger header size.

For example, to set the HTTP header maximum parse length to 8192, enter:

```
host1/Admin(config-parammap-http)# set header-maxparse-length 8192
```

To reset the HTTP header maximum parse length to the default of 4096 bytes, enter:

```
host1/Admin(config-parammap-http)# no set header-maxparse-length
```

Configuring the ACE Behavior when a URL or Cookie Exceeds the Maximum Parse Length

You can configure how the ACE handles cookies, HTTP headers, and URLs that exceed the maximum parse length by using the **length-exceed** command in HTTP parameter-map configuration mode. The syntax of this command is as follows:

length-exceed {**continue** | **drop**}

The keywords are as follows:

- **continue**—Specifies how to continue load balancing. When you specify this keyword, the **persistence-rebalance** command is disabled if the total length of all cookies, HTTP headers, and URLs exceeds the maximum parse length value. For details on setting the maximum parse length, see the "Setting the Maximum Number of Bytes to Parse for Cookies, HTTP Headers, and URLs" section.
- **drop**—(Default) Specifies how to stop load balancing and discard the packet.

For example, enter:

```
host1/Admin(config-parammap-http)# length-exceed continue
```

To reset the ACE to the default of stopping load balancing and discarding a packet when its URL or cookie exceeds the maximum parse length, enter:

```
host1/Admin(config-parammap-http)# no length-exceed continue
```

Configuring HTTP Persistence Rebalance

When persistence rebalance is disabled (the default for the module), the ACE matches an HTTP request to a Layer 7 class map in a Layer 7 load-balancing policy map and load balances the request to one of the servers in the serverfarm associated with that class map. The ACE sends all subsequent requests on the same TCP connection to the same server regardless of whether or not they match the same Layer 7 class map.

You can enable the persistence rebalance feature by using the **persistence-rebalance** command in HTTP parameter-map configuration mode. (Be sure to apply the HTTP parameter map to a Layer 4 multi-match policy map.) The syntax of this command is as follows:

persistence rebalance

With persistence rebalance enabled, when the first HTTP request arrives, the ACE matches the request to a Layer 7 class map in a Layer 7 policy map and load balances the request to one of the servers in the serverfarm associated with that class map. The ACE matches all subsequent requests on the same TCP connection to a Layer 7 class map. If the subsequent request matches the same Layer 7 class map as the previous request, then the ACE sends the request to the same server as the previous request. This behavior produces less overhead and better performance.

If the request matches a different Layer 7 class map, then the ACE load balances the request to one of the servers in the server farm associated with the newly matched Layer 7 class map according to the serverfarm predictor.

When persistence rebalance is enabled, header insertion and cookie insertion, if enabled, occur for every request instead of only the first request. For information about header insertion, see the "Configuring HTTP Header Insertion" section in this chapter. For information about cookie insertion, see the "Enabling Cookie Insertion" section in Chapter 5, Configuring Stickiness.

Note If a real server is enabled with the NTLM Microsoft authentication protocol, we recommend that you leave persistence rebalance disabled. NTLM is a security measure that is used to perform authentication with Microsoft remote access protocols. When a real server is enabled with NTLM, every connection to the real server must be authenticated; typically, each client user will see a pop-up window prompting for a username and password. Once the connection is authenticated, all subsequent requests on the same connection will not be challenged. However, when the server load balancing function is enabled and configured with persistence rebalance, a subsequent request may point to a different real server causing a new authentication handshake.

The following example specifies the **parameter-map type http** command to configure URL cookie delimiter strings, set the maximum number of bytes to parse for URLs and cookies, and enable HTTP persistence:

```
host1/Admin(config)# parameter-map type http http_parameter_map
host1/Admin(config-parammap-http)# secondary-cookie-delimiters !@#$
host1/Admin(config-parammap-http)# header-maxparse-length 8192
length-exceed continue
host1/Admin(config-parammap-http)# persistence-rebalance
```

To reset persistence to the default setting of disabled, enter:

```
host1/Admin(config-parammap-http)# no persistence-rebalance
```

Configuring Persistence with Load Balancing on Each HTTP Request

When the persistence-rebalance feature is enabled on the ACE, it does not load balance successive GET requests on the same TCP connection unless it matches a load-balancing class map that is different from the load-balancing class map matched by the previous request (see the "Configuring HTTP Persistence Rebalance" section). To configure the ACE to load balance each subsequent GET request on the same TCP connection independently, use the **persistence-rebalance strict** command in HTTP parameter-map configuration mode.

The syntax of this command is as follows:

persistence-rebalance strict

This command allows the ACE to load balance each HTTP request to a potentially different Layer 7 class or real server.

For example, to enable the strict persistence rebalance feature, enter:

```
host1/Admin(config)# parameter-map type http http_parameter_map
host1/Admin(config-parammap-http)# persistence-rebalance strict
```

To reset persistence to the default setting of disabled, enter:

```
host1/Admin(config-parammap-http)# no persistence-rebalance
```

To change to persistence rebalance behavior that does not load balance successive requests to the same connection, use the **persistence-rebalance** command.

Configuring TCP Server Reuse

TCP server reuse allows the ACE to reduce the number of open connections on a server by allowing connections to persist and be reused by multiple client connections. The ACE maintains a pool of TCP connections based on TCP options. New client connections can reuse those connections in the pool if the new client connections and prior server connections share the same TCP options. For information about configuring how the ACE handles TCP options, see the *Cisco Application Control Engine Module Security Configuration Guide*.

To ensure proper operation of this feature, follow these TCP server reuse recommendations and restrictions:

- Ensure that the ACE maximum segment size (MSS) is the same as the server MSS.
- Configure port address translation (PAT) on the interface that is connected to the real server. PAT prevents collisions when a client stops using a server connection, and then that connection is reused by another client. Without PAT, if the original client tries to reuse the original server connection, it is no longer available. For details about configuring PAT, see the *Cisco Application Control Engine Module Security Configuration Guide*.
- Configure the ACE with the same TCP options that exist on the TCP server.
- Ensure that each server farm is homogeneous (all real servers within a server farm have identical configurations).

Note Always configure PAT with TCP reuse. If you configure TCP reuse and NAT only, unexpected results may occur.

Another effect of TCP server reuse is that header insertion and cookie insertion, if enabled, occur for every request instead of only the first request. For information about header insertion, see the "Configuring HTTP Header Insertion" section in this chapter. For information about cookie insertion, see the "Enabling Cookie Insertion" section in Chapter 5, Configuring Stickiness.

You can configure TCP server reuse by using the **server-conn reuse** command in HTTP parameter-map configuration mode. The syntax of this command is as follows:

```
server-conn reuse
```

For example, to enable TCP server reuse, enter:

```
host1/Admin(config-parammap-http)# server-conn reuse
```

To disable TCP server reuse, enter:

```
host1/Admin(config-parammap-http)# no server-conn reuse
```

Configuring an RTSP Parameter Map

You can use a parameter map to combine related RTSP actions for a Layer 3 and Layer 4 policy map. You reference this parameter map in the policy map using the **appl-parameter rtsp advanced-options** command. See the "Associating a Generic, HTTP, or RTSP Parameter Map with a Layer 3 and Layer 4 Policy Map" section.

You can configure advanced RTSP behavior for SLB connections by using the **parameter-map type rtsp** command in configuration mode. The syntax of this command is as follows:

parameter-map type rtsp *name*

The *name* argument is the identifier assigned to the parameter map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# parameter-map type rtsp RTSP_PARAMETER_MAP
```

To remove an RTSP parameter map from the configuration, enter:

```
host1/Admin(config)# no parameter-map type rtsp RTSP_PARAMETER_MAP
```

You can define the advanced RTSP parameter map by using the commands in the following topics:

- Disabling Case-Sensitivity Matching for RTSP
- Setting the Maximum Number of Bytes to Parse for RTSP Headers

Disabling Case-Sensitivity Matching for RTSP

By default, the ACE CLI is case sensitive. To disable case-sensitivity matching for RTSP, use the **case-insensitive** command in RTSP parameter-map configuration mode. With case-insensitive matching enabled, uppercase and lowercase letters are considered the same. When case sensitivity is disabled, it applies to the following:

- RTSP header names and values
- RTSP URL strings
- RTSP inspection (for details, see the *Cisco Application Control Engine Module Security Configuration Guide*)

The syntax is as follows:

case-insensitive

For example, to disable case sensitivity, enter:

```
host1/Admin(config-parammap-rtsp)# case-insensitive
```

To reenable case-sensitive matching after it has been disabled, enter:

```
host1/Admin(config-parammap-rtsp)# no case-insensitive
```

Setting the Maximum Number of Bytes to Parse for RTSP Headers

You can set the maximum number of bytes to parse for RTSP headers by using the **set header-maxparse-length** command in RTSP parameter map configuration mode. The syntax of this command is as follows:

set header-maxparse-length *bytes*

The *bytes* argument is the maximum number of bytes to parse for the total length of all RTSP headers. Enter an integer from 1 to 65535. The default is 2048 bytes.

For example, to set the RTSP header maximum parse length to 16384, enter:

```
host1/Admin(config-parammap-rtsp)# set header-maxparse-length 16384
```

To reset the RTSP header maximum parse length to the default of 2048 bytes, enter:

```
host1/Admin(config-parammap-rtsp)# no set header-maxparse-length
```

Configuring a Layer 3 and Layer 4 Class Map for SLB

A Layer 3 and Layer 4 class map contains match criteria to classify network traffic that can pass through the ACE. The ACE uses these Layer 3 and Layer 4 traffic classes to perform server load balancing (SLB). For a Layer 3 and Layer 4 traffic classification, the match criteria in a class map include the VIP address, protocol, and port of the ACE. You can configure multiple commands in a single class map to specify the match criteria in a group that you then associate with a traffic policy.

You can create a Layer 3 and Layer 4 class map to classify network traffic passing through the ACE and enter class-map configuration mode by using the **class-map** command in configuration mode. The syntax of this command is as follows:

class-map [**match-all** | **match-any**] *map_name*

The arguments and options are as follows:

- **match-all | match-any**—(Optional) Determines how the ACE evaluates Layer 3 and Layer 4 network traffic when multiple match criteria exist in a class map. The class map is considered a match if the **match** commands meet one of the following conditions:
  - **match-all**—(Default) Network traffic needs to satisfy all of the match criteria (implicit AND) to match the class map.
  - **match-any—**Network traffic needs to satisfy only one of the match criteria (implicit OR) to match the load-balancing class map.
- *map_name*—Name assigned to the class map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. The class name is used for both the class map and to configure policy for the class in the policy map.

For example, to create a class map named L4VIPCLASS that specifies the network traffic must satisfy all the match criteria (**match-all** is the default), enter:

```
host1/Admin(config)# class-map L4VIPCLASS
```

To remove the class map from the configuration, enter:

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

After you have created a Layer 3 and Layer 4 class map for SLB, use the commands in the following topics to configure a description and the VIP match criteria for the class map:

- Defining a Class Map Description
- Defining VIP Address Match Criteria

Defining a Class Map Description

You can provide a brief summary about the Layer 3 and Layer 4 class map by using the **description** command in class-map configuration mode. The syntax of this command is as follows:

description *text*

For the *text* argument, enter an unquoted text string with no spaces and a maximum of 240 alphanumeric characters.

For example, to specify a description where the class map filters network traffic to the server, enter:

```
host1/Admin(config)# class-map match-any L4VIPCLASS
host1/Admin(config-cmap)# description filter server traffic
```

To remove the description from the class map, enter:

```
host1/Admin(config-cmap)# no description
```

Defining VIP Address Match Criteria

You can define a 3-tuple flow of VIP address, protocol, and port as matching criteria for SLB by using the **match virtual-address** command in class-map configuration mode. You can configure multiple match criteria statements to define the VIPs for SLB.

Note Previous to release A2(2.0), the ACE allowed you to configure a class-map VIP address that overlaps with an ACE interface IP address. The ACE no longer ignores this configuration and displays the following warning:

```
Error: Entered VIP address is not the first address in the VIP
range
```

The syntax of this command is as follows:

[*line_number*] **match virtual-address** *vip_address* {[*mask*] | **any** | {**tcp** | **udp** {**any** | **eq** *port_number* | **range** *port1 port2*}} | *protocol_number*}

The keywords, arguments, and options are as follows:

- *line_number*—(Optional) Line numbers that you use for editing or deleting individual **match** commands. For example, you can enter **no** *line_number* to delete long **match** commands instead of entering the entire line.
- *vip_address*—Virtual IP (VIP) address of the virtual server in the ACE specified in dotted-decimal notation (192.168.1.2). VIPs are public addresses owned by the customer.
- *mask*—(Optional) Mask for the VIP address of the ACE to allow connections to an entire network specified in dotted decimal format (255.255.255.0).
- **any**—Specifies the wildcard value for the IP protocol value.
- **tcp** | **udp**—Specifies the protocol, TCP or UDP.
  - **any**—Specifies the wildcard value for the TCP or UDP port number. With **any** used in place of either the **eq** or **range** values, packets from any incoming port match.
  - **eq** *port_number*—Specifies that the TCP or UDP port number must match the specified value. Enter an integer from 0 to 65535. A value of 0 instructs the ACE to include all ports. Alternatively, you can enter the keyword name of a well-known TCP port from Table 3-6 or a well-known UDP port from Table 3-7.

Table 3-6 Well-Known TCP Port Numbers and Keywords

Keyword	Port Number	Description
domain	53	Domain Name System (DNS)
ftp	21	File Transfer Protocol (FTP)
ftp-data	20	FTP data connections
http	80	Hypertext Transfer Protocol (HTTP)
https	443	HTTP over TLS or SSL (HTTPS)
irc	194	Internet Relay Chat (IRC)
matip-a	350	Mapping of Airline Traffic over Internet Protocol (MATIP) Type A
nntp	119	Network News Transport Protocol (NNTP)
pop2	109	Post Office Protocol (POP) v2
pop3	110	Post Office Protocol (POP) v3
rdp	3389	Remote Desktop Protocol (RDP)
rtsp	554	Real-Time Streaming Protocol (RTSP)
sip	5060	Session Initiation Protocol (SIP)
skinny	2000	Skinny Client Control Protocol (SCCP)
smtp	25	Simple Mail Transfer Protocol (SMTP)
telnet	23	Telnet
www	80	World Wide Web (WWW)
xot	1998	X.25 over TCP

Note The ACE supports both the **http** and the **www** literal keywords for port 80 as well as the port number **80** itself. Regardless of whether you configure a literal keyword or port 80, the "www" literal appears in the running configuration.

Table 3-7 Well-Known UDP Port Numbers and Keywords

Keyword	Port Number	Description
domain	53	Domain Name System
radius-acct	1813	Remote Authentication Dial-In User Service (accounting)
radius-auth	1812	Remote Authentication Dial-In User Service (server)
sip	5060	Session Initiation Protocol (SIP)
wsp	9200	Connectionless Wireless Session Protocol (WSP)
wsp-wtls	9202	Secure Connectionless WSP
wsp-wtp	9201	Connection-based WSP
wsp-wtp-wtls	9203	Secure Connection-based WSP

  – **range** *port1 port2*—Specifies a port range to use for the TCP or UDP port. Valid port ranges are from 0 to 65535. A value of 0 instructs the ACE to match all ports.

- *protocol_number*—Number of an IP protocol. Enter an integer from 1 to 254 that represents an IP protocol number.

Note The ACE always attempts to match incoming traffic to the configured classes in a Layer 4 multi-match policy on a first-match basis. When you configure two or more class maps with the same VIP address match criteria and you configure the protocol as **any** in the first class map, the ACE will not match incoming traffic to a more specific class map (one with a specified protocol and port) that follows the non-specific class map in a policy map. For example, if you configure **match virtual-address 192.168.12.15 any** in the first class map and **match virtual-address 192.168.12.15 tcp eq https** in the second class map and associate the classes in that order in a policy map, any HTTPS traffic received by the ACE will never match the intended class. Therefore, the ACE will loadbalance the traffic to the wrong server and you will receive The Page Cannot Be Displayed error in your browser. Always configure the more specific class first, or else you may experience unexpected results. If you configure the two classes in separate Layer 4 policy maps, be sure to apply the policies in the correct order on the interface using a service policy.

The following example specifies that the class map L4VIPCLASS matches traffic destined to VIP address 192.168.1.10 and TCP port number 80:

```
host1/Admin(config)# class-map L4VIPCLASS
host1/Admin(config-cmap)# match virtual-address 192.168.1.10 tcp port
eq 80
```

To remove the VIP match statement from the class map, enter:

```
host1/Admin(config-cmap)# no match virtual-address 192.168.1.10 tcp
port eq 80
```

Configuring a Layer 3 and Layer 4 Policy Map for SLB

For a Layer 3 and Layer 4 SLB traffic classification, you create an SLB policy map that contains actions that are related to a VIP. A policy map associates a predefined traffic class (class map) with a series of actions to be performed on the traffic that matches the classifications defined in the traffic class. At the Layer 3 and Layer 4 network traffic level, there is a single policy map for each network traffic feature. The Layer 3 and Layer 4 policy maps are typed accordingly and, through the **service-policy** command, applied to a single interface or globally to all interfaces in a context.

The sequence in which the ACE applies actions for a specific policy are independent of the actions configured for a class inside a policy. The ACE follows an implicit feature lookup order that is dictated by the policy map actions and features, which can mean that the user-configured order of class maps does not necessarily have an effect on the lookup order. For example, if you configure one or more security ACLs in a policy map, an ACL may not allow a certain flow through the ACE even if you want to perform operations such as SLB on that flow. For details on the lookup order that the ACE uses, see the *Cisco Application Control Engine Module Administration Guide*.

To create an SLB policy map, use the **policy-map** command in configuration mode. The syntax of this command is as follows:

policy-map multi-match *map_name*

The keywords, arguments, and options are as follows:

- **multi-match**—Allows the inclusion of multiple Layer 3 and Layer 4 network traffic-related actions in the same policy map, enabling the ACE to execute all possible actions applicable for a specific classification (for example, SLB, NAT, and AAA).
- *map_name*—Unique identifier of the policy map. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a Layer 3 and Layer 4 network traffic policy map, enter:

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

To remove a policy map from the configuration, enter:

```
host1/Admin(config)# no policy-map multi-match L4SLBPOLICY
```

If there are multiple instances of actions of the same type (feature) configured in a policy map, the ACE performs the first action encountered of that type.

This section contains the following topics:

- Defining a Layer 3 and Layer 4 Policy Map Description
- Associating a Layer 3 and Layer 4 Class Map with a Policy Map
- Specifying Layer 3 and Layer 4 SLB Policy Actions

Defining a Layer 3 and Layer 4 Policy Map Description

You can use the **description** command to provide a brief summary about the Layer 3 and Layer 4 policy map.

You must access the policy map configuration mode to specify the **description** command.

The syntax of this command is as follows:

description *text*

Use the *text* argument to enter an unquoted text string with a maximum of 240 alphanumeric characters.

For example, to specify a description that the policy map is to filter network traffic to a VIP, enter:

```
host1/Admin(config-pmap)# description filter traffic matching a VIP
```

To remove the description from the class map, enter:

```
host1/Admin(config-pmap)# no description
```

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

You can associate a Layer 3 and Layer 4 SLB class map with a Layer 3 and Layer 4 SLB policy map by using the **class** command in policy-map configuration mode. The syntax of this command is as follows:

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

The keywords, arguments, and options are as follows:

- *name1*—Name of a previously defined traffic class configured with the **class-map** command. 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 none of the specified classifications match the traffic, then the ACE performs the action specified under the **class class-default** command. The **class-default** class map has an implicit match any statement in it enabling it to match all traffic.
- **insert-before** *name2*—(Optional) Places the current class map ahead of an existing class map specified by the *name2* argument in the policy-map configuration. The ACE does not preserve the command in the running configuration but does retain the configured order of class maps in the policy map.

For example, to use the **insert-before** command to define the sequential order of two class maps in the policy map, enter:

```
host1/Admin(config-pmap)# class L4VIPCLASS insert-before FILTERHTML
```

To remove a class map from a Layer 3 and Layer 4 policy map, enter:

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

Specifying Layer 3 and Layer 4 SLB Policy Actions

After you associate a Layer 3 and Layer 4 class map with an SLB policy map, you need to specify the actions that the ACE should take when network traffic matches one or more match statements in a class map. To specify the Layer 3 and Layer 4 SLB policy actions, use the commands described in the following topics:

- Associating a Layer 7 SLB Policy Map with a Layer 3 and Layer 4 SLB Policy Map
- Associating a Generic, HTTP, or RTSP Parameter Map with a Layer 3 and Layer 4 Policy Map
- Associating a Connection Parameter Map with a Layer 3 and Layer 4 Policy Map
- Enabling the Advertising of a Virtual Server IP Address
- Enabling a VIP to Reply to ICMP Requests
- Enabling Per-Packet Load Balancing for UDP Traffic
- Enabling a VIP

Associating a Layer 7 SLB Policy Map with a Layer 3 and Layer 4 SLB Policy Map

The ACE treats all Layer 7 policy maps as child policies, so you must always associate a Layer 7 SLB policy map with a Layer 3 and Layer 4 SLB policy map. You can apply on an interface or globally on all interfaces in a context only a Layer 3 and Layer 4 policy map and not a Layer 7 policy map. For details on creating a Layer 7 load-balancing policy map, see the "Configuring a Layer 7 Policy Map for SLB" section.

You can associate a Layer 7 SLB policy map with a Layer 3 and Layer 4 SLB policy map by using the **loadbalance** command in policy-map class configuration mode.

The syntax of this command is as follows:

loadbalance policy *name*

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

For example, to reference the Layer 7 L7SLBPOLICY policy map within the Layer 3 and Layer 4 L4SLBPOLICY policy 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)# serverfarm FARM2

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

To dissociate the Layer 7 SLB policy from the Layer 3 and Layer 4 SLB policy, enter:

```
host1/Admin(config-pmap-c)# no loadbalance policy L7LBPOLICY
```

Associating a Generic, HTTP, or RTSP Parameter Map with a Layer 3 and Layer 4 Policy Map

You can configure generic, HTTP, or RTSP parameters by creating a parameter map to define related actions. See the "Configuring a Generic Protocol Parameter Map", "Configuring an HTTP Parameter Map", or "Configuring an RTSP Parameter Map" section.

You can associate a parameter map with a Layer 3 and Layer 4 policy map by using the **appl-parameter advanced-options** command in policy-map class configuration mode.

The syntax of this command is as follows:

appl-parameter {**generic** | **http** | **rtsp**} **advanced-options** *name*

The keywords and arguments are as follows:

- **generic**—Specifies a generic protocol parameter map.
- **http**—Specifies an HTTP parameter map.
- **rtsp**—Specifies an RTSP parameter map.
- *name*—Identifier of an existing generic, HTTP, or RTSP parameter map. Parameter maps aggregate related traffic actions together.

For example, to specify the **appl-parameter http advanced-options** command as an action for the SLB policy map, enter:

```
host1/Admin(config)# policy-map multi-match L4SLBPOLICY
host1/Admin(config-pmap)# class FILTERHTTP
host1/Admin(config-pmap-c)# appl-parameter http advanced-options
HTTP_PARAM_MAP1
```

To disassociate the HTTP parameter map as an action from the SLB policy map, enter:

```
host1/Admin(config-pmap-c)# no appl-parameter http advanced-options
HTTP_PARAM_MAP1
```

Associating a Connection Parameter Map with a Layer 3 and Layer 4 Policy Map

You can configure TCP/IP normalization and connection parameters by creating a connection parameter map to define related actions. See the *Cisco Application Control Engine Module Security Configuration Guide*.

You can associate a connection parameter map with a Layer 3 and Layer 4 policy map by using the **connection advanced-options** command in policy-map class configuration mode. The syntax of this command is as follows:

connection advanced-options *name*

For the *name* argument, enter the name of an existing parameter map as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config-pmap-c)# connection advanced-options TCP_PARAM_MAP
```

To dissociate the TCP parameter map from a policy map, enter:

```
host1/Admin(config-pmap-c)# no connection advanced-options
TCP_PARAM_MAP
```

Enabling the Advertising of a Virtual Server IP Address

You can allow the ACE to advertise the IP address of the virtual server as the host route by using the **loadbalance vip advertise** command in policy-map class configuration mode. This function is used with route health injection (RHI) to allow the ACE to advertise the availability of a VIP address throughout the network.

The syntax of this command is as follows:

loadbalance vip advertise [**active**] | [**metric** *number*]

The keywords, arguments, and options are as follows:

- **active**—(Optional) Allows the ACE to advertise the IP address of the virtual server (VIP) as the host route only if there is at least one active real server in the server farm. Without the **active** option, the ACE always advertises the VIP whether or not there is any active real server associated with this VIP.
- **metric** *number*—(Optional) Specifies the distance metric for the route that needs to be entered in the routing table. Enter an integer from 1 to 254. The default is 77.

Note You must enable the advertising of a VIP using the **loadbalance vip advertise** command before you can enter a distance **metric** value for the route. Otherwise, the ACE returns an error message.

Note If you configured the **loadbalance vip advertise metric** command and then you enter the **no loadbalance vip advertise** [**active**] command, the ACE resets the **metric** value to the default of 77.

For example, to specify the **loadbalance vip advertise** command as an action for the SLB policy map, enter:

```
host1/Admin(config)# policy-map multi-match L4SLBPOLICY
host1/Admin(config-pmap)# class FILTERHTTP
host1/Admin(config-pmap-c)# loadbalance vip advertise active
```

To stop advertising the host route as an action in the SLB policy map and to reset the value of the **loadbalance vip advertise metric** command (if configured) to the default of 77, enter:

```
host1/Admin(config-pmap-c)# no loadbalance vip advertise active
```

To specify a distance metric for the route, enter:

```
host1/Admin(config-pmap-c)# loadbalance vip advertise metric 30
```

To remove the distance metric from the advertised VIP, enter:

```
host1/Admin(config-pmap-c)# no loadbalance vip advertise metric 30
```

Enabling a VIP to Reply to ICMP Requests

You can enable a VIP to reply to ICMP ECHO requests by using the **loadbalance vip icmp-reply** command in policy-map class configuration mode. For example, if a user sends an ICMP ECHO request to a VIP, this command instructs the VIP to send an ICMP ECHO-REPLY. The syntax of this command is as follows:

loadbalance vip icmp-reply [**active** [**primary-inservice**]]

The options are:

- **active**—(Optional) Instructs the ACE to reply to an ICMP request only if the configured VIP is active. If the VIP is not active and the **active** option is specified, the ACE discards the ICMP request and the request times out.
- **primary-inservice**—Instructs the ACE to reply to an ICMP ping only if the primary server farm state is UP, regardless of the state of the backup server farm. If this option is enabled and the primary server farm state is DOWN, the ACE discards the ICMP request and the request times out.

Note The **loadbalance vip icmp-reply** command controls a ping to a VIP on the ACE. This command implicitly downloads an ICMP access control list entry for the VIP. When you configure this command on the ACE, any configured ACLs that deny ICMP traffic have no effect on a client's ability to ping the VIP.

To complete the configuration when you configure the **active** option of this command, be sure to configure a Telnet probe and associate it with the server farm. The probe monitors the health of all the real servers in the server farm and ensures that the VIP responds with an ICMP ECHO REPLY only if the server port is active. If the server port is down or unreachable, the probe fails and the VIP stops responding to the ECHO request. For details about configuring probes, see Chapter 4, Configuring Health Monitoring.

Note For security reasons, the ACE does not allow pings from an interface on a VLAN on one side of the ACE through the module to an interface on a different VLAN on the other side of the module. For example, you cannot ping a VIP from a server if the VIP is on a VLAN that is different from the server VLAN.

For example, enter:

```
host1/Admin(config)# policy-map multi-match L4SLBPOLICY
host1/Admin(config-pmap)# class FILTERHTTP
host1/Admin(config-pmap-c)# loadbalance vip icmp-reply active
```

To view the current states of the loadbalance vip icmp-reply command, the primary server farm, and the backup server farm, use the **show service-policy** *policy-name* **detail** command. For details, see the "Displaying Service-Policy Statistics" section.

Enabling Per-Packet Load Balancing for UDP Traffic

By default, the ACE can load balance UDP DNS address (A-record) request packets using the same tuple to the same real server on an existing connection. You can close the connection immediately after a DNS response is sent back to the client, enabling per-packet load balancing for this traffic, by using the **loadbalance vip udp-fast-age** command in policy-map class configuration mode. The syntax of this command is as follows:

loadbalance vip udp-fast-age

You must configure this command under a VIP class under a Layer-4 policy. When you use this command, the ACE load balances all new DNS A-record requests to a new real server in the server farm according to the predictor algorithm. All retransmitted UDP DNS A-record packets from the client go to the same real server. This command is only applicable for UDP DNS A-record flows. It also works with fragmented packets of this type. The ACE performs the reassembly and then forwards the response, and deletes the connection record.

Note You can use the **loadbalance vip udp-fast-age** command with a generic Layer-7 policy only. For more information on configuring the generic Layer-7 policy, see the "Configuring a Layer 7 Policy Map for SLB" section.

For example, enter:

```
host1/Admin(config-pmap-c)# loadbalance vip udp-fast-age
```

To reset the default behavior, use the **no loadbalance vip udp-fast-age** command. For example, enter:

```
host1/Admin(config-pmap-c)# no loadbalance vip udp-fast-age
```

The following example provides a running configuration using the **loadbalance vip udp-fast-age** command:

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

rserver host RS1
  ip address 10.6.252.245
  inservice
rserver host RS2
  ip address 10.6.252.246
  inservice
serverfarm host SF1
  rserver RS1
    inservice
  rserver RS2
    inservice

class-map match-any GENERIC_VIP
  2 match virtual-address 10.6.252.19 udp any

policy-map type loadbalance generic first-match GENERIC_LB1
  class class-default
    serverfarm SF1

policy-map multi-match LB
  class GENERIC_VIP
    loadbalance vip udp-fast-age
    loadbalance vip inservice
    loadbalance policy GENERIC_LB1

interface vlan 10
  ip address 10.6.252.12 255.255.255.0
  access-group input TEST
  service-policy input LB
  no shutdown
```

Enabling a VIP

You can enable a VIP for SLB operations by using the **loadbalance vip inservice** command in policy-map class configuration mode. The syntax of this command is as follows:

loadbalance vip inservice

The following example specifies the **loadbalance vip inservice** command as an action for the SLB policy map:

```
host1/Admin(config)# policy-map multi-match L4SLBPOLICY
host1/Admin(config-pmap)# class FILTERHTTP
host1/Admin(config-pmap-c)# loadbalance vip inservice
```

To disable a VIP, enter:

```
host1/Admin(config-pmap-c)# no loadbalance vip inservice
```

Applying a Layer 3 and Layer 4 Policy to an Interface

Use the **service-policy** command to do the following:

- Apply a previously created policy map.
- Attach the traffic policy to a specific VLAN interface or globally to all VLAN interfaces in the same context.

The **service-policy** command is available at both the interface configuration mode and at the configuration mode. Specifying a policy map in the configuration mode applies the policy to all of the VLAN interfaces associated with a context.

The syntax of this command is as follows:

service-policy input *policy_name*

The keywords, arguments, and options are as follows:

- **input**—Specifies that the traffic policy is to be attached to the input direction of an interface. The traffic policy evaluates all traffic received by that interface.
- *policy_name*—Name of a previously defined policy map, configured with a previously created **policy-map** command. The name can be a maximum of 64 alphanumeric characters.

Follow these guidelines when creating a service policy:

- Policy maps, applied globally in a context, are internally applied on all interfaces associated with the context.
- You can apply the policy in an input direction only.
- A policy activated on an interface overwrites any specified global policies for overlapping classification and actions.

- The ACE allows only one policy of a specific feature type to be activated on a given interface.
- You can apply a maximum of 128 service policies on each interface.

When you configure multiple VIPs on an interface, the match criteria for incoming traffic follow the order in which you configure the service-policy statements on that interface. Each service policy that you configure on an interface applies a Layer 3 and Layer 4 multi-match policy map to the interface. Each multi-match policy map may contain one or more features as defined in the class maps associated with the policy map.

Because service policies do not have line numbers, the order in which you configure them on an interface is extremely important. The reason is that the ACE has an implicit feature lookup order as follows:

1. Access control (permit or deny a packet)
2. Management traffic
3. TCP normalization and connection parameters
4. Server load balancing
5. Application protocol inspection
6. Source NAT
7. Destination NAT

When you apply multiple service policies to an interface, the ACE appends the last service policy at the end of the list. If you need to change the order of the service policies on an interface, you must first remove the service policies and then add them back in the appropriate order. This process is disruptive to the network.

As an alternative to reordering the service policies, you can configure multiple class maps in the same multi-match policy map, where you can define the order of the class maps. This process is not disruptive to the network. When you add new class maps to an existing policy, use the **insert-before** command to place the new class map in the desired order.

The following example shows how to configure two class maps in a policy map, where the VIP-ACCESS-MANAGER-80 is the more specific class map. To ensure that the ACE matches traffic to the more specific classification, configure VIP-ACCESS-MANAGER-80 class map first under the LB-TRAFFIC policy map. This example and the one that follows include only the Layer 3 and Layer 4 traffic classification portions of the configuration.

```
class-map match-all VIP-ACCESS-MANAGER-ANY
  2 match virtual-address 10.238.45.200 tcp eq any
class-map match-all VIP-ACCESS-MANAGER-80
  2 match virtual-address 10.238.45.200 tcp eq www
policy-map multi-match LB-TRAFFIC
  class VIP-ACCESS-MANAGER-80
    loadbalance vip inservice
    loadbalance policy POLICY-ACCESS-MANAGER-80
    loadbalance vip icmp-reply active
  class VIP-ACCESS-MANAGER-ANY
    loadbalance vip inservice
    loadbalance policy POLICY-ACCESS-MANAGER-ANY
    loadbalance vip icmp-reply active

interface vlan 758
  description CLIENT-SIDE-VLAN
  bridge-group 100
  access-group input ALL
  service-policy input LB-TRAFFIC
  no shutdown
```

The following example shows how to use two policy maps, one for each class map, and achieve the same results as in the previous example. In this example, you configure the more specific policy map first under the interface using a service policy.

```
policy-map multi-match LB-TRAFFIC-80
  class VIP-ACCESS-MANAGER-80
    loadbalance vip inservice
    loadbalance policy POLICY-ACCESS-MANAGER-80
    loadbalance vip icmp-reply active

policy-map multi-match LB-TRAFFIC-ANY
  class VIP-ACCESS-MANAGER-ANY
    loadbalance vip inservice
    loadbalance policy POLICY-ACCESS-MANAGER-ANY
    loadbalance vip icmp-reply active
```

```
interface vlan 758
  description CLIENT-SIDE-VLAN
  bridge-group 100
  access-group input ALL
  service-policy input LB-TRAFFIC-80
  service-policy input LB-TRAFFIC-ANY
  no shutdown
```

The following example specifies an interface VLAN and applies a Layer 3 and Layer 4 SLB policy map to the VLAN:

```
host1/Admin(config)# interface vlan50
host1/Admin(config-if)# mtu 1500
host1/Admin(config-if)# ip address 172.20.1.100 255.255.0.0
host1/Admin(config-if)# service-policy input L4SLBPOLICY
```

To apply the Layer 3 and Layer 4 SLB policy map to all interfaces in the context, enter:

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

To detach a traffic policy from an interface, enter:

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

To globally detach a traffic policy from a context, enter:

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

When you detach a traffic policy either individually from the last VLAN interface on which you applied the service policy or globally from all VLAN interfaces in the same context, the ACE automatically resets the associated service-policy statistics. The ACE performs this action to provide a new starting point for the service-policy statistics the next time that you attach a traffic policy to a specific VLAN interface or globally to all VLAN interfaces in the same context.

Configuring UDP Booster

If you have a network application that requires very high UDP connection rates, configure the UDP booster feature. This feature is well suited to applications where statistical load-balancing algorithms are adequate. The most common application of this feature is DNS load balancing, but you can use it wherever you require very high UDP connection rates.

To achieve the very high connection rates with this feature, the ACE load balances new UDP requests. Then, it attempts to match subsequent UDP packets that hit the appropriate interface to a previous load-balancing result. Subsequent packets that match previous load-balancing results likely will have source IP addresses different from the loadbalanced packets. The ACE load balances UDP misses normally using the configured load balancing predictor for the server farm.

The ACE keeps track of hash hits in a special-purpose connection table. The hash range is 1 to 16384 (16 K). A full table contains 65,536 (64 K) entries, based on one client and one server for each entry, and each of the two network processors has its own independent table.

For client-side interface hits, the ACE NATs the VIP to the associated server IP address, while preserving the source IP address. For return traffic from the server, the ACE NATs the server IP address to the VIP, while preserving the destination address.

For requests where the source and destination ports are the same (common for DNS applications with a source and a destination port number of 53), the ACE translates the source port using implicit PAT. The ACE reserves the port range of 1 to 1023 for the source port hash. Therefore, incoming UDP traffic must either have identical source and destination ports or the source port must be greater than 1023. Otherwise, the feature does not work properly.

Note Do not configure UDP booster with NAT or with *any* Layer 7 feature such as per-packet UDP load balancing (also called UDP fast-age). Otherwise, unexpected results may occur. For details about per-packet UDP load balancing, see the "Enabling Per-Packet Load Balancing for UDP Traffic" section. For details about NAT, see the *Cisco Application Control Engine Module Security Configuration Guide*.

To configure the UDP booster feature, use the **udp** command in interface configuration mode. The syntax of this command is as follows:

udp {**ip-source-hash** | **ip-destination-hash**}

The keywords are as follows:

- **ip-source-hash**—Instructs the ACE to hash the source IP address of UDP packets that hit a source-hash VLAN interface before performing a connection match. Configure this keyword on a client-side interface.
- **ip-destination-hash**—Instructs the ACE to hash the destination IP address of UDP packets that hit a destination-hash VLAN interface before performing a connection match. Configure this keyword on a server-side interface.

Note To use this feature, you must configure both keywords on the appropriate interfaces, and configure standard load balancing on the ACE. Be aware that when you configure the UDP booster feature, the ACE drops all traffic in the opposite direction (for example, UDP connections initiated from real servers).

For details on configuring load balancing, see the chapters in this guide.

For example, to configure this feature on client-side VLAN interface 101 and on server-side VLAN interface 201, enter:

```
host1/Admin(config)# interface vlan 101
host1/Admin(config-if)# udp ip-source-hash
host1/Admin(config-if)# exit
host1/Admin(config)# interface vlan 201
host1/Admin(config-if)# udp ip-destination-hash
```

To remove the UDP booster feature from the above-mentioned interfaces, enter:

```
host1/Admin(config)# interface vlan 101
host1/Admin(config-if)# no udp ip-source-hash
host1/Admin(config-if)# exit
host1/Admin(config)# interface vlan 201
host1/Admin(config-if)# no udp ip-destination-hash
```

Configuring the ACE to Perform Hashing When the Source and Destination Ports Are Equal

By default, when the source and destination ports of a TCP or UDP packet are equal, the classification and distribution engine (CDE) uses the source IP address and destination IP address to perform the hash function. When they are not equal, the CDE only uses the ports.

To configure the CDE to perform the hash function using the source and destination ports when the TCP or UDP packets are equal, use the **hw-module cde-same-port-hash** command. When you configure this command, the ACE also disables implicit PAT on packets so that the source port does not change. This command is available only in the Admin context.

When this command is configured and the ports are equal, the CDE uses a slightly different hash method from the default method.

For example, enter:

```
switch/Admin(config)# hw-module cde-same-port-hash
```

To reset the default behavior, enter:

```
switch/Admin(config)# no hw-module cde-same-port-hash
```

Configuring RDP Load Balancing

The Microsoft Remote Desktop Protocol (RDP) provides users with remote display and input capabilities over network connections for Windows-based applications running on a terminal server. One of the key features supported by RDP is called roaming disconnect. With roaming disconnect, you can manually disconnect from a terminal server session without logging off the session. When you later log on to the system using either the same device or a different device, you are automatically reconnected to your disconnected session. Also, when your session is unexpectedly terminated by a network or client failure, you are disconnected but not logged off.

In a load-balancing configuration, the ACE distributes incoming session connections across the terminal servers in a server farm according to the load-balancing method configured on the server farm. The session directory (SD) keeps a list of user sessions indexed by username and allows you to reconnect to the terminal server where your disconnected session resides and to resume that session. When you authenticate yourself with a terminal server in the server farm, the SD is queried with your username. If a session with your username exists on one of the terminal servers, the SD redirects you to that terminal server. This feature allows you to disconnect a session with applications running, whether intentionally or because of a network failure, and then reconnect at a later time to the same session with the same applications running. The SD passes to the client a routing token with login information and the server IP address embedded in it.

To parse the routing token and to use the IP address embedded inside it to load balance client sessions to terminal servers, the ACE needs to look inside the RDP packet. The ACE makes the load-balancing decision based on the presence or absence of the routing token in the RDP packet. If the token is present, the ACE forwards the packet to the server that has an address and port embedded in the token. If the token is not present, the ACE uses the Layer 3 and Layer 4 load-balancing configuration to determine the real server.

Note The ACE supports RDP load balancing based on routing tokens. If the client does not send the routing token in the request to the ACE, the ACE load balances the client request to one of the available servers in the server farm using the configured predictor.

This section contains the following topics on configuring the ACE for RDP load balancing:

- Configuring Real Servers and a Server Farm
- Configuring a Layer 7 RDP Load-Balancing Policy
- Configuring a Layer 3 and Layer 4 RDP Policy
- Applying the Layer 3 and Layer 4 RDP Policy to an Interface
- Example of an RDP Load-Balancing Configuration

Configuring Real Servers and a Server Farm

For information about configuring real servers and server farms, see Chapter 2, Configuring Real Servers and Server Farms.

Configuring a Layer 7 RDP Load-Balancing Policy

To configure a Layer 7 RDP load-balancing policy, perform the following steps:

Step 1 Create an RDP Layer 7 load-balancing policy map.

```
(config)# policy-map type loadbalance rdp first-match RDP_L7_POLICY
host1/Admin(config-pmap-lb-rdp)#
```

Step 2 Associate the default class map with the policy map. The default class map is the only class map that you can associate with the Layer 7 RDP policy map.

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

Step 3 Associate the server farm with the policy map.

```
host1/Admin(config-pmap-lb-rdp-C)# serverfarm sf1
```

For more details about configuring a Layer 7 load-balancing policy, see the "Configuring a Layer 7 Class Map for SLB" section and the "Configuring a Layer 7 Policy Map for SLB" section.

Configuring a Layer 3 and Layer 4 RDP Policy

To configure a Layer 3 and Layer 4 RDP policy, perform the following steps:

Step 1 Create a Layer 3 and Layer 4 class map for RDP and specify a VIP match statement. The **rdp** keyword in the match statement sets the port to the default RDP port of 3389. You can also enter the port number directly.

```
host1/Admin(config)# class-map match-any RDP_L4_CLASS
host1/Admin(config-cmap)# match virtual-address 192.168.12.15 tcp port
eq rdp
```

Step 2 Create a Layer 3 and Layer 4 policy map and associate the Layer 3 and Layer 4 class map with it.

```
host1/Admin(config)# policy-map multi-match RDP_L4_POLICY
host1/Admin(config-pmap)# class RDP_L4_CLASS
host1/Admin(config-pmap-c)#
```

Step 3 Place the VIP inservice.

```
host1/Admin(config-pmap-c)# loadbalance vip inservice
```

Step 4 Associate the Layer 7 policy map that you created in the "Configuring a Layer 7 RDP Load-Balancing Policy" section with the Layer 3 and Layer 4 policy map.

```
host1/Admin(config-pmap-c)# loadbalance policy RDP_L7_POLICY
```

For more details about configuring a Layer 3 and Layer 4 LB policy, see the "Configuring a Layer 3 and Layer 4 Class Map for SLB" and the "Configuring a Layer 3 and Layer 4 Policy Map for SLB" sections.

Applying the Layer 3 and Layer 4 RDP Policy to an Interface

For information about applying a Layer 3 and Layer 4 load-balancing policy to an interface, see the "Applying a Layer 3 and Layer 4 Policy to an Interface" section.

Example of an RDP Load-Balancing Configuration

The following example provides a running configuration for RDP load balancing:

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

rserver host RS1
  ip address 10.6.252.245
  inservice
rserver host RS2
  ip address 10.6.252.246
  inservice

serverfarm host SF1
  rserver RS1
    inservice
  rserver RS2
    inservice

class-map match-any RDP_L4_CLASS
  2 match virtual-address 10.6.252.19 tcp eq rdp

policy-map type loadbalance rdp first-match RDP_L7_POLICY
  class class-default
    serverfarm SF1

policy-map multi-match RDP_L4_POLICY
  class RDP_L4_CLASS
    loadbalance vip inservice
    loadbalance RDP_L7_POLICY

interface vlan 10
  ip address 10.6.252.12 255.255.255.0
  access-group input ACL1
  service-policy input RDP_L4_POLICY
  no shutdown
```

Configuring RADIUS Load Balancing

The ACE uses the RADIUS protocol to load balance client requests. The ACE processes RADIUS client requests for accounting and server access by using the following:

- Layer 3 and Layer 4 load balancing to determine which server will process the first packet
- Stickiness to load balance to the same server subsequent packets from the same client based on a specific RADIUS field in those packets

If the ACE receives any retransmitted requests, it sends them to the same server where it sent the original request. The module uses the source and destination addresses and ports and a header field called "identifier" in the RADIUS header to identify a retransmission.

To ensure that the ACE forwards client data frames to the same RADIUS server where the RADIUS requests were sent, you need to configure a Layer 3 and Layer 4 load-balancing policy with a catch-all VIP and a RADIUS sticky group as the action. Upon receiving the framed IP address in the Access-Accept or the Accounting-Request message, the ACE forwards the data frames to the appropriate server.

In a load-balanced service gateway environment, the ACE supports partitioned subscriber databases across data centers by mapping the defined calling station ID or username ranges to a specific server farm. This feature allows you to partition your subscriber database so that each server farm will have a limited set of subscribers to service. You can specify subscriber attributes by configuring a RADIUS class map with match criteria set to the subscriber attributes.

The ACE does not load balance RADIUS accounting on/off messages. Instead, it replicates those messages to each real server in the server farm that is configured in the RADIUS LB policy.

This section contains the following topics on configuring RADIUS load balancing:

- Configuring Real Servers and a Server Farm
- Configuring a RADIUS Sticky Group
- Configuring a Layer 7 RADIUS Load-Balancing Policy
- Configuring a Layer 3 and Layer 4 RADIUS Load-Balancing Policy
- Configuring a Traffic Policy for Non-RADIUS Data Forwarding

- Applying a Layer 3 and Layer 4 RADIUS Policy to an Interface
- Examples of RADIUS Load-Balancing Configurations

Configuring Real Servers and a Server Farm

For information about configuring real servers and server farms, see Chapter 2, Configuring Real Servers and Server Farms.

Configuring a RADIUS Sticky Group

You can configure a RADIUS sticky group based on one of the following criteria:

- Framed IP only
- Framed IP and calling station ID
- Framed IP and username

To configure a RADIUS sticky group, perform the following steps:

Step 1 Create a RADIUS sticky group.

```
host1/admin(config)# sticky radius framed-ip calling-station-id
RADIUS_GROUP
host1/admin(config-sticky-radius)#
```

The RADIUS sticky group ensures that the ACE forwards subsequent messages that belong to the same user session (identified by username, calling ID, or framed IP) to the same server as the first message in that user session. Multiple sessions can come from the same client.

Step 2 Associate a server farm with the sticky group.

```
host1/admin(config-sticky-radius)# serverfarm sf2
```

For more information about configuring RADIUS stickiness, see Chapter 5, Configuring Stickiness.

Configuring a Layer 7 RADIUS Load-Balancing Policy

To configure a Layer 7 RADIUS load-balancing policy, perform the following steps:

Step 1 Create a Layer 7 load-balancing class map.

```
host1/Admin(config)# class-map type radius loadbalance match-any
RADIUS_L7_CLASS
host1/Admin(config-cmap-radius-lb)#
```

You can also omit the first two steps and use the default class map (class-default) instead. If you use the default class map, you cannot specify match criteria for the RADIUS calling station ID or username.

Note A match-all class map cannot have more than one of the same type of match. A match-any class map cannot have more than one of a different type of match.

Step 2 Define match criteria for the class map.

```
host1/Admin(config-cmap-radius-lb)# match radius attribute
calling-station-id 122
host1/Admin(config-cmap-radius-lb)# match radius attribute username
JSMITH
```

Step 3 Create a Layer 7 RADIUS load-balancing policy map.

```
host1/Admin(config)# policy-map type loadbalance radius first-match
RADIUS_L7_POLICY
host1/Admin(config-pmap-lb-radius)#
```

Step 4 Associate the class map that you created in Step 1 with the policy map that you created in Step 3.

```
host1/Admin(config-pmap-lb-radius)# class RADIUS_L7_CLASS
host1/Admin(config-pmap-lb-radius-c)#
```

Step 5 Associate the sticky group with the policy map.

```
host1/Admin(config-pmap-lb-radius-C)# sticky-serverfarm RADIUS_GROUP
```

Note You can specify a simple (nonsticky) server farm in the Layer 7 policy instead of a sticky server farm. If you do, the only features available would be the forwarding of retransmissions to the same real server and replication of accounting on or off to all real servers.

For more details about configuring a Layer 7 load-balancing policy, see the "Configuring a Layer 7 Class Map for SLB" section and the "Configuring a Layer 7 Policy Map for SLB" section.

Configuring a Layer 3 and Layer 4 RADIUS Load-Balancing Policy

To configure a Layer 3 and Layer 4 RADIUS policy, perform the following steps:

Step 1 Configure a Layer 3 and Layer 4 class map for RADIUS load balancing and specify a VIP match statement. The **radius-auth** keyword in the match statement sets the port to the default RADIUS port of 1812 and the **radius-acct** keyword sets the port to the default RADIUS port or 1813. You can also enter the port numbers directly.

```
host1/Admin(config)# class-map match-any RADIUS_L4_CLASS
host1/Admin(config-cmap)# match virtual-address 192.168.12.15 udp eq
radius-auth
host1/Admin(config-cmap)# match virtual-address 192.168.12.15 udp eq
radius-acct
```

You can also use a single match statement with a range of ports as follows:

```
host1/Admin(config-cmap)# match virtual-address 192.168.12.15 udp
range 1812 1813
```

Step 2 Configure a Layer 3 and Layer 4 RADIUS policy map and associate the Layer 3 and Layer 4 RADIUS class map that you created in Step 1 with it.

```
host1/Admin(config)# policy-map multi-match RADIUS_L4_POLICY
host1/Admin(config-pmap)# class RADIUS_L4_CLASS
host1/Admin(config-pmap-c)#
```

Step 3 Place the VIP inservice.

```
host1/Admin(config-pmap-c)# loadbalance vip inservice
```

Step 4 Associate the Layer 7 policy map that you created in the "Configuring a Layer 7 RADIUS Load-Balancing Policy" section with the Layer 3 and Layer 4 policy map.

```
host1/Admin(config-pmap-c)# loadbalance policy RADIUS_L7_POLICY
```

For more details about configuring a Layer 3 and Layer 4 LB policy, see the "Configuring a Layer 3 and Layer 4 Class Map for SLB" and the "Configuring a Layer 3 and Layer 4 Policy Map for SLB" sections.

For examples of RADIUS load-balancing policies, see the "Examples of RADIUS Load-Balancing Configurations" section.

Configuring a Traffic Policy for Non-RADIUS Data Forwarding

You can configure the ACE to forward non-RADIUS data packets from a client to the same server where the ACE sent the RADIUS packets from the same client. This feature is used in mobile wireless implementations with Cisco Services Selection Gateways (SSGs). The end user traffic passes through the load balancer and must be forwarded to the same real server (SSG) that authenticated or accounted for the user identified by the framed IP.

When the ACE needs to forward a non-RADIUS packet using framed-IP stickiness, it uses the source IP address of the packet to look up the framed IP address in the sticky database. If the lookup succeeds, the ACE sends the packet to the appropriate real server IP address as the next hop (the destination IP address of the packet does not change). If the lookup fails, the ACE routes the packet.

Note This procedure includes the use of a catch-all VIP (IP address and netmask of 0.0.0.0 0.0.0.0). Use this type of VIP very carefully. Whenever possible, use a VIP address and netmask that are as close as possible to the actual traffic that the ACE needs to forward.

The following procedure contains the steps necessary to configure those parts of the configuration that pertain to the data forwarding process. Use this procedure with the previous three configuration procedures.

Step 1 Create a Layer 4 class map with a catch-all VIP.

```
host1/Admin(config)# class-map match-any LAYER4_CLASS
host1/Admin(config-cmap)# 4 match virtual-address 0.0.0.0 0.0.0.0 any
host1/Admin(config-cmap)# exit
```

Step 2 Create a simple Layer 7 load-balancing policy map.

```
host1/Admin(config)# policy-map type loadbalance first-match
DATA_FORWARD_L7POLICY
host1/Admin(config-pmap-lb)#
```

Step 3 Associate the default class map with the Layer 7 policy map to match any non-RADIUS incoming traffic. The default class map is the only class map that you can configure under this policy map.

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

Step 4 Associate the RADIUS sticky group that you created in the "Configuring a RADIUS Sticky Group" section with the Layer 7 policy map.

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

Step 5 Associate the Layer 4 class map with the Layer 4 multimatch policy map that you created in the "Configuring a Layer 3 and Layer 4 RADIUS Load-Balancing Policy" section.

```
host1/Admin(config)# policy-map multi-match RADIUS_L4_POLICY
host1/Admin(config-pmap)# class LAYER4_CLASS
```

Step 6 Enable the catch-all VIP that you configured in Step 1.

```
host1/Admin(config-pmap-c)# loadbalance vip inservice
```

Step 7 Associate the Layer 7 load-balancing policy that you created in Step 2 with the Layer 4 multimatch policy.

```
host1/Admin(config-pmap-c)# loadbalance policy DATA_FORWARD_L7POLICY
```

For an example of the entire configuration for a data forwarding RADIUS traffic policy, see the "End User Data Forwarding Policy" section.

Applying a Layer 3 and Layer 4 RADIUS Policy to an Interface

For information about applying a Layer 3 and Layer 4 load balancing policy to an interface, see the "Applying a Layer 3 and Layer 4 Policy to an Interface" section.

Examples of RADIUS Load-Balancing Configurations

The following configuration examples provide RADIUS load-balancing configurations with and without a Layer 7 RADIUS class map:

- Without a Layer 7 RADIUS Class Map
- With a Layer 7 RADIUS Class Map
- End User Data Forwarding Policy

Without a Layer 7 RADIUS Class Map

The following configuration example shows the running configuration for RADIUS load-balancing without a Layer 7 RADIUS class map. The parts of the configuration that pertain strictly to the RADIUS load-balancing configuration are shown in bold text.

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

rserver host RS1
  ip address 10.6.252.245
  inservice
rserver host RS2
  ip address 10.6.252.246
  inservice

serverfarm host SF1
  rserver RS1
    inservice
  rserver RS2
    inservice

sticky radius framed-ip calling-station-id RADIUS_GROUP
  serverfarm SF1
```

```
class-map match-any RADIUS_L4_CLASS
  2 match virtual-address 12.1.1.11 udp range 1812 1813

policy-map type loadbalance radius first-match RADIUS_L7_POLICY
  class class-default
    sticky-serverfarm RADIUS_GROUP

policy-map multi-match RADIUS_L4_POLICY
  class RADIUS_L4_CLASS
    loadbalance vip inservice
    loadbalance RADIUS_L7_POLICY

interface vlan 10
  ip address 192.168.12.13 255.255.255.0
  service-policy input RADIUS_DATA_L4_POLICY
  no shutdown
```

With a Layer 7 RADIUS Class Map

The following configuration example shows the running configuration for RADIUS load-balancing with a Layer 7 RADIUS class map. The parts of the configuration that pertain strictly to the RADIUS load-balancing configuration are shown in bold text.

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

rserver host RS1
  ip address 10.6.252.245
  inservice
rserver host RS2
  ip address 10.6.252.246
  inservice

serverfarm host SF1
  rserver RS1
    inservice
  rserver RS2
    inservice

sticky radius framed-ip calling-station-id RADIUS_GROUP1
  serverfarm SF1

sticky radius framed-ip calling-station-id RADIUS_GROUP2
  serverfarm SF2

class-map match-any RADIUS_L4_CLASS
  2 match virtual-address 192.168.12.15 udp range 1812 1813
```

```
class-map type radius loadbalance match-any RADIUS_L7_CLASS1
   2 match radius attribute calling-station-id 122*
   3 match radius attribute calling-station-id 133*

class-map type radius loadbalance match-any RADIUS_L7_CLASS2
   2 match radius attribute calling-station-id 144*

policy-map type loadbalance radius first-match RADIUS_L7_POLICY
  class RADIUS_L7_CLASS1
    sticky-serverfarm RADIUS_GROUP1
  class RADIUS_L7_CLASS2
    sticky-serverfarm RADIUS_GROUP2

policy-map multi-match RADIUS_L4_POLICY
  class RADIUS_L4_CLASS
    loadbalance vip inservice
    loadbalance RADIUS_L7_POLICY

interface vlan 10
  ip address 192.168.12.12 255.255.255.0
  service-policy input RADIUS_L4_POLICY
  no shutdown
```

End User Data Forwarding Policy

The following configuration example provides the commands necessary to instruct the ACE to forward non-RADIUS data packets to the same RADIUS server where the ACE sent the first RADIUS packet. The parts of the configuration that pertain strictly to the data forwarding policy are shown in bold text.

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

rserver host RS1
  ip address 10.6.252.245
  inservice
rserver host RS2
  ip address 10.6.252.246
  inservice

serverfarm host SF1
  rserver RS1
    inservice
  rserver RS2
    inservice
```

```
sticky radius framed-ip RADIUS_GROUP1
  serverfarm SF1

class-map type radius loadbalance match-any RADIUS_L7_CLASS
  2 match radius attribute calling-station-id 133*

class-map match-any RADIUS_L4_CLASS
  3 match virtual-address 192.168.12.15 udp range 1812 1813

class-map match-any LAYER4_CLASS
  4 match virtual-address 0.0.0.0 0.0.0.0 any

policy-map type loadbalance radius first-match RADIUS_L7_POLICY
  class RADIUS_L7_CLASS
    sticky-serverfarm RADIUS_GROUP1

policy-map type loadbalance first-match DATA_FORWARD_L7POLICY
  class class-default
    sticky-serverfarm RADIUS_GROUP1

policy-map multi-match RADIUS_L4_POLICY
  class RADIUS_L4_CLASS
    loadbalance vip inservice
    loadbalance policy RADIUS_L7_POLICY
    loadbalance vip icmp-reply
    loadbalance vip advertise
  class LAYER4_CLASS
    loadbalance vip inservice
    loadbalance policy LAYER7_POLICY

interface vlan 10
  ip address 192.168.12.12 255.255.255.0
  service-policy input RADIUS_L4_POLICY
  no shutdown
```

Configuring RTSP Load Balancing

The Real-Time Streaming Protocol (RTSP) is used for streaming audio, video, and simulation data from a server to a client by such applications as the following:

- Cisco IP/TV
- Apple QuickTime 4
- RealAudio
- RealNetworks
- RealPlayer

The ACE supports RTSP over TCP. The ACE bases Layer 3 and Layer 4 RTSP load-balancing decisions on the configured VIP and RTSP port. For Layer 7 load balancing, the ACE uses the RTSP URL or RTSP header. The format of the RTSP URL is rtsp://cisco.com/video.

An RTSP session can have multiple request and response message exchanges over the same or different connections. For example, when a media player tries to play a media file, the following message exchange may occur:

- OPTIONS query the server for the supported features (some players skip this).
- DESCRIBE retrieves the description of a media resource.
- SETUP specifies the transport mechanism to be used for the media.
- PLAY tells the server to start sending streamed data via the mechanism specified in SETUP.
- PAUSE causes the stream delivery to be interrupted temporarily.
- TEARDOWN stops the stream delivery and frees the resources.

When the ACE receives the first request message (usually OPTIONS or DESCRIBE) for a new RTSP session, it makes a load-balancing decision based on the configured policy and forwards the message to the selected server. The ACE forwards all subsequent messages in this session to the same server. RTSP messages are text based and similar to HTTP.

RTSP sticky uses the Session header to stick a client to a server. The Session header contains a session ID assigned by the server in the SETUP response. When the server responds to the SETUP request, the ACE creates an entry in the sticky database with the session ID from the server. All subsequent message exchanges will contain the same Session header.

Note If you expect all RTSP requests for the same session to arrive over a single connection, the sticky part of the configuration is optional.

The inspection part of the configuration is optional. However, to support RTSP data traffic running on a separate connection over Real-Time Transport Protocol (RTP), you must configure RTSP inspection. If you configure inspection, the ACE performs an inspection and fixes the RTSP packets, including any necessary rewriting of the packet and opening of the restricted ports. For information about RTSP inspection, see the *Cisco Application Control Engine Module Security Configuration Guide*.

This section contains the following topics on configuring RTSP load balancing:

- Configuring Real Servers and a Server Farm
- Configuring an RTSP Sticky Group
- Configuring a Layer 7 RTSP Load-Balancing Policy
- Configuring a Layer 3 and Layer 4 RTSP Load-Balancing Policy
- Applying a Layer 3 and Layer 4 RTSP Policy to an Interface
- Example of an RTSP Load-Balancing Configuration

Configuring Real Servers and a Server Farm

For information about configuring real servers and server farms, see Chapter 2, Configuring Real Servers and Server Farms.

Configuring an RTSP Sticky Group

This section describes how to configure RTSP stickiness. It is an optional procedure and is not required for RTSP load balancing.

To configure an RTSP sticky group, perform the following steps:

Step 1 Create an RTSP sticky group. The RTSP sticky group ensures that the ACE sends subsequent requests for the same session to the same server as the first request for that session.

Note If you expect all RTSP requests for the same URL to arrive over a single connection, the sticky part of the configuration is optional.

```
host1/admin(config)# sticky rtsp-header Session RTSP_GROUP
host1/admin(config-sticky-rtsp)#
```

Step 2 Associate a server farm with the sticky group.

```
host1/admin(config-sticky-rtsp)# serverfarm SF4
```

For more information about configuring stickiness, see Chapter 3, Configuring Stickiness.

Configuring a Layer 7 RTSP Load-Balancing Policy

To configure a Layer 7 RTSP load-balancing policy, perform the following steps:

Step 1 Create a Layer 7 RTSP load-balancing class map. Y

```
host1/Admin(config)# class-map type rtsp loadbalance match-any
RTSP_L7_CLASS
host1/Admin(config-cmap-rtsp-lb)#
```

Step 2 Define match criteria for the class map.

```
host1/Admin(config-cmap-rtsp-lb)# match rtsp url
rtsp://cisco.com/movie
host1/Admin(config-cmap-rtsp-lb)# match rtsp header Accept
header-value application/sdp
host1/Admin(config-cmap-rtsp-lb)# match source-address 192.168.12.15
255.255.255.0
```

Step 3 Create a Layer 7 RTSP load-balancing policy map.

```
host1/Admin(config)# policy-map type loadbalance rtsp first-match
RTSP_L7_POLICY
host1/Admin(config-pmap-lb-rtsp)#
```

Step 4 Associate the class map that you created in Step 1 (or the default class) with the policy map that you created in Step 3.

```
host1/Admin(config-pmap-lb-rtsp)# class RTSP_L7_CLASS
host1/Admin(config-pmap-lb-rtsp-c)#
```

Step 5 Associate the sticky group with the policy map.

```
host1/Admin(config-pmap-lb-rtsp-C)# sticky-serverfarm RTSP_GROUP
```

For more details about configuring a Layer 7 load balancing policy, see the "Configuring a Layer 7 Class Map for SLB" section and the "Configuring a Layer 7 Policy Map for SLB" section.

Configuring a Layer 3 and Layer 4 RTSP Load-Balancing Policy

To configure a Layer 3 and Layer 4 RTSP load-balancing policy, perform the following steps:

Step 1 Configure a Layer 3 and Layer 4 class map for RTSP load balancing and specify a VIP match statement. The **rtsp** keyword in the match statement sets the port to the default RTSP port.

```
host1/Admin(config)# class-map match-all RTSP_L4_CLASS
host1/Admin(config-cmap)# match virtual-address 192.168.12.15 tcp eq
rtsp
```

Step 2 Configure a Layer 3 and Layer 4 RTSP policy map and associate the Layer 3 and Layer 4 RTSP class map that you created in Step 1 with it.

```
host1/Admin(config)# policy-map multi-match RTSP_L4_POLICY
host1/Admin(config-pmap)# class RTSP_L4_CLASS
host1/Admin(config-pmap-c)#
```

Step 3 Place the VIP inservice.

```
host1/Admin(config-pmap-c)# loadbalance vip inservice
```

Step 4 Associate the Layer 7 policy map that you created in the "Configuring a Layer 7 RADIUS Load-Balancing Policy" section with the Layer 3 and Layer 4 policy map.

```
host1/Admin(config-pmap-c)# loadbalance policy RTSP_L7_POLICY
```

For more details about configuring a Layer 3 and Layer 4 load balancing policy, see the "Configuring a Layer 3 and Layer 4 Class Map for SLB" and the "Configuring a Layer 3 and Layer 4 Policy Map for SLB" sections.

Applying a Layer 3 and Layer 4 RTSP Policy to an Interface

For information about applying a Layer 3 and Layer 4 load-balancing policy to an interface, see the "Applying a Layer 3 and Layer 4 Policy to an Interface" section.

Example of an RTSP Load-Balancing Configuration

The following is a sample RTSP configuration. The sticky group and RTSP inspection portions are optional. If a client uses different connections for multiple requests in one session, you must configure the sticky group. If you use RTP for data traffic that runs on separate connections, an inspection is needed to open the proper pinholes.

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

rserver host RS1
  ip address 10.6.252.245
  inservice
```

```
rserver host RS2
  ip address 10.6.252.246
  inservice

serverfarm host SF4
  rserver RS1
    inservice
  rserver RS2
    inservice

sticky rtsp-header Session RTSP_GROUP
  serverfarm SF4

class-map type rtsp loadbalance match-any RTSP_L7_CLASS
  match rtsp url rtsp://cisco.com/movie
  match rtsp header Accept header-value application/sdp

class-map match-all RTSP_L4_CLASS
  match virtual-address 192.168.12.15 tcp eq rtsp

policy-map type loadbalance rtsp first-match RTSP_L7_POLICY
  class RTSP_L7_CLASS
    sticky-serverfarm RTSP_GROUP

policy-map multi-match RTSP_L4_POLICY
  class RTSP_L4_CLASS
    loadbalance vip inservice
    loadbalance policy RTSP_L7_POLICY
    inspect rtsp

interface vlan 10
  ip address 192.168.12.12 255.255.255.0
  service-policy input RTSP_L4_POLICY
  no shutdown
```

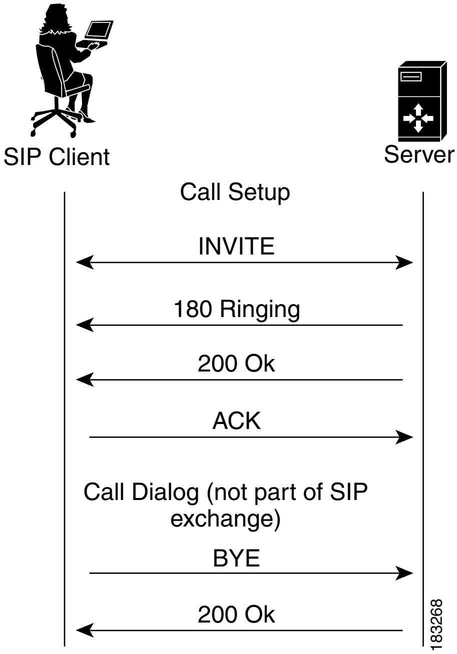
Configuring SIP Load Balancing

The Session Initiation Protocol (SIP) functions as a signaling mechanism between user devices and media servers. SIP is a peer-to-peer protocol where end-devices (the user agent clients) initiate interactive communications sessions with SIP servers. These sessions can include Internet multimedia conferences, Internet telephone calls (Voice-over-IP), and multimedia distribution. Examples of client devices include hardware, software, handheld IP telephones, and personal digital assistants (PDAs). The ACE supports SIP over TCP or UDP.

The session Call-ID is a unique call identifier that resides in the text-based SIP messages sent from the client to the SIP proxy server (for example, a Cisco Softswitch placed between the clients and the ACE). A proxy server can reuse the same connection for multiple calls to the ACE simultaneously. The ACE independently load balances each call that has a different Call-ID, but keeps the back-end connections open at the same time. This behavior is different from HTTP persistence-rebalance where the ACE closes the existing connection after it makes the new load-balancing decision. For more information about persistence-rebalance, see the "Configuring HTTP Persistence Rebalance" section.

During a SIP session, the client and the server may exchange many messages. See Figure 3-2.

Figure 3-2 Example of a SIP Call Setup and Call Dialog



When the ACE receives the first request message (usually INVITE) from a client for a new SIP session, it makes a load-balancing decision based on the configured policy and forwards the message to the selected server. The ACE forwards all subsequent messages that have the same Call-ID in this session to the same server.

Call-ID stickiness ensures that messages for a particular Call-ID from different TCP or UDP connections reach the correct servers. Stickiness by Call-ID is particularly important for stateful call services that use the Call-ID to identify current SIP sessions and make decisions based on the content of a message.

Note If you expect all SIP requests for the same Call-ID to arrive over a single connection, then the sticky part of the configuration is optional.

If SIP stickiness is configured and the ACE finds the Call-ID in the header of the SIP messages sent from the client to the server, the ACE generates a key (hash value) based on the Call-ID. The ACE uses the key to look up an entry in the sticky table. If the entry exists, the ACE sends the client to the sticky server indicated by the table entry. If the entry does not exist, the ACE creates a new sticky entry, hashes the SIP Call-ID value into a key, and saves the key in the entry.

In most cases, you need to enable at least basic SIP inspection (configuring the **inspect sip** command without an inspection policy) for SIP load balancing to work. SIP inspection fixes the SIP packets, including any necessary rewriting of the packet and opening of restricted ports. These actions ensure that SIP traffic is routed properly through the ACE. For information about SIP inspection, see the *Cisco Application Control Engine Module Security Configuration Guide*.

This section contains the following topics on configuring SIP load balancing:

- Configuring Real Servers and a Server Farm
- Configuring a SIP Sticky Group
- Configuring a Layer 7 SIP Load-Balancing Policy
- Configuring a Layer 3 and Layer 4 SIP Load-Balancing Policy
- Applying a Layer 3 and Layer 4 SIP Policy to an Interface
- Example of a SIP Load-Balancing Configuration

Configuring Real Servers and a Server Farm

For information about configuring real servers and server farms, see Chapter 2, Configuring Real Servers and Server Farms.

Configuring a SIP Sticky Group

This section describes how to configure SIP stickiness. It is an optional procedure and is not required for SIP load balancing.

To configure a SIP sticky group, perform the following steps:

Step 1 Create a SIP sticky group.

```
host1/admin(config)# sticky sip-header Call-ID SIP_GROUP
host1/admin(config-sticky-sip)#
```

The SIP sticky group ensures that the ACE sends subsequent packets from the same client to the same server as the first packet from that client.

Note If you expect all SIP requests for the same Call-ID to arrive over a single connection, the sticky part of the configuration is optional.

Step 2 Associate a server farm with the sticky group.

```
host1/admin(config-sticky-sip)# serverfarm sf3
```

For more information about configuring SIP stickiness, see Chapter 3, Configuring Stickiness.

Configuring a Layer 7 SIP Load-Balancing Policy

To configure a Layer 7 SIP load-balancing policy, perform the following steps:

Step 1 Create a Layer 7 SIP LB class map.

```
host1/Admin(config)# class-map type sip loadbalance match-any
SIP_L7_CLASS
host1/Admin(config-cmap-sip-lb)#
```

You can also omit the first two steps and use the default class map (class-default) instead. If you use the default class map, you cannot specify match criteria for other SIP headers. Instead, the ACE performs the action specified under the **class class-default** command.

The class-default class map contains an implicit match-any statement that enables it to match all traffic. Additionally, the ACE ensures that messages with the same Call-ID are load balanced to the same server. For an example configuration, see the "SIP Load Balancing Without Match Criteria" section.

Step 2 Define match criteria for the class map.

```
host1/Admin(config-cmap-sip-lb)# match sip header To header-value
.*@cisco.com
```

Step 3 Create a Layer 7 SIP load-balancing policy map.

```
host1/Admin(config)# policy-map type loadbalance sip first-match
SIP_L7_POLICY
host1/Admin(config-pmap-lb-sip)#
```

Step 4 Associate the class map that you created in Step 1 (or the default class) with the policy map that you created in Step 3.

```
host1/Admin(config-pmap-lb-sip)# class SIP_L7_CLASS
host1/Admin(config-pmap-lb-sip-c)#
```

Step 5 Associate the sticky group with the policy map.

```
host1/Admin(config-pmap-lb-sip-C)# sticky-serverfarm SIP_GROUP
```

For more details about configuring a Layer 7 load-balancing policy, see the "Configuring a Layer 7 Class Map for SLB" section and the "Configuring a Layer 7 Policy Map for SLB" section.

Configuring a Layer 3 and Layer 4 SIP Load-Balancing Policy

To configure a Layer 3 and Layer 4 SIP load-balancing policy, perform the following steps:

Step 1 Configure a Layer 3 and Layer 4 class map for SIP load balancing and specify a VIP match statement. The **sip** keyword in the match statement sets the port to the default SIP port of 5060.

```
host1/Admin(config)# class-map match-all SIP_L4_CLASS
host1/Admin(config-cmap)# match virtual-address 192.168.12.15 udp eq
sip
```

Step 2 Configure a Layer 3 and Layer 4 SIP policy map and associate the Layer 3 and Layer 4 SIP class map that you created in Step 1 with it.

```
host1/Admin(config)# policy-map multi-match SIP_L4_POLICY
host1/Admin(config-pmap)# class SIP_L4_CLASS
host1/Admin(config-pmap-c)#
```

Step 3 Place the VIP inservice.

```
host1/Admin(config-pmap-c)# loadbalance vip inservice
```

Step 4 Associate the Layer 7 policy map that you created in the "Configuring a Layer 7 RADIUS Load-Balancing Policy" section with the Layer 3 and Layer 4 policy map.

```
host1/Admin(config-pmap-c)# loadbalance policy SIP_L7_POLICY
```

For more details about configuring a Layer 3 and Layer 4 load-balancing policy, see the "Configuring a Layer 3 and Layer 4 Class Map for SLB" and the "Configuring a Layer 3 and Layer 4 Policy Map for SLB" sections.

Applying a Layer 3 and Layer 4 SIP Policy to an Interface

For information about applying a Layer 3 and Layer 4 LB policy to an interface, see the "Applying a Layer 3 and Layer 4 Policy to an Interface" section.

Example of a SIP Load-Balancing Configuration

The following examples are from a sample SIP configuration:

- SIP Load Balancing Without Match Criteria
- SIP Load Balancing Based on SIP headers and SIP Inspection

The sticky group and SIP inspection are optional. If you do not configure class-map match criteria, the ACE performs the action specified under the **class class-default** command. The class-default class map contains an implicit match any statement that enables it to match all traffic. Additionally, the ACE ensures that messages with the same Call-ID are load balanced to the same server.

SIP Load Balancing Without Match Criteria

The following configuration example shows the running configuration for SIP load-balancing with match criteria. The parts of the configuration that pertain strictly to the SIP load-balancing configuration are shown in bold text.

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

rserver host RS1
  ip address 10.6.252.245
  inservice
rserver host RS2
  ip address 10.6.252.246
  inservice
serverfarm host SF1
  rserver RS1
    inservice
  rserver RS2
    inservice

sticky sip-header Call-ID SIP_GROUP
  serverfarm SF3

class-map match-all SIP_L4_CLASS
  match virtual-address 192.168.12.15 udp eq sip

policy-map type loadbalance sip first-match SIP_L7_POLICY
  class class-default
    sticky-serverfarm SIP_GROUP
```

```
policy-map multi-match SIP_L4_POLICY
  class SIP_L4_CLASS
    loadbalance vip inservice
    loadbalance policy SIP_L7_POLICY
    inspect sip

interface vlan 10
  ip address 192.168.12.12 255.255.255.0
  service-policy input SIP_L4_POLICY
  no shutdown
```

SIP Load Balancing Based on SIP headers and SIP Inspection

The following configuration example shows the running configuration for SIP load-balancing on SIP headers and SIP inspection. The parts of the configuration that pertain strictly to the SIP load-balancing configuration are shown in bold text.

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

rserver host RS1
  ip address 10.6.252.245
  inservice
rserver host RS2
  ip address 10.6.252.246
  inservice

serverfarm host SF3
  rserver RS1
    inservice
  rserver RS2
    inservice

sticky sip-header Call-ID SIP_GROUP
  serverfarm SF3

class-map type sip loadbalance match-any SIP_L7_CLASS
  match sip header Call_ID header-value sip:

class-map match-all SIP_L4_CLASS
  match virtual-address 192.168.12.15 tcp eq sip

policy-map type loadbalance sip first-match SIP_L7_POLICY
  class SIP_L7_CLASS
    sticky-serverfarm SIP_GROUP
```

```
policy-map multi-match SIP_L4_POLICY
  class SIP_L4_CLASS
    loadbalance vip inservice
    loadbalance policy SIP_L7_POLICY
    inspect sip

interface vlan 10
  ip address 192.168.12.12 255.255.255.0
  service-policy input SIP_L4_POLICY
  no shutdown
```

Example of a Server Load-Balancing Policy Configuration

The following example shows a running configuration that includes multiple class maps and policy maps that define a traffic policy for SLB. The class map and policy map configuration appears in bold in the example.

In this configuration, when a server farm is chosen for a connection, the connection is sent to a real server based on one of several load-balancing predictors. The leastconns predictor method load balances connections to the server that has the lowest number of open connections.

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

probe tcp TCP
  interval 5
  faildetect 2
  passdetect interval 10
  open 3

parameter-map type http PERSIST-REBALANCE
  persistence-rebalance
parameter-map type connection PRED-CONNS-UDP_CONN
  set timeout inactivity 300
serverfarm host PRED-CONNS
  predictor leastconns
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice
  rserver SERVER4
    inservice
  rserver SERVER5
    inservice
  rserver SERVER6
    inservice
  rserver SERVER7
    inservice
  rserver SERVER8
    inservice
serverfarm host PRED-CONNS-UDP
  failaction purge
  predictor leastconns
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    probe ICMP
    inservice
  rserver SERVER5
    inservice
  rserver SERVER6
    inservice
  rserver SERVER7
    inservice
serverfarm host PREDICTOR
  probe TCP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER6
    inservice
  rserver SERVER7
    inservice

sticky http-cookie COOKIE_TEST STKY-GRP-43
  cookie offset 1 length 999
  timeout 30
  replicate sticky
  serverfarm PREDICTOR

class-map match-all L4PRED-CONNS-UDP-VIP_128:2222_CLASS
  2 match virtual-address 192.168.120.128 udp eq 0
class-map match-all L4PRED-CONNS-VIP_128:80_CLASS
  2 match virtual-address 192.168.120.128 tcp eq www
```

```
class-map match-all L4PREDICTOR_117:80_CLASS
  2 match virtual-address 192.168.120.117 tcp eq www
policy-map type loadbalance first-match L7PLBSF_PRED-CONNS_POLICY
  class class-default
    serverfarm PRED-CONNS
policy-map type loadbalance first-match L7PLBSF_PRED-CONNS-UDP_POLICY
  class class-default
    serverfarm PRED-CONNS-UDP
policy-map type loadbalance first-match L7PLBSF_PREDICTOR_POLICY
  class class-default
    sticky-serverfarm STKY-GRP-43
policy-map multi-match L4SH-Gold-VIPs_POLICY
  class L4PREDICTOR_117:80_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_PREDICTOR_POLICY
    loadbalance vip icmp-reply active
    nat dynamic 1 vlan 120
    appl-parameter http advanced-options PERSIST-REBALANCE
  class L4PRED-CONNS-VIP_128:80_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_PRED-CONNS_POLICY
    loadbalance vip icmp-reply active
    nat dynamic 1 vlan 120
    appl-parameter http advanced-options PERSIST-REBALANCE
  class L4PRED-CONNS-UDP-VIP_128:2222_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_PRED-CONNS-UDP_POLICY
    loadbalance vip icmp-reply active
    nat dynamic 1 vlan 120
    appl-parameter http advanced-options PERSIST-REBALANCE
    connection advanced-options PRED-CONNS-UDP_CONN

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 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```

Displaying Load-Balancing Configuration Information and Statistics

Use the commands in the following sections to display configuration information and statistics for Layer 3 and Layer 4, and Layer 7 class maps and policy maps. This section contains the following topics:

- Displaying Class-Map Configuration Information
- Displaying Policy-Map Configuration Information
- Displaying Parameter Map Configuration Information
- Displaying Load-Balancing Statistics
- Displaying HTTP Parameter Map Statistics
- Displaying Service-Policy Statistics
- Displaying the Layer 7 Match HTTP URL Statement Hit Counts
- Displaying HTTP Statistics

Displaying Class-Map Configuration Information

You can display class-map configuration information by using the **show running-config class-map** command in Exec mode. This command displays the names of all the class maps configured in the contexts to which you have access and the match statements configured in each class map. The syntax of this command is as follows:

show running-config class-map

Displaying Policy-Map Configuration Information

You can display policy-map configuration information by using the **show running-config policy-map** command in Exec mode. This command displays the names of all the policy maps configured in the contexts to which you have access and the class maps and actions configured in each policy map. The syntax of this command is as follows:

show running-config policy-map

Displaying Parameter Map Configuration Information

You can display a list of parameter maps and their configurations by using the **show running-config parameter-map** command in Exec mode. The syntax of this command is as follows:

show running-config parameter-map

Displaying Load-Balancing Statistics

You can display load-balancing statistics by using the **show stats loadbalance** command in Exec mode. The syntax of this command is as follows:

show stats loadbalance [**radius** | **rdp**]

The keywords and options are as follows:

- **radius**—(Optional) Displays Remote Authentication Dial-In User Service (RADIUS) load-balancing statistics associated with the current context.
- **rdp**—(Optional) Displays Reliable Datagram Protocol (RDP) load-balancing statistics associated with the current context.

For example, to see all load-balancing statistics, enter:

```
host1/Admin# show stats loadbalance
```

Table 3-8 describes the fields in the **show stats loadbalance** command output for an HTTP parameter map.

Table 3-8 Field Descriptions for the show stats loadbalance Command Output

Field	Description
Total Version Mismatch	VIP configuration changed during the load-balancing decision and the ACE rejected the connection.
Total Layer4 Decisions	Total number of times that the ACE made a load-balancing decision at Layer 4.
Total Layer4 Rejections	Total number of Layer 4 connections that the ACE rejected.

Table 3-8 Field Descriptions for the show stats loadbalance Command Output (continued)

Field	Description
Total Layer7 Decisions	Total number of times that the ACE made a load-balancing decision at Layer 7.
Total Layer7 Rejections	Total number of Layer 7 connections that the ACE rejected.
Total Layer4 LB Policy Misses	Total number of connection requests that did not match a configured Layer 4 load-balancing policy.
Total Layer7 LB Policy Misses	Total number of connection requests that did not match a configured Layer 7 load-balancing policy.
Total Times Rserver Was Unavailable	Total number of times that the ACE attempted to load balance a request to a real server in a server farm, but the real server was not in service or was otherwise unavailable.
Total ACL denied	Total number of connection requests that were denied by an ACL.
Total IDMap Lookup Failures	Total number of times that the ACE failed to find the local ACE to peer ACE ID mapping for a real-server or a server-farm in the ID Map table for redundancy. A failure can occur if the peer ACE did not send a proper remote ID for the local ACE to look up and so the local ACE could not perform a mapping or if the ID Map table was not created.

Table 3-9 describes the fields in the **show stats loadbalance radius** command output.

Table 3-9 Field Descriptions for the show stats loadbalance radius Command Output

Field	Description
Total Requests Received	Total number of RADIUS client (NAS) requests received
Total Responses Received	Total number of RADIUS server responses received

Table 3-9 Field Descriptions for the show stats loadbalance radius Command Output (continued)

Field	Description
Total Retry Packets Received	Total number of retransmitted RADIUS requests received
Total Header Parse Results Received	Total number of RADIUS headers parsed and received by the load balance module
Total Body Parse Results Received	Total number of RADIUS attributes parsed and received by the load balance module
Total Data Parse Results Received	Total number of RADIUS packets (requests and responses) parsed and received by the load balance module
Total Packets Sent Out	Total number of RADIUS packets sent out by the ACE on both inbound and outbound interfaces
Total Sessions Allocated	Total number of RADIUS sessions allocated
Total Sessions Deleted	Total number of RADIUS sessions deleted
Total Username Sticky Added	Total number of sticky entries added based on the User-Name attribute
Total Calling-station Sticky Added	Total number of sticky entries added based on the Calling-Station-Id attribute
Total Framed-ip Sticky Added	Total number of sticky entries added based on the Framed-IP-Address attribute
Total End-user Packet Sticky Success	Total number of End-user packets that hit a FIP sticky entry previously created during RADIUS AAA processing
Total End-user Packet Sticky Failure	Total number of End-user packets that failed to hit a FIP sticky entry previously created during RADIUS AAA processing

Table 3-9 Field Descriptions for the show stats loadbalance radius Command Output (continued)

Field	Description
Total Acct-On/Off Requests Received	Total number of RADIUS Accounting On/Off requests received
Total Acct-On/Off Responses Received	Total number of RADIUS Accounting On/Off responses received
Total Acct-On/Off with No Rules	Total number of RADIUS Accounting On/Off requests for which no valid policy is found
Total Acct-On/Off Req Processing Done	Total number of RADIUS Accounting On/Off requests processed successfully by the ACE
Total NULL Packet Received Errors	Total number of invalid (with no data) RADIUS packets received
Total Parse Errors	Total number of RADIUS packets received with no valid RADIUS header
Total Proxy Mapper Errors	Number of proxy mapper entries creation failure. The proxy mapper entry structure maps each inbound connection with multiple outbound connections for point-to-multipoint protocols
Total Sticky Addition Failures	Total number of failures in creating a RADIUS-based sticky entry
Total memory allocation failures	Total number of failures in the allocation of any of the main RADIUS structures is use
Total stale packet errors	Total number of stale RADIUS packets upon receiving a new RADIUS request

Table 3-10 describes the fields in the **show stats loadbalance rdp** command output.

Table 3-10 Field Descriptions for the show stats loadbalance rdp Command Output

Field	Description
Total Parse Results Received	Total number of RDP parse results received by the load balance module
Total Packets Load Balanced	Total number of RDP packets load balanced
Total Packets with Routing Token	Total number of RDP packets received containing a Routing Token
Total Packets with Token Matching No Rserver	Total number for RDP packets for which the Routing Token does not match any of the configured real servers

Displaying HTTP Parameter Map Statistics

You can display statistics for an HTTP parameter map by using the **show parameter-map** command in Exec mode. The syntax of this command is as follows:

show parameter-map [*name*]

The optional *name* argument is the identifier of an HTTP parameter map. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to display statistics for the HTTP parameter map called HTTP_PARAMMAP, enter:

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

Table 3-11 describes the fields in the **show parameter-map** command output for an HTTP parameter map.

Table 3-11 Field Descriptions for the show parameter-map Command Output

Field	Description
Parameter-map	Unique identifier of the HTTP parameter map.
Type	HTTP.
Server-side connection reuse	Status of TCP server reuse feature: enabled or disabled.
Case-insensitive parsing	Status of the **case-insensitive** command: enabled or disabled.
Persistence-rebalance	Status of the **persistence-rebalance** command: strict, enabled or disabled.
Header-maxparse-length	Configured value or the default value of the **header-maxparse-length** command.
Content-maxparse-length	Configured value or the default value of the **content-maxparse-length** command.
Parse length-exceed action	Configured action for the **length-exceed** command: continue or drop.
Urlcookie-delimiters	Configured URL cookie delimiters.

Table 3-11 Field Descriptions for the show parameter-map Command Output (continued)

Field	Description
Urlcookie-start	Displays the start string of the secondary cookie or the none setting configured by the **set secondary-cookie-start** command in parameter map HTTP configuration mode. The default string of ?.
User-agent	Text string in the request to match as specified in the HTTP parameter map. A user agent is a client that initiates a request. Examples of user agents include browsers, editors, or other end user tools. The ACE does not compress the response to a request when the request contains a matching user agent string. The maximum size is 64 characters. The default is none.

Displaying Service-Policy Statistics

You can display statistics for service policies enabled globally within a context or on a specific interface by using the **show service-policy** command. If you do not enter an option with this command, the ACE displays all enabled policy statistics. This command also allows you to display statistics for a specific class map in a policy or a summary of statistics. The syntax of this command is as follows:

show service-policy [*policy_name* [**class-map** *class_name*]] [**detail** | **summary** | **url-summary**]

The keywords, arguments, and options are as follows:

- *policy_name*—(Optional) The name of an existing policy map that is currently in service (applied to an interface). Enter an unquoted text string with no spaces. If you do not enter the name of an existing policy map, the ACE displays information and statistics for all policy maps.
- **class-map** *class_name*—(Optional) Displays the statistics for the specified class map associated with the policy.
- **detail**—(Optional) Displays detailed statistics and status for the policy or class map.

- **summary**—(Optional) Displays a summary of policy or class map statistics and status information in a table format.
- **url-summary**—(Optional) Displays the number of times that a connection is established (hit count) based on match HTTP URL statements for a class map in a Layer 7 (L7) HTTP policy map. For information on this option and descriptions of the fields, see the "Displaying the Layer 7 Match HTTP URL Statement Hit Counts" section.

Note The ACE updates the counters that the **show service-policy** command displays after the applicable connections are closed.

For example, to display detailed statistics and current status of the service policy MGMT_POLICYMAP, enter:

```
host1/Admin# show service-policy MGMT_POLICYMAP detail
```

Table 3-12 describes the fields in the **show service-policy** command output.

Table 3-12 Field Descriptions for the show service-policy Command Output

Field	Description
Status	Current operational state of the service policy. Possible states are ACTIVE or INACTIVE.
Description	User-entered description of the policy map.
Interface	VLAN ID of the interface to which the policy map has been applied
Service Policy	Unique identifier of the policy map.
Class	Name of the class map associated with the service policy.
Loadbalance	
L7 Policy	Name of the Layer 7 policy map associated with the service policy.
VIP Route Metric	Specifies the distance metric for the route as specified with the **loadbalance vip advertise** command. The ACE writes the value you specify in its routing table. Possible values are integers from 1 to 254.

Table 3-12 Field Descriptions for the show service-policy Command Output (continued)

Field	Description
VIP Route Advertise	Operational state of the **loadbalance vip advertise** command: ENABLED or DISABLED. This command is used with route health injection (RHI) to allow the ACE to advertise the availability of a VIP address throughout the network.
VIP ICMP Reply	Operational state of the **loadbalance vip icmp-reply** command. Possible states are: ENABLED, DISABLED, ENABLED-WHEN-ACTIVE, or ENABLED-WHEN-PRIMARY-SF-UP.
VIP State	Operational state of the virtual server: INSERVICE or OUTOFSERVICE.
Curr Conns	Number of active connections to the VIP.
Hit Count	Number of times a connection was established with this VIP.
Dropped Conns	Number of connections that the ACE discarded.
Client Pkt Count	Number of packets received from the client.
Client Byte Count	Number of bytes received from the client.
Server Pkt Count	Number of packets received from the server.
Server Byte Count	Number of bytes received from the server.
Max-conn-limit	Configured value of the **conn-limit max** command. See Chapter 2, Configuring Real Servers and Server Farms.
Drop-count	Number of connections that were dropped because the maximum connection limit was exceeded.
Conn-rate-limit	Configured value of the **rate-limit connection** command. See Chapter 2, Configuring Real Servers and Server Farms.
Drop-count	Number of connections that were dropped because the connection rate limit was exceeded.

Table 3-12 Field Descriptions for the show service-policy Command Output (continued)

Field	Description
bandwith-rate-limit	Configured value of the **rate-limit bandwidth** command. See Chapter 2, Configuring Real Servers and Server Farms.
Drop-count	Number of connections that were dropped because the bandwidth rate limit was exceeded.
L4 Policy Stats	
Total Req/ Resp	Total number of requests and responses for the policy map.
Total Allowed	Total number of packets received and allowed.
Total Dropped	Total number of packets received and discarded.
Total Logged	Total number of errors logged.
L7 loadbalance policy	Identifier of the Layer 7 policy map.
Class-map	Identifier of the associated class map.
LB action	Actions specified within the Layer 7 policy map as follows: • **Sticky group—**Name of the sticky group associated with this policy. • **Primary server farm**—identifier of the primary server farm • **State**—Current state of the primary server farm: UP or DOWN • **Backup server farm**—Identifier of the backup server farm • **State**—Current state of the primary server farm: UP or DOWN
Hit Count	Cumulative number of connections to the primary or backup server farm.
Dropped Conns	Number of attempted connections to the primary or backup server farm that the ACE discarded.

Table 3-12 Field Descriptions for the show service-policy Command Output (continued)

Field	Description
Hit count	Number of times a connection was established with this policy.
Dropped conns	Number of connections associated with this policy that were dropped.

Displaying the Layer 7 Match HTTP URL Statement Hit Counts

You can display the number of times that a connection is established (hit count) based on match HTTP URL statements for a class map in a Layer 7 (L7) HTTP policy map by using the **show service-policy url-summary** command. If you do not enter a policy map name with this command, the ACE displays the match URL statement hit counts for all class maps in L7 HTTP policy maps.

The syntax of this command is as follows:

show service-policy [*policy_name* [**class-map** *class_name*]] **url-summary**

The options are as follows:

- *policy_name*—(Optional) Name of an existing Layer 3 and Layer 4 HTTP policy map. Enter an unquoted text string with no spaces. If you do not enter a policy map name with this command, the ACE displays the match URL statement hit counts for all class maps in L7 HTTP policy maps.
- **class-map** *class_name*—(Optional) Displays the statement hit counts for the specified class map associated with the policy. Enter the name as an unquoted text string with no spaces.

For example, to display the hit count for the match HTTP URL statements for all class maps in all policy maps, enter the following command:

```
host1/Admin# show service-policy url-summary
```

Table 3-13 describes the fields in the **show service-policy url-summary** command output.

Table 3-13 Field Descriptions for the show service-policy url-summary Command Output

Field	Description
Service Policy	Unique identifier of the policy map.
L3-Class	Name of the Layer 3 class map associated with the service policy.
L7-Class	Identifier of the Layer 7 class map.
match http url	The HTTP URL match statement.
hit	The number of times that a connection is established based on a specific URL match statement. **Note** The URL hit counter is per match statement per load-balancing Layer 7 policy. If you are using the same combination of Layer 7 policy and class maps with URL match statements in different VIPs, the count is combined. If the ACE configuration exceeds 64K URL and load-balancing policy combinations, this counter displays NA.

Displaying HTTP Statistics

You can display HTTP statistics, including header insertion and server reuse statistics by using the **show stats http** command in Exec mode. The syntax of this command is as follows:

show stats http

For example, enter:

```
host1/Admin# show stats http

+------------------------------------------+
+-------------- HTTP statistics -----------+
+------------------------------------------+
 LB parse result msgs sent : 0          , TCP data msgs sent       : 0
 Inspect parse result msgs : 0          , SSL data msgs sent       : 0
                      sent
 TCP fin msgs sent         : 0          , TCP rst msgs sent       : 0
 Bounce fin msgs sent      : 0          , Bounce rst msgs sent    : 0
 SSL fin msgs sent         : 0          , SSL rst msgs sent       : 0
 Drain msgs sent           : 0          , Particles read           : 0
 Reuse msgs sent           : 0          , HTTP requests            : 0
 Reproxied requests        : 0          , Headers removed          : 0
 Headers inserted          : 0          , HTTP redirects           : 0
 HTTP chunks               : 0          , Pipelined requests       : 0
 HTTP unproxy conns        : 0          , Pipeline flushes         : 0
 Whitespace appends        : 0          , Second pass parsing      : 0
 Response entries recycled : 0          , Analysis errors          : 0
 Header insert errors      : 0          , Max parselen errors      : 0
 Static parse errors       : 0          , Resource errors          : 0
 Invalid path errors       : 0          , Bad HTTP version errors  : 0
 Headers rewritten         : 0          , Header rewrite errors   : 0
 Unproxy msgs sent        : 0
```

Clearing SLB Statistics

This section describes the commands that you can use to clear load-balancing statistics. It includes the following topics:

- Clearing Load-Balancing Statistics
- Clearing Service-Policy Statistics
- Clearing HTTP Statistics

Clearing Load-Balancing Statistics

You can clear all load-balancing statistics in the current context by using the **clear stats loadbalance** command in Exec mode. The syntax of this command is as follows:

clear stats loadbalance [**radius** | **rdp** | **rtsp** | **sip**]

The keywords and options are as follows:

- **radius**—(Optional) Clears Remote Authentication Dial-In User Service (RADIUS) load-balancing statistical information.
- **rdp**—(Optional) Clears Reliable Datagram Protocol (RDP) load-balancing statistical information.
- **rtsp**—(Optional) Clears Real-Time Streaming Protocol (RTSP) load-balancing statistical information.
- **sip**—(Optional) Clears Session Initiation Protocol (SIP) load-balancing statistical information.

For example, enter:

```
host1/Admin# clear stats loadbalance
```

Note If you have redundancy configured, you need to explicitly clear load-balancing statistics on both the active and the standby ACEs. Clearing statistics on the active module only leaves the standby module’s statistics at the old values.

Clearing Service-Policy Statistics

You can clear service policy statistics by using the **clear service-policy** command. The syntax of this command is as follows:

clear service-policy *policy_name*

For the *policy_name* argument, enter the identifier of an existing policy map that is currently in service (applied to an interface).

For example, to clear the statistics for the policy map L4SLBPOLICY that is currently in service, enter:

```
host1/Admin# clear service-policy L4SLBPOLICY
```

Note If you have redundancy configured, you need to explicitly clear service-policy statistics on both the active and the standby ACEs. Clearing statistics on the active module only leaves the standby module's statistics at the old values.

Clearing HTTP Statistics

You can clear all HTTP statistics in the current context by using the **clear stats http** command in Exec mode. The syntax of this command is as follows:

clear stats http

For example, enter:

```
host1/Admin# clear stats http
```

Note If you have redundancy configured, you need to explicitly clear HTTP statistics on both the active and the standby ACEs. Clearing statistics on the active module only leaves the standby module's statistics at the old values.

Where to Go Next

To configure health probes for your real servers, go to Chapter 4, Configuring Health Monitoring. To configure stickiness (connection persistence), see Chapter 5, Configuring Stickiness. To configure firewall load balancing (FWLB), see Chapter 6, Configuring Firewall Load Balancing.

CHAPTER 4

Configuring Health Monitoring

This chapter describes how to configure health monitoring on the ACE to track the state of a server by sending out probes. Also referred to as out-of-band health monitoring, the ACE verifies the server response or checks for any network problems that can prevent a client to reach a server. Based on the server response, the ACE can place the server in or out of service, and can make reliable load-balancing decisions.

You can also use health monitoring to detect failures for a gateway or a host in high-availability (redundancy) configurations. For more information, see the *Cisco Application Control Engine Module Administration Guide*.

The ACE identifies the health of a server in the following categories:

- Passed—The server returns a valid response.
- Failed—The server fails to provide a valid response to the ACE and is unable to reach a server for a specified number of retries.

By configuring the ACE for health monitoring, the ACE sends active probes periodically to determine the server state. The ACE supports 4096 unique probe configurations, which includes ICMP, TCP, HTTP, and other predefined health probes. The ACE can execute only up to 200 concurrent script probes at a time. The ACE also allows the opening of 2048 sockets simultaneously.

You can associate the same probe with multiple real servers or server farms. Each time that you use the same probe again, the ACE counts it as another probe instance. You can allocate a maximum of 16 K probe instances.

This chapter contains the following major sections:

- Clearing Probe Statistics
- Where to Go Next

Configuring Active Health Probes

By default, no active health probes are configured on the ACE. You can configure health probes on the ACE to actively make connections and explicitly send traffic to servers. The probes determine whether the health status of a server passes or fails by its response.

Configuring active probes is a three-step process:

1. Configure the health probe with a name, type, and attributes.
2. Associate the probe with one of the following:
   - A real server.
   - A real server and then associate the real server with a server farm. You can associate a single probe or multiple probes to real servers within a server farm.
   - A server farm. All servers in the server farm receive probes of the associated probe types.
3. Activate the real server or server farm.

For information on associating a probe with a real server or a server farm, and putting it into service, see Chapter 2, Configuring Real Servers and Server Farms.

You can also configure one or more probes to track a gateway or host. For more information, see the *Cisco Application Control Engine Module Administration Guide*.

This section contains the following topics:

- Defining an Active Probe and Accessing Probe Configuration Mode
- Configuring General Probe Attributes
- Configuring an ICMP Probe
- Configuring a TCP Probe
- Configuring a UDP Probe
- Configuring an Echo Probe

- Configuring a Finger Probe
- Configuring an HTTP Probe
- Configuring an HTTPS Probe
- Configuring an FTP Probe
- Configuring a Telnet Probe
- Configuring a DNS Probe
- Configuring an SMTP Probe
- Configuring an IMAP Probe
- Configuring a POP3 Probe
- Configuring a SIP Probe
- Configuring an RTSP Probe
- Configuring a RADIUS Probe
- Configuring an SNMP-Based Server Load Probe
- Configuring a Scripted Probe
- Example of a UDP Probe Load-Balancing Configuration

Defining an Active Probe and Accessing Probe Configuration Mode

When you initially configure a health probe, you define its type and name. The CLI then enters the probe configuration mode, which allows you to configure the attributes for the probe type.

To define a probe and access its configuration mode, use the **probe** command in configuration mode. The syntax of this command is as follows:

probe *probe_type probe_name*

The arguments are as follows:

- *probe_type*—Probe type that determines what the probe sends to the server. Enter one of the following keywords:
  - **icmp**—Specifies an Internet Control Message Protocol (ICMP) probe type and accesses its configuration mode. For configuration details, see the "Configuring an ICMP Probe" section.
  - **tcp**—Specifies a TCP probe type and accesses its configuration mode. For configuration details, see the "Configuring a TCP Probe" section.
  - **udp**—Specifies a UDP probe type and accesses its configuration mode. For configuration details, see the "Configuring a UDP Probe" section.
  - **echo** {**tcp** | **udp**}—Specifies an ECHO TCP or UDP probe type and accesses its configuration mode. For configuration details, see the "Configuring an Echo Probe" section.
  - **finger**—Specifies a Finger probe type and accesses its configuration mode. For configuration details, see the "Configuring a Finger Probe" section.
  - **http**—Specifies an HTTP probe type and accesses its configuration mode. For configuration details, see the "Configuring an HTTP Probe" section.
  - **https**—Specifies an HTTPS probe type for SSL and accesses its configuration mode. For configuration details, see the "Configuring an HTTPS Probe" section.
  - **ftp** —Specifies an FTP probe type and accesses its configuration mode. For configuration details, see the "Configuring an FTP Probe" section.
  - **telnet**—Specifies a Telnet probe type and accesses its configuration mode. For configuration details, see the "Configuring a Telnet Probe" section.
  - **dns**—Specifies a DNS probe type and accesses its configuration mode. For configuration details, see the "Configuring a DNS Probe" section.
  - **smtp**—Specifies a Simple Mail Transfer Protocol (SMTP) probe type and accesses its configuration mode. For configuration details, see the "Configuring an SMTP Probe" section.
  - **imap**—Specifies an Internet Message Access Protocol (IMAP) probe type and accesses its configuration mode. For configuration details, see the "Configuring an IMAP Probe" section.

  - **pop**—Specifies a POP probe type and accesses its configuration mode. For configuration details, see the "Configuring a POP3 Probe" section.
  - **sip** {**tcp** | **udp**}—Specifies the SIP TCP or UDP probe and accesses its configuration mode. For configuration details, see the "Configuring a SIP Probe" section.
  - **rtsp**—Specifies the RTSP probe and accesses its configuration mode. For configuration details, see the "Configuring an RTSP Probe" section.
  - **radius**—Specifies a RADIUS probe type and accesses its configuration mode. For configuration details, see the "Configuring a RADIUS Probe" section.
  - **snmp**—Specifies an SNMP-based server load probe type and accesses its configuration mode. For configuration details, see the "Configuring an SNMP-Based Server Load Probe" section.
  - **scripted**—Specifies a scripted probe type and accesses its configuration mode. For configuration details, see the "Configuring a Scripted Probe" section. For information on scripts, see Appendix A, "Using TCL Scripts with the ACE".

- *probe_name*—Name that you want to assign to the probe. Use the probe name to associate the probe to the real server or server farm. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a TCP probe named PROBE1 and access the TCP probe configuration mode, enter:

```
host1/Admin(config)# probe tcp PROBE1
host1/Admin(config-probe-tcp)#
```

To delete a TCP probe named PROBE1, enter:

```
host1/Admin(config)# no probe tcp PROBE1
```

Some probe attributes and associated commands apply to all probe types. For information on configuring these attributes, see the "Configuring General Probe Attributes" section.

Configuring General Probe Attributes

When you access probe configuration mode to configure the attributes for the probe, the ACE provides a set of commands that you can configure for all probe types, except as indicated. The following topics describe how to configure the general attributes for a probe:

- Configuring a Probe Description
- Configuring the Destination IP Address
- Configuring the Port Number
- Configuring the Time Interval Between Probes
- Configuring the Retry Count for Failed Probes
- Configuring the Wait Period and Threshold for Successful Probes
- Configuring the Wait Interval for the Opening of the Connection
- Configuring the Timeout Period for a Probe Response

Configuring a Probe Description

You can provide a description for a probe by using the **description** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

description *text*

The *text* argument is a description of the probe. Enter a text string with a maximum of 240 alphanumeric characters.

For example, to configure a description THIS PROBE IS FOR TCP SERVERS, enter:

```
host1/Admin(config-probe-type)# description THIS PROBE IS FOR TCP
SERVERS
```

To remove the description for the probe, use the **no description** command. For example, enter:

```
host1/Admin(config-probe-type)# no description
```

Configuring the Destination IP Address

By default, the probe uses the IP address from the real server or server farm configuration for the destination IP address. You can configure the destination address that the probe uses by using the **ip address** command. This command is available for all probe-type configuration modes except scripted. The syntax of this command is as follows:

ip address *ip_address* [**routed**]

The argument and option are as follows:

- *ip_address*—Destination IP address. Enter a unique IPv4 address in dotted-decimal notation (for example, 192.8.12.15).
- **routed**—(Optional) Specifies that the ACE will route the address according to the ACE internal routing table.

Note For HTTPS probes, the non-routed mode (without the **routed** keyword) behaves the same as the routed mode.

For example, to configure an IP address of 192.8.12.15, enter:

```
host1/Admin(config-probe-type)# ip address 192.8.12.15
```

To reset the default behavior of the probe using the IP address from the real server or server farm configuration, use the **no ip address** command. For example, enter:

```
host1/Admin(config-probe-type)# no ip address
```

Configuring the Port Number

By default, the probe uses the port number based on its type. Table 4-1 lists the default port numbers for each probe type.

Table 4-1 Default Port Numbers for Probe Types

Probe Type	Default Port Number
DNS	53
Echo	7

Table 4-1 Default Port Numbers for Probe Types (continued)

Probe Type	Default Port Number
Finger	79
FTP	21
HTTP	80
HTTPS	443
ICMP	Not applicable
IMAP	143
POP3	110
RADIUS	1812
RTSP	554
SIP (both TCP and UDP)	5060
SNMP	161
SMTP	25
TCP	80
Telnet	23
UDP	53

To configure the port number that the probe uses, use the **port** command. This command is available for all probe-type configuration modes except ICMP. The syntax of this command is as follows:

port *number*

The *number* argument is the number of the port. Enter a number from 1 to 65535.

For example, to configure a port number of 88 for an HTTP probe, enter:

```
host1/Admin(config-probe-http)# port 88
```

To reset the port number to its default value, use the **no port** command. For example, to remove an HTTP probe port number of 88 and reset an HTTP probe port number to its default setting of 80, enter:

```
host1/Admin(config-probe-http)# no port
```

Configuring the Time Interval Between Probes

The time interval between probes is the frequency that the ACE sends probes to the server marked as passed. You can change the time interval between probes by using the **interval** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

interval *seconds*

The *seconds* argument is the time interval in seconds. Enter a number from 2 to 65535. By default, the time interval is 120.

The open timeout value for TCP- or UDP- based probes and the receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to these timeout values and the server takes a long time to respond or it fails to reply within the timeout values, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For UDP probes or UDP-based probes, we recommend a time interval value of 30 seconds. The reason for this recommendation is that the ACE data plane has a management connection limit of 100,000. Management connections are used by all probes as well as Telnet, SSH, SNMP, and other management applications. In addition, the ACE has a default timeout for UDP connections of 120seconds. This means that the ACE does not remove the UDP connections even though the UDP probe has been closed for two minutes. Using a time interval less than 30 seconds may limit the number of UDP probes that can be configured to run without exceeding the management connection limit, which may result in skipped probes.

For example, to configure a time interval of 50 seconds, enter:

```
host1/Admin(config-probe-type)# interval 50
```

To reset the time interval to the default setting of 120, use the **no interval** command. For example, enter:

```
host1/Admin(config-probe-type)# no interval
```

Configuring the Retry Count for Failed Probes

Before the ACE marks a server as failed, it must detect that probes have failed a consecutive number of times. By default, when three consecutive probes have failed, the ACE marks the server as failed. You can configure this number of failed probes by using the **faildetect** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

faildetect *retry_count*

The *retry_count* argument is the consecutive number of failed probes before marking the server as failed. Enter a number from 1 to 65535. By default, the count is 3.

For example, to configure the number of failed probes at 5 before declaring the server as failed, enter:

```
host1/Admin(config-probe-type)# faildetect 5
```

To reset the number of probe failures to the default setting of 3, use the **no faildetect** command. For example, enter:

```
host1/Admin(config-probe-type)# no faildetect
```

Configuring the Wait Period and Threshold for Successful Probes

After the ACE marks a server as failed, it waits a period of time (passdetect interval) and then sends a probe to the failed server. When the ACE receives a successful response and the configured number of consecutive successful responses from the server (passdetect count) is not reached, it uses the passdetect interval and sends another probe. This process continues until the passdetect count is reached. Then, the ACE marks the server as passed. By default, the ACE waits 300 seconds before sending out a probe to a failed server and marks a server as passed if it receives 3 consecutive successful responses.

Note After a probe recovers and its state changes to Success, the ACE uses the passdetect interval before sending the next probe. For subsequent successful probes, the ACE uses the interval as configured by the **interval** command.

To configure the time interval after which the ACE sends a probe to a failed server and the number of consecutive successful probes required to mark the server as passed, use the **passdetect** command. This command is available for all probe-type configuration modes.

The syntax of this command is as follows:

passdetect {**interval** *seconds* | **count** *number*}

The keyword and argument are as follows:

- **interval** *seconds*—Specifies the wait time interval in seconds. Enter a number from 2 to 65535. The default is 300.

  Note For best results, we recommend that you do not configure a **passdetect interval** value of less than 30 seconds. If you configure a **passdetect interval** value of less than 30 seconds, the **open timeout** and **receive timeout** values are set to their default values of 10 seconds each, and a real server fails to respond to a probe, overlapping probes may result, which can cause management resources to be consumed unnecessarily and the No. Probes skipped counter to increase.

- **count** *number*—Specifies the number of successful probe responses from the server. Enter a number from 1 to 65535. The default is 3.

Note The receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to this timeout value and the server takes a long time to respond or it fails to reply within the timeout value, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For example, to configure wait interval at 60 seconds, enter:

```
host1/Admin(config-probe-type)# passdetect interval 60
```

For example, to configure five success probe responses from the server before declaring it as passed, enter:

```
host1/Admin(config-probe-type)# passdetect count 5
```

To reset the wait interval to its default setting, use the **no passdetect interval** command. For example, enter:

```
host1/Admin(config-probe-type)# no passdetect interval
```

To reset the successful probe responses to its default setting, use the **no passdetect count** command. For example, enter:

```
host1/Admin(config-probe-type)# no passdetect count
```

Configuring the Wait Interval for the Opening of the Connection

When the ACE sends a probe, it waits for the SYN-ACK after sending a SYN to open and then send an ACK to establish the connection with the server. You can configure the time interval for a connection to be established by using the **open** command. This command is available in Echo TCP, Finger, FTP, HTTP, HTTPS, IMAP, POP, scripted, SIP, SMTP, TCP, and Telnet probe configuration mode (all TCP-based probes). The syntax of this command is as follows:

open *timeout*

The *timeout* argument is the time to wait in seconds to open a connection with a server. Enter an integer from 1 to 65535. The default wait interval is 10 seconds.

Note The open timeout value for TCP-based probes and the receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to these timeout values and the server takes a long time to respond or it fails to reply within the timeout values, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For example, to configure the wait time interval to 25 seconds, enter:

```
host1/Admin(config-probe-type)# open 25
```

To reset the time interval to its default setting of 10 seconds, use the **no open** command. For example, enter:

```
host1/Admin(config-probe-type)# no open
```

Configuring the Timeout Period for a Probe Response

By default, when the ACE sends a probe, it expects a response within a time period of 10 seconds. For example, for an HTTP probe, the timeout period is the number of seconds to receive an HTTP reply for a GET or HEAD request. If the server fails to respond to the probe, the ACE marks the server as failed.

You can configure this time period to receive a server response to the probe by using the **receive** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

receive *timeout*

The *timeout* argument is the timeout period in seconds. Enter a number from 1 to 65535. By default, the timeout period is 10.

Note The open timeout value for TCP-based probes and the receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to these timeout values and the server takes a long time to respond or it fails to reply within the timeout values, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For example, to configure the timeout period for a response at 5 seconds, enter:

```
host1/Admin(config-probe-type)# receive 5
```

To reset the time period to receive a response from the server to its default setting of 10 seconds, use the **no receive** command.

For example, enter:

```
host1/Admin(config-probe-type)# no receive
```

Configuring an ICMP Probe

An ICMP probe sends an ICMP echo request and listens for a response. If a server returns a response, the ACE marks the server as passed. If the server does not send a response causing the probe to time out, or if the server sends an unexpected ICMP echo response type, the ACE marks the probe as failed.

You can create an ICMP probe and access its configuration mode by using the **probe icmp** *name* command in configuration mode.

For example, to define an ICMP probe named PROBE3 and access its mode, enter:

```
host1/Admin(config)# probe icmp PROBE3
host1/Admin(config-probe-icmp)#
```

After you create an ICMP probe, you can configure the attributes in the "Configuring General Probe Attributes" section.

Configuring a TCP Probe

A TCP probe initiates a TCP 3-way handshake and expects the server to send a response. By default, a successful response causes the probe to mark the server as passed. The probe then sends a FIN to end the session. If the response is not valid or if there is no response, the probe marks the server as failed.

Optionally, you can configure the probe to send an RST or specific data, and to expect a specific response in order to mark the server as passed. You can also configure the probe to send specific data and receive a specific response from the server. If the response is not valid or there is no response, the probe marks the server as failed.

You can create a TCP probe and access its configuration mode by using the **probe tcp** *name* command in configuration mode.

For example, to define a TCP probe named PROBE1 and access its mode, enter:

```
host1/Admin(config)# probe tcp PROBE1
host1/Admin(config-probe-tcp)#
```

You can configure attributes for a TCP probe, as described in the following topics:

- Configuring the Termination of the TCP Connection
- Configuring an Expected Response String from the Server
- Configuring Data that the Probe Sends to the Server Upon Connection

You can also configure the attributes described in the "Configuring General Probe Attributes" section.

Configuring the Termination of the TCP Connection

A TCP probe makes a connection, and if the connection through a 3-way handshake (SYN, SYN-ACK, and ACK) is successful, when the ACE receives FIN-ACK from the server, the server is marked as passed. By default, the ACE terminates a TCP connection gracefully by sending a FIN to the server.

Note If a probe is configured for the default graceful connection termination (FIN) and the target server does not send the expected data, the probe terminates the TCP connection to the server with a reset (RST). The probe will continue to send a RST to terminate the server connection as long as the returned data is not the expected data. When the server responds with the correct data again, the probe reverts to terminating the connection with a FIN.

To configure the ACE to terminate a TCP connection by sending an RST, use the **connection term** command. This command is available for TCP-based connection-oriented probes (ECHO TCP, Finger, FTP, HTTP, HTTPS, IMAP, POP, RTSP, SIP TCP, SMTP, TCP, and Telnet probe configuration modes). The syntax of this command is as follows:

connection term forced

For example, enter:

```
host1/Admin(config-probe-tcp)# connection term forced
```

To reset the method to terminate a connection gracefully, use the **no connection term** command. For example, enter:

```
host1/Admin(config-probe-tcp)# no connection term forced
```

Configuring an Expected Response String from the Server

When you configure a probe to expect a regular expression (regex) response string from a server, it searches the response for it. If the ACE finds it, the server is marked as passed. If you do not configure an expected string, the ACE ignores the server response.

You can configure what the ACE expects as a response string from the probe destination server by using the **expect regex** command. This command is available in Finger, HTTP, HTTPS, SIP, TCP, and UDP probe configuration modes.

Note For HTTP or HTTPS probes, the server response must include the Content-Length header for the **expect regex** command to function. Otherwise, the probe does not attempt to parse the regex.

When you configure the **expect regex** command for a TCP probe, you must configure the **send-data** command for the expect function to work. Otherwise, the TCP probe makes a socket connection and disconnects without checking the data.

The syntax of this command is as follows:

expect regex *string* [**offset** *number*]

The argument and option are as follows:

- *string*—Regular expression expected response string from the probe destination. Enter an unquoted text string with no spaces. If the string includes spaces, enclose the string in quotes. The string can be a maximum of 255 alphanumeric characters.
- **offset** *number*—(Optional) Sets the number of characters into the received message or buffer where the ACE starts searching for the defined expression. Enter a number from 1 to 4000.

For example, to configure the ACE to expect a response string of ack, enter:

```
host1/Admin(config-probe-tcp)# expect regex ack
```

To remove the expectation of a response string, use the **no expect regex** command. For example, enter:

```
host1/Admin(config-probe-http)# no expect regex
```

Configuring Data that the Probe Sends to the Server Upon Connection

You can configure the ASCII data that the probe sends when the ACE connects to the server by using the **send-data** command. This command is available in Echo, Finger, TCP, and UDP probe configuration modes. The syntax of this command is as follows:

send-data *expression*

The *expression* argument is the data that the probe sends. Enter an unquoted text string with a maximum of 255 alphanumeric characters including spaces.

Note If you do not configure the **send-data** command for a UDP probe, the probe sends one byte, 0x00.

When you configure the **expect regex** command for a TCP probe, you must configure the **send-data** command for the expect function to work. Otherwise, the TCP probe makes a socket connection and disconnects without checking the data.

For example, to configure the probe to send TEST as the data, enter:

```
host1/Admin(config-probe-tcp)# send-data test
```

To remove the data, use the **no send-data** command. For example, enter:

```
host1/Admin(config-probe-tcp)# no send-data
```

Configuring a UDP Probe

Note When configuring a UDP probe, you must configure a management-based policy. For more information about policies, see the *Cisco Application Control Engine Module Administration Guide*.

By default, the UDP probe sends a UDP packet to a server and marks the server as failed if the server returns an ICMP Host or Port Unreachable message. If the ACE does not receive any ICMP errors for the UDP request that was sent, the server is marked as passed. Optionally, you can configure this probe to send specific data and expect a specific response to mark the server as passed.

If the real server is not directly connected to the ACE (for example, it is connected via a gateway) and the IP interface of the server is down or disconnected, the UDP probe by itself would not know that the UDP application is not reachable. If the real server is directly connected to the ACE and the IP interface of the server is down, then the UDP probe fails.

You can create a UDP probe and access its configuration mode by using the **probe udp** *name* command.

For example, to define a UDP probe named PROBE2 and access its mode, enter:

```
host1/Admin(config)# probe udp PROBE2
host1/Admin(config-probe-udp)#
```

You can configure the following attributes for a UDP probe:

- To configure what the ACE expects as a response from the probe destination server, use the **expect regex** command. For more details about this command, see the "Configuring an Expected Response String from the Server" section.
- To configure the data sent on the connection for a UDP probe, use the **send-data** *expression* command. For more details about this command, see the "Configuring Data that the Probe Sends to the Server Upon Connection" section.

You can also configure the attributes described in the "Configuring General Probe Attributes" section.

Configuring an Echo Probe

The Echo probe sends a specified string to the server and compares the response with the original string. You must configure the string that needs to be echoed. If the response string matches the original string, the server is marked as passed. If you do not configure a string, the probe behaves like a TCP or UDP probe (see the "Configuring a TCP Probe" section or the "Configuring a UDP Probe" section).

You can create an Echo probe and access its configuration mode by using the **probe echo** command. The syntax of this command is as follows:

probe echo {**tcp** | **udp**} *name*

The keywords and argument are as follows:

- *name*—Identifier of the probe. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- **tcp**—Configures the probe for a TCP connection.
- **udp**—Configures the probe for a UDP connection.

For example, to define a TCP Echo probe named PROBE and access its mode, enter:

```
host1/Admin(config)# probe echo tcp PROBE
host1/Admin(config-probe-echo-tcp)#
```

For example, to define a UDP Echo probe named PROBE17 and access its mode, enter:

```
host1/Admin(config)# probe echo udp PROBE17
host1/Admin(config-probe-echo-udp)#
```

For Echo TCP and Echo UDP probes, you can configure the attributes described in the "Configuring General Probe Attributes" section.

For an Echo TCP probe (configured using the **tcp** keyword), you can also configure the attributes described in the "Configuring a TCP Probe" section.

For an Echo UDP probe (configured using the **udp** keyword), you can also configure the attributes described in the "Configuring a UDP Probe" section.

Configuring a Finger Probe

The Finger probe uses a Finger query to a server for an expected response string. The ACE searches the response for the configured string. If the ACE finds the expected response string, the server is marked as passed. If you do not configure an expected response string, the ACE ignores the server response.

You can create a Finger probe and access its configuration mode by using the **probe finger** command. The syntax of this command is as follows:

probe finger *name*

The *name* argument is the identifier of the probe. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a Finger probe named PROBE8 and access its mode, enter:

```
host1/Admin(config)# probe finger PROBE8
host1/Admin(config-probe-finger)#
```

To configure the attributes for a Finger probe, see the "Configuring General Probe Attributes" and "Configuring a TCP Probe" sections.

Configuring an HTTP Probe

An HTTP probe establishes a TCP connection and issues an HTTP request to the server for an expected string and status code. The ACE can compare the received response with configured codes, looking for a configured string in the received HTTP page, or verifying hash for the HTTP page. If any of these checks fail, the server is marked as failed.

For example, if you configure an expected string and status code and the ACE finds them both in the server response, the server is marked as passed. However, if the ACE does not receive either the server response string or the expected status code, it marks the server as failed.

Note If you do not configure an expected status code, any response from the server is marked as failed. However, if you configure the **expect regex** command without configuring a status code, the probe will pass if the regular expression response string is present.

Note The server response must include the Content-Length header for the **expect regex** or **hash** command to function. Otherwise, the probe does not attempt to parse the regex or hash value.

To create an HTTP probe and access its configuration mode, use the **probe http** *name* command. For example, to define an HTTP probe named PROBE4 and access its mode, enter:

```
host1/Admin(config)# probe http PROBE4
host1/Admin(config-probe-http)#
```

To configure attributes for an HTTP probe, see the following topics:

- Configuring the Credentials for a Probe
- Configuring the Header Field for the HTTP Probe
- Configuring the HTTP Method for the Probe
- Configuring the Status Code from the Destination Server
- Configuring an MD5 Hash Value

After you create an HTTP probe, you can configure the general probe attributes described in the "Configuring General Probe Attributes" section. You can also configure the TCP probe attributes, including an expected response string, described in the "Configuring a TCP Probe" section.

Configuring the Credentials for a Probe

The credentials for a probe are the username and password used for authentication on the server. You can configure the credentials for the probe by using the **credentials** command. The syntax of this command is as follows:

credentials *username* [*password*]

The arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—(Optional) The password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-http)# credentials ENG1 TEST
```

To delete the credentials for the probe, use the **no credentials** command. For example, enter:

```
host1/Admin(config-probe-http)# no credentials
```

Configuring the Header Field for the HTTP Probe

You can configure an HTTP header or multiple header fields for the HTTP probe by using the **header** command. The syntax of this command is as follows:

header *field_name* **header-value** *value*

The keywords and arguments are as follows:

- *field_name*—Identifier for a standard header field. Enter a text string with no spaces and a maximum of 64 alphanumeric characters. If the header field includes spaces, enclose its string with quotes. You can also enter one of the following header keywords:
  - **Accept**—Accept request header
  - **Accept-Charset**—Accept-Charset request header
  - **Accept-Encoding**—Accept-Encoding request header
  - **Accept-Language**—Accept-Language request header
  - **Authorization**—Authorization request header
  - **Cache-Control**—Cache-Control general header
  - **Connection**—Connection general header
  - **Content-MD5**—Content-MD5 entity header
  - **Expect**—Expect request header
  - **From**—From request header
  - **Host**—Host request header
  - **If-Match**—If-Match request header
  - **Pragma**—Pragma general header
  - **Referer**—Referer request header

  - **Transfer-Encoding**—Transfer-Encoding general header
  - **User-Agent**—User-Agent request header
  - **Via**—Via general header

- **header-value** *field_value*—Specifies the value assigned to the header field. Enter a text string with a maximum of 255 alphanumeric characters. If the value string includes spaces, enclose the string with quotes.

For example, to configure the Accept-Encoding HTTP header with a field value of IDENTITY, enter:

```
host1/Admin(config-probe-http)# header Accept-Encoding header-value
IDENTITY
```

To remove the header configuration for the probe, use the **no** form of the **header** command. For example, to remove a header with the Accept-Encoding field name, enter:

```
host1/Admin(config-probe-http)# no header Accept-Encoding
```

Configuring the HTTP Method for the Probe

By default, the HTTP request method is a GET with the URL "/". If you do not configure a URL, the probe functions as a TCP probe.

You can configure the HTTP method and URL used by the probe by using the **request method** command. The syntax of this command is as follows:

request method {**get** | **head**} **url** *path*

The keywords and arguments are as follows:

- **get** | **head**—Configures the HTTP request method. The keywords are as follows:
  - **get** —The HTTP GET request method directs the server to get the page. This method is the default.
  - **head** —The HTTP HEAD request method directs the server to get only the header for the page.
- **url** *path*—Specifies the URL path. The *path* argument is a character string of up to 255 alphanumeric characters. The default path is "/".

For example, to configure the HEAD HTTP method and the /digital/media/graphics.html URL used by an HTTP probe, enter:

```
host1/Admin(config-probe-http)# request method head url
/digital/media/graphics.html
```

To reset the HTTP method for the probe to GET, use the **no request method** command. For example, enter:

```
host1/Admin(config-probe-http)# no request method head url
/digital/media/graphics.html
```

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed. However, if you configure the **expect regex** command without configuring a status code, the probe will pass if the regular expression response string is present.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

expect status *min_number max_number*

The arguments and options are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure an expected status code of 200 that indicates that the HTTP request was successful, enter:

```
host1/Admin(config-probe-http)# expect status 200 200
```

To configure a range of expected status codes from 200 to 210, enter:

```
host1/Admin(config-probe-http)# expect status 200 210
```

To configure multiple ranges of expected status codes from 200 to 202 and 204 to 205, you must configure each range separately. For example, enter:

```
host1/Admin(config-probe-http)# expect status 200 202
host1/Admin(config-probe-http)# expect status 204 205
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 200, enter:

```
host1/Admin(config-probe-http)# no expect status 200 200
```

To remove a range of expected status codes, enter the range when using the **no expect status** command. For example, to remove a range of 200 to 202 from a range of 200 to 210, enter:

```
host1/Admin(config-probe-http)# no expect status 200 202
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (200 to 202 and 204 to 205), enter:

```
host1/Admin(config-probe-http)# no expect status 200 202
host1/Admin(config-probe-http)# no expect status 204 205
```

Configuring an MD5 Hash Value

By default, no MD5 hash value is configured on the ACE. To configure the ACE to dynamically generate the hash value or manually configure the value, use the **hash** command. If you do not use this command to configure the hash value, the ACE does not calculate a hash value on the HTTP data returned by the probe. The syntax of this command is as follows:

hash [*value*]

When you enter this command with no argument, the ACE generates the hash on the HTTP data returned by the first successful probe. If subsequent HTTP server hash responses match the generated hash value, the ACE marks the server as passed.

If a mismatch occurs due to changes to the HTTP data, the probe fails and the **show probe ... detail** command displays an MD5 mismatch error in the Last disconnect error field. To clear the reference hash and have the ACE recalculate the hash value at the next successful probe, change the URL or method by using the **request method** command. For more information, see the "Configuring the HTTP Method for the Probe" section.

The optional *value* argument is the MD5 hash value that you want to manually configure. Enter the MD5 hash value as a hexadecimal string with exactly 32 characters (16 bytes).

Note The server response must include the Content-Length header for the **hash** command to function. Otherwise, the probe does not attempt to parse the hash value.

You can configure the **hash** command on a probe using the HEAD method, however there is no data to hash and has no effect causing the probe to always succeed.

For example, to configure the ACE to generate the hash on the HTTP data returned by the first successful probe, enter:

```
host1/Admin(config-probe-http)# hash
```

To manually configure a hash value, enter:

```
host1/Admin(config-probe-http)# hash 0123456789abcdef0123456789abcdef
```

To configure the ACE to no longer compare the referenced hash value to the computed hash value, enter:

```
host1/Admin(config-probe-http)# no hash
```

To configure the ACE to no linger use a manually configured has value, enter:

```
host1/Admin(config-probe-http)# no hash 
0123456789abcdef0123456789abcdef
```

Configuring an HTTPS Probe

An HTTPS probe is similar to an HTTP probe except that it uses SSL to generate encrypted data. HTTPS probes are hardware assisted, which causes the ACE to send them from the data plane instead of the control plane. This feature causes the ACE to use the routing table (which may bypass the real server IP address) to direct HTTPS probes to their destination regardless of whether you specify the **routed** option or not in the **ip address** command. For more information about the **ip address** command, see the "Configuring the Destination IP Address" section. Also, ACLs may impact HTTPS probes if you apply them incorrectly. For more information about ACLs, see the *Cisco Application Control Engine Module Security Configuration Guide*.

Note The server response must include the Content-Length header for the **expect regex** or **hash** command to function. Otherwise, the probe does not attempt to parse the regex or hash value.

You can create an HTTPS probe and access its configuration mode by using the **probe https** command. The syntax of this command is as follows:

probe https *name*

For the *name* argument, enter an identifier for the HTTPS probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define an HTTPS probe named PROBE5 and access its mode, enter:

```
host1/Admin(config)# probe https PROBE5
host1/Admin(config-probe-https)#
```

To configure attributes for an HTTPS probe, see the following topics:

- Configuring the Cipher Suite for the HTTPS Probe
- Configuring the Supported SSL or TLS Version

After you create an HTTPS probe, you can configure the general probe attributes described in the "Configuring General Probe Attributes" section. You can also configure the HTTP probe attributes described in the "Configuring an HTTP Probe" section.

Configuring the Cipher Suite for the HTTPS Probe

By default, the HTTPS probe accepts any of the RSA configured cipher suites. You can configure the probe to expect a specific type of RSA cipher suite from the back-end server by using the **ssl cipher** command. The syntax of this command is as follows:

ssl cipher RSA_ANY | *cipher_suite*

The keyword and argument are as follows:

- **RSA_ANY**—Specifies that any of the RSA cipher suites from those allowed on the ACE is accepted from the server. This is the default setting.
- *cipher_suite*—RSA cipher suite that the probe expects from the back-end server. Enter one of the following keywords:
  - **RSA_EXPORT1024_WITH_DES_CBC_SHA**
  - **RSA_EXPORT1024_WITH_RC4_56_MD5**
  - **RSA_EXPORT1024_WITH_RC4_56_SHA**
  - **RSA_EXPORT_WITH_DES40_CBC_SHA**
  - **RSA_EXPORT_WITH_RC4_40_MD5**
  - **RSA_WITH_3DES_EDE_CBC_SHA**
  - **RSA_WITH_AES_128_CBC_SHA**
  - **RSA_WITH_AES_256_CBC_SHA**
  - **RSA_WITH_DES_CBC_SHA**
  - **RSA_WITH_RC4_128_MD5**
  - **RSA_WITH_RC4_128_SHA**

For example, to configure the HTTPS probes with the RSA_WITH_RC4_128_SHA cipher suite, enter:

```
host1/Admin(config-probe-https)# ssl cipher RSA_WITH_RC4_128_SHA
```

To reset the default behavior of the HTTPs probes accepting any RSA cipher suite, enter:

```
host1/Admin(config-probe-https)# no ssl cipher
```

Configuring the Supported SSL or TLS Version

The version in the ClientHello message sent to the server indicates the highest supported version. By default, the probe supports **all** as the SSL version. You can configure the version of SSL that the probe supports by using the **ssl version** command in probe HTTPS configuration mode.

Note For hardware-assisted SSL (HTTPS) probes, the ACE uses the **all** option for the default SSL version and uses the routing table (which may bypass the real server IP address) to direct HTTPS probes to their destination regardless of whether you specify the **routed** option in the **ip address** command.

Additionally, hardware-assisted probes are subject to the same key-pair size limitations as SSL termination. The maximum size of a public key in a server SSL certificate that the ACE can process is 2048 bits.

The syntax of this command is as follows:

ssl version {**all** | **SSLv3** | **TLSv1**}

The keywords are as follows:

- **all**—(Default) Specifies all SSL versions.
- **SSLv3**—Specifies SSL version 3.
- **TLSv1**—Specifies TLS version 1.

For example, to configure all SSL versions, enter:

```
host1/Admin(config-probe-https)# ssl version all
```

To reset the default setting, enter:

```
host1/Admin(config-probe-https)# no ssl version
```

Configuring an FTP Probe

An FTP probe establishes a TCP connection to the server. After the ACE receives the service ready message from the server, the ACE performs one of the following actions:

- Issues a **quit** command if the probe is configured for a graceful close.
- Resets the connection for forceful terminations

You can create an FTP probe and access its configuration mode by using the **probe ftp** command. The syntax of this command is as follows:

probe ftp *name*

For the *name* argument, enter the identifier of the FTP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define an FTP probe named PROBE8 and access its mode, enter:

```
host1/Admin(config)# probe ftp PROBE8
host1/Admin(config-probe-ftp)#
```

The "Configuring the Status Code from the Destination Server" section describes how to configure status codes for the probe.

You can also configure the attributes described in the "Configuring General Probe Attributes" and "Configuring a TCP Probe" sections.

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

expect status *min_number max_number*

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure an expected status code of 200 that indicates that the request was successful, enter:

```
host1/Admin(config-probe-ftp)# expect status 200 200
```

To configure a range of expected status codes from 200 to 201, enter:

```
host1/Admin(config-probe-ftp)# expect status 200 201
```

To configure multiple ranges of expected status codes from 200 to 201 and 230 to 250, you must configure each range separately. For example, enter:

```
host1/Admin(config-probe-ftp)# expect status 200 201
host1/Admin(config-probe-ftp)# expect status 230 250
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 200, enter:

```
host1/Admin(config-probe-ftp)# no expect status 200 200
```

To remove a range of expected status codes, enter the range when using the **no expect status** command. For example, to remove a range of 200 to 201, enter:

```
host1/Admin(config-probe-ftp)# no expect status 200 201
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (200 to 201 and 230 to 250), enter:

```
host1/Admin(config-probe-ftp)# no expect status 200 201
host1/Admin(config-probe-ftp)# no expect status 230 250
```

Configuring a Telnet Probe

A Telnet probe establishes a connection to the server and verifies that a greeting from the application was received. You can create a Telnet probe and access its configuration mode by using the **probe telnet** command. The syntax of this command is as follows:

probe telnet *name*

For the *name* argument, enter an identifier for the Telnet probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a Telnet probe named PROBE6 and access its mode, enter:

```
host1/Admin(config)# probe telnet PROBE6
host1/Admin(config-probe-telnet)#
```

You can also configure the attributes described in the "Configuring General Probe Attributes" and "Configuring a TCP Probe" sections.

Configuring a DNS Probe

A DNS probe sends a request to a DNS server giving it a configured domain (by default, the domain is www.cisco.com). To determine if the server is up, the ACE must receive one of the configured IP addresses for that domain. You can create a DNS probe and access its configuration mode by using the **probe dns** command. The syntax of this command is as follows:

probe dns *name*

For the *name* argument, enter an identifier for the DNS probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a DNS probe named PROBE7 and access its mode, enter:

```
host1/Admin(config)# probe dns PROBE7
host1/Admin(config-probe-dns)#
```

To configure attributes for a DNS probe, see the following topics:

- Configuring the Domain Name
- Configuring the Expected IP Address

You can also configure the attributes described in the "Configuring General Probe Attributes" section.

Configuring the Domain Name

The DNS probe sends a domain name for the DNS server to resolve. By default, the probe uses the www.cisco.com domain. You can configure the domain name that the probe sends to the server by using the **domain** command. The syntax of this command is as follows:

domain *name*

The *name* argument is the domain that the probe sends to the DNS server. Enter an unquoted text string with a maximum of 255 alphanumeric characters.

For example, to configure the domain name of support.cisco.com, enter:

```
host1/Admin(config-probe-dns)# domain support.cisco.com
```

To reset the domain to www.cisco.com, use the **no domain** command. For example, enter:

```
host1/Admin(config-probe-dns)# no domain
```

Configuring the Expected IP Address

When a DNS probe sends a domain name resolve request to the server, it verifies the returned IP address by matching the received IP address with the configured addresses. You can configure the IP address that the ACE expects as a server response to a DNS request by using the **expect address** command. The syntax of this command is as follows:

expect address *ip_address*

The *ip_address* argument is the expected returned IP address. Enter a unique IPv4 address in dotted-decimal notation (for example, 192.8.12.15).

You can specify multiple IP addresses with this command by entering the command with a different address separately. For example, to configure an expected IP address of 192.8.12.15 and 192.8.12.23, enter:

```
host1/Admin(config-probe-dns)# expect address 192.8.12.15
host1/Admin(config-probe-dns)# expect address 192.8.12.23
```

To remove an IP address, use the **no expect address** command. For example, enter:

```
host1/Admin(config-probe-dns)# no expect address 192.8.12.15
```

Configuring an SMTP Probe

SMTP probes initiates an SMTP session by logging into the server, sends a HELLO message, and then disconnects from the server. You can create an SMTP probe and access its configuration mode by using the **probe smtp** command. The syntax of this command is as follows:

probe smtp *name*

For the *name* argument, enter the identifier of the SMTP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a SMTP probe named PROBE10 and access its mode, enter:

```
host1/Admin(config)# probe smtp PROBE10
host1/Admin(config-probe-smtp)#
```

After you create an SMTP probe, you can configure the status codes as described in the "Configuring the Status Code from the Destination Server" section.

You can also configure the attributes described in the "Configuring General Probe Attributes" section, and configure connection termination as described in the "Configuring the Termination of the TCP Connection" section.

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

expect status *min_number max_number*

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure a single expected status code of 211, enter:

```
host1/Admin(config-probe-smtp)# expect status 211 211
```

To configure a range of expected status codes from 211 to 250, enter:

```
host1/Admin(config-probe-smtp)# expect status 211 250
```

To configure multiple ranges of expected status codes from 211 and 250 and 252 to 254, you must configure each range separately as follows:

```
host1/Admin(config-probe-smtp)# expect status 211 250
host1/Admin(config-probe-smtp)# expect status 252 254
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 211, enter:

```
host1/Admin(config-probe-smtp)# no expect status 211 211
```

To remove a range of expected status codes, enter the range when using the **no expect status** command. For example, to remove a range of 211 to 250, enter:

```
host1/Admin(config-probe-smtp)# no expect status 211 250
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (211 and 250 and 252 to 254), enter:

```
host1/Admin(config-probe-smtp)# no expect status 211 250
host1/Admin(config-probe-smtp)# no expect status 252 254
```

Configuring an IMAP Probe

An Internet Message Access Protocol (IMAP) probe makes a server connection and sends user credential (login, password, and mailbox) information. The ACE can send a configured command. Based on the server response, the ACE marks the probe as passed or failed.

You can create an IMAP probe and access its configuration mode by using the **probe imap** command. The syntax of this command is as follows:

probe imap *name*

For the *name* argument, enter the identifier of the IMAP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define an IMAP probe named PROBE11 and access its mode, enter:

```
host1/Admin(config)# probe imap PROBE11
host1/Admin(config-probe-imap)#
```

You can configure attributes for an IMAP probe, as described in the following topics:

- Configuring the Username Credentials
- Configuring the Mailbox
- Configuring the Request Command for the Probe

You can also configure the general attributes described in the "Configuring General Probe Attributes" section and configure connection termination as described in the "Configuring the Termination of the TCP Connection" section.

Configuring the Username Credentials

The credentials for an IMAP probe are the username and password used for authentication on the server. You can configure the credentials for the probe by using the **credentials** *username* command. The syntax of this command is as follows:

credentials *username password*

The arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—Password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-imap)# credentials ENG1 TEST
```

To delete the username credentials for the probe, use the **no credentials** *username* command. For example, to delete the username ENG1, enter:

```
host1/Admin(config-probe-imap)# no credentials ENG1
```

Configuring the Mailbox

You can configure the name of the mailbox from which the probe retrieves e-mail by using the **credentials mailbox** command. The syntax of this command is as follows:

credentials mailbox *name*

Note You must configure the credentials for an IMAP probe using the **credentials** command before you configure the mailbox or the ACE will ignore the specified user mailbox name. See the "Configuring the Username Credentials" section.

The **mailbox** *name* keyword and argument specify the user mailbox name from which to retrieve e-mail for an IMAP probe. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the user mailbox LETTERS, enter:

```
host1/Admin(config-probe-imap)# credentials mailbox LETTERS
```

To delete the mailbox for the probe, use the **no credentials mailbox** command. For example, enter:

```
host1/Admin(config-probe-imap)# no credentials mailbox
```

Configuring the Request Command for the Probe

You can configure the request command used by an IMAP probe by using the **request command** command. The syntax of this command is as follows:

request command *command*

Note You must configure the name of the mailbox using the **credentials mailbox** command before you configure the request command used by an IMAP probe or the ACE will ignore the specified request command. See the "Configuring the Mailbox" section.

The *command* argument is the request command for the probe. Enter a text string with a maximum of 32 alphanumeric characters with no spaces.

For example, to configure the last request command for an IMAP probe, enter:

```
host1/Admin(config-probe-imap)# request command last
```

To remove the request command for the probe, use the **no request** command. For example, enter:

```
host1/Admin(config-probe-imap)# no request
```

Configuring a POP3 Probe

You can configure Post Office Protocol 3 (POP3) probes to initiate a session and send the configured credentials. The ACE can also send a configured command. Based on the server response, the ACE marks the probe as passed or failed.

You can create a POP probe and access its configuration mode by using the **probe pop** command. The syntax of this command is as follows:

probe pop *name*

For the *name* argument, enter an identifier for the POP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a POP probe named PROBE12 and access its mode, enter:

```
host1/Admin(config)# probe pop PROBE12
host1/Admin(config-probe-pop)#
```

To configure attributes for a POP probe, see the following topics:

- Configuring the Credentials for a Probe
- Configuring the Request Command for the Probe

You can also configure the general attributes described in the "Configuring General Probe Attributes" section and configure connection termination as described in the "Configuring the Termination of the TCP Connection" section.

Configuring the Credentials for a Probe

The credentials for a probe are the username and password used for authentication on the server. You can configure the credentials for the probe by using the **credentials** command. The syntax of this command is as follows:

credentials *username* [*password*]

The arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—(Optional) Password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-pop)# credentials ENG1 TEST
```

To delete the credentials for the probe, use the **no credentials** command. For example, enter:

```
host1/Admin(config-probe-pop)# no credentials
```

Configuring the Request Command for the Probe

You can configure the request method used by a POP probe by using the **request command** command. The syntax of this command is as follows:

request command *command*

The *command* argument is the request method command for the probe. Enter a text string with a maximum of 32 alphanumeric characters with no spaces.

For example, to configure the last request command for a POP probe, enter:

```
host1/Admin(config-probe-pop)# request method last
```

To remove the request command for the probe, use the **no request** command. For example, enter:

```
host1/Admin(config-probe-pop)# no request
```

Configuring a SIP Probe

You can use a SIP probe to establish a TCP or UDP connection and send an OPTIONS request packet to the user agent on the server. The ACE can compare the response with the configured response code or expected string, or both, to determine the probe has succeeded.

For example, if you configure an expected string and status code and the ACE finds them both in the response, the server is marked as passed. However, if the ACE does not receive either the server response string or the expected status code, it marks the server as failed.

Note If you do not configure an expected status code, any response from the server is marked as failed.

You can create a SIP probe and access its configuration mode by using the **probe sip** {**tcp** | **udp**} *name* command. The syntax of this command is as follows:

probe sip {**tcp** | **udp**} *name*

The keywords and argument are as follows:

- **tcp**—Creates the probe for a TCP connection.
- **udp**—Creates the probe for a UDP connection.
- *name*—Identifier of the probe. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to define a SIP probe using TCP named probe13 and access its mode, enter:

```
host1/Admin(config)# probe sip tcp probe13
host1/Admin(config-probe-sip-tcp)#
```

To define a SIP probe using UDP named probe14 and access its mode, enter:

```
host1/Admin# probe sip udp probe14
host1/Admin(config-probe-sip-udp)#
```

You can configure most general probe attributes described in the "Configuring General Probe Attributes" section. If the probe uses a:

- TCP connection, as configured through the **tcp** keyword, you can configure the TCP attributes in the "Configuring a TCP Probe" section.
- UDP connection, as configured through the **udp** keyword, you can configure the UDP attributes in the "Configuring a UDP Probe" section.

Note The send data option of UDP probes is not applicable to SIP UDP probes.

You can also use the additional commands to configure attributes for a SIP probe. The following sections describes how to configure additional probe attributes:

- Configuring the Request Method for the Probe
- Configuring the Status Code from the Destination Server

Configuring the Request Method for the Probe

By default, the SIP request method is the OPTIONS method. Currently, this is the only method available for SIP probes. You can configure the OPTIONS request method that is used by the probe by using the **request method options** command. The syntax of this command is as follows:

request method options

For example, to configure the OPTIONS method, enter:

```
host1/Admin(config-probe-sip-tcp)# request method options
```

To reset the method for the probe to OPTIONS, use the **no request method** command. For example, enter:

```
host1/Admin(config-probe-sip-tcp)# no request method
```

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, there are no status codes configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

expect status *min_number max_number*

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For SIP, the expected status code is 200, which indicates a successful probe. For example, to configure an expected status code of 200 that indicates that the request was successful, enter:

```
host1/Admin(config-probe-sip-tcp)# expect status 200 200
```

Configuring an RTSP Probe

You can configure an RTSP probe to establish a TCP connection and send a request packet to the server. The ACE compares the response with the configured response code to determine whether the probe has succeeded. When configuring these probes, you use the **probe rtsp** *name* command to create the probe and access probe configuration mode.

For example, to define an RTSP probe named probe15 and access its mode, enter:

```
host1/Admin(config)# probe rtsp probe15
host1/Admin(config-probe-rtsp)#
```

After you create an RTSP probe, you can configure the general probe attributes described in the "Configuring General Probe Attributes" section. You can also configure the ACE to terminate a TCP connection by sending a RST and an expected response string as described in the "Configuring a TCP Probe" section.

You can also use the additional commands to configure attributes for an RTSP probe. The following topics describe how to configure additional probe attributes:

- Configuring the Request Method
- Configuring the Header Field for the RTSP Probe
- Configuring the Status Code from the Destination Server

Configuring the Request Method

By default, the RTSP request method is the OPTIONS method. You can also configure the DESCRIBE method. You can configure the request method that is used by the probe by using the **request method** command. The syntax of this command is as follows:

request method {**options** | **describe url** *url_string*}

The keywords and arguments are as follows:

- **options**—Configures the OPTIONS request method. This is the default method. The ACE uses the asterisk (*) request URL for this method.
- **describe url** *url_string*—Configures the DESCRIBE request method. The *url_string* is the URL request for the RTSP media stream on the server. Enter a URL string with a maximum of 255 characters.

For example, to configure an RTSP probe to use the URL for rtsp://media/video.smi, enter:

```
host1/Admin(config-probe-rtsp)# request method describe url
rtsp://192.168.10.1/media/video.smi
```

For example, to configure an RTSP probe to use the PATH for rtsp://media/video.smi, enter:

```
host1/Admin(config-probe-rtsp)# request method describe path
/media/video.smi
```

In the example shown above, the IP address is taken from the probe target IP address.

To reset the default OPTIONS request method, use the **no request method** or the **request method options** command. For example, enter:

```
host1/Admin(config-probe-rtsp)# no request method
```

Configuring the Header Field for the RTSP Probe

You can configure a header field value for the probe by using the **header** command. The syntax of this command is as follows:

header {**require** | **proxy-require**} **header-value** *value*

The keywords and arguments are as follows:

- **require**—Specifies the Require header.
- **proxy-require**—Specifies the Proxy-Require header.
- **header-value** *value*—Specifies the header value. For the value, enter an alphanumeric string with no spaces and a maximum of 255 characters.

For example, to configure the REQUIRE header with a field value of implicit-play, enter:

```
host1/Admin(config-probe-rtsp)# header require header-value
implicit-play
```

To remove the header configuration for the probe, use the **no** form of the **header** command. For example, to remove a Require header, enter:

```
host1/Admin(config-probe-rtsp)# no header require
```

To remove a Proxy-Require header, enter:

```
host1/Admin(config-probe-rtsp)# no header proxy-require
```

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

expect status *min_number max_number*

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure an expected status code of 200 that indicates that the request was successful, enter:

```
host1/Admin(config-probe-rtsp)# expect status 200 200
```

To configure a range of expected status codes from 100 to 200, enter:

```
host1/Admin(config-probe-rtsp)# expect status 100 200
```

To configure multiple ranges of expected status codes from 100 to 200 and from 250 to 305, you must configure each range separately. For example, enter:

```
host1/Admin(config-probe-rtsp)# expect status 100 200
host1/Admin(config-probe-rtsp)# expect status 250 305
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 200, enter:

```
host1/Admin(config-probe-rtsp)# no expect status 200 200
```

To remove a range of expected status codes, enter the range using the **no expect status** command. For example, to remove a range of 250 to 302 from a range of 250 to 305, enter:

```
host1/Admin(config-probe-rtsp)# no expect status 250 305
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (100 to 200 and 250 to 305), enter:

```
host1/Admin(config-probe-rtsp)# no expect status 100 200
host1/Admin(config-probe-rtsp)# no expect status 250 305
```

Configuring a RADIUS Probe

A RADIUS probe sends a query using a configured username, password, and shared secret to a RADIUS server. If the server is up, it is marked as passed. If you configure a Network Access Server (NAS) address, the ACE uses it in the outgoing packet. Otherwise, the ACE uses the IP address associated with the outgoing interface as the NAS address.

You can create the RADIUS probe and access its configuration mode by using the **probe radius** command. The syntax of this command is as follows:

probe radius *name*

For the *name* argument, enter an identifier of the RADIUS probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a RADIUS probe named PROBE and access its mode, enter:

```
host1/Admin(config)# probe radius PROBE
host1/Admin(config-probe-radius)#
```

To configure probe attributes for a RADIUS probe, see the following topics:

- Configuring the Credentials and Shared Secret for a Probe
- Configuring the Network Access Server IP Address

You can also configure the general attributes described in the "Configuring General Probe Attributes" section.

Configuring the Credentials and Shared Secret for a Probe

The credentials for a probe are the username and password used for authentication on the server and an optional shared secret to allow probe access to the RADIUS server. You can configure the credentials for the probe by using the **credentials** command. The syntax of this command is as follows:

credentials *username password* [**secret** *shared_secret*]

The keywords and arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—Password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- **secret** *shared_secret*—(Optional) Specifies the shared secret. Enter the shared secret as a case-sensitive string with no spaces and a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-radius)# credentials ENG1 TEST
```

To delete the credentials for the probe, use the **no credentials** command. For example, enter:

```
host1/Admin(config-probe-radius)# no credentials
```

Configuring the Network Access Server IP Address

If a Network Access Server (NAS) address is not configured for the RADIUS probe, the ACE uses the IP address associated with the outgoing interface as the NAS address. You can configure an NAS address by using the **nas ip address** command. The syntax of this command is as follows:

nas ip address *ip_address*

The *ip_address* argument is the NAS IP address. Enter a unique IPv4 address in dotted-decimal notation (for example, 192.8.12.15).

For example, to configure a NAS address of 192.8.12.15, enter:

```
host1/Admin(config-probe-radius)# nas ip address 192.8.12.15
```

To remove the NAS IP address, use the **no nas ip address** command. For example, enter:

```
host1/Admin(config-probe-radius)# no nas ip address
```

Configuring an SNMP-Based Server Load Probe

An SNMP-based server load probe establishes a UDP connection and allows you to configure a maximum of eight SMNP OID queries to probe the server. The ACE weighs and averages the load information that is retrieved and uses it as input to the least-loaded algorithm for load-balancing decisions. If the retrieved value is within the configured threshold, the server is marked as passed. If the threshold is exceeded, the server is marked as failed.

When configuring these probes, you use the **probe snmp** *name* command to create the probe and access probe configuration mode.

For example, to define an SNMP probe named probe18 and access its mode, enter:

```
host1/Admin(config)# probe snmp probe18
host1/Admin(config-probe-snmp)#
```

You can configure the general attributes described in the "Configuring General Probe Attributes" section. You can also use the additional commands to configure attributes for an SNMP probe. The following topics describe how to configure additional probe attributes:

- Configuring the Community String
- Configuring the SNMP Version
- Configuring the OID String
- Configuring the OID Value Type
- Configuring the OID Threshold
- Configuring the OID Weight

Configuring the Community String

The ACE probes access the server through its community string. By default, the community string is not set. You can configure the community string by using the **community** command. The syntax of the command is as follows:

community *text*

The *text* argument is the name of the SNMP community string for the server. Enter a text string with a maximum of 255 alphanumeric characters.

For example, to configure the private community string, enter:

```
host1/Admin(config-probe-snmp)# community private
```

To remove the community string, enter:

```
host1/Admin(config-probe-snmp)# no community
```

Configuring the SNMP Version

The version in the SNMP OID query sent to the server indicates the supported SNMP version. By default, the probe supports SNMP version 1.

You can configure the version of SNMP that the probe supports by using the **version** command. The syntax of this command is as follows:

version {**1** | **2c**}

The keywords are as follows:

- **1**—Specifies that the probe supports SNMP version 1 (default).
- **2c**—Specifies that the probe supports SNMP version 2c.

For example, to configure SNMP version 2c, enter:

```
host1/Admin(config-probe-snmp)# version 2c
```

To reset the default setting of SNMP version 1, enter:

```
host1/Admin(config-probe-snmp)# no version
```

Configuring the OID String

When the ACE sends a probe with an SNMP OID query, the ACE uses the retrieved values as input to the least-loaded algorithm for load-balancing decisions. Least-loaded load balancing bases the server selection on the server with the lowest load value. You can configure a maximum of eight OIDs.

To configure the OID string and access probe SNMP OID configuration mode, use the **oid** command in probe SNMP configuration mode. The syntax of the command is as follows:

oid *string*

The *string* argument is the OID that the probe uses to query the server for a value. Enter an unquoted string with a maximum of 255 alphanumeric characters in dotted-decimal notation. The OID string is based on the server type. The dots (.) in the string count as characters. For example, if the OID string is 10.0.0.1.1, then the character count is 10.

Accessing probe-snmp-oid configuration mode allows you to configure the threshold, the OID value type, and the weight assigned to the OID, as described in the following sections.

Note If you configure more than one OID and they are used in a load-balancing decision, you must configure a weight value.

For example, to configure the OID string .1.3.6.1.4.1.2021.10.1.3.1 for a 1-minute average of CPU load on a Linux server and access probe-snmp-oid configuration mode, enter:

```
host1/Admin(config-probe-snmp)# oid .1.3.6.1.4.1.2021.10.1.3.1
host1/Admin(config-probe-snmp-oid)#
```

To remove the OID string, enter:

```
host1/Admin(config-probe-snmp)# no oid .1.3.6.1.4.1.2021.10.1.3.1
```

Configuring the OID Value Type

By default, the retrieved OID value type is a percentile value. To configure the OID value type as absolute and define its maximum expected value, use the **type absolute max** command in probe SNMP OID configuration mode. The syntax of this command is as follows:

type absolute max *integer*

The *integer* argument specifies the maximum expected absolute value for the OID. Enter an integer from 1 to 4294967295. By default, the OID value is a percentile value.

Note When you configure the **type absolute max** command, we recommend that you also configure the value for the **threshold** command because the default threshold value is set to the integer value specified in the **type absolute max** command.

For example, to configure an absolute value type with its maximum expected value of 65535, enter:

```
host1/Admin(config-probe-snmp-oid)# type absolute max 65535
```

To reset the OID value type to a percentile value, enter:

```
host1/Admin(config-probe-snmp)# no type
```

Note The **no type** command resets the OID type to percentile and sets the **threshold** command to a value of 100.

Configuring the OID Threshold

The OID threshold specifies the value to take the server out of service.

- When the OID value is based on a percentile, the default threshold value is 100.
- When the OID is based on an absolute value, the threshold range is based on what you specified in the **type absolute max** command (see the "Configuring the OID Threshold" section).

To configure the threshold, use the **threshold** command in probe SNMP OID configuration mode. The syntax of this command is as follows:

threshold *integer*

The *integer* argument specifies the threshold value to take the server out of service.

- When the OID value is based on a percentile, enter an integer from 1 to 100, with a default value of 100.
- When the OID is based on an absolute value, the threshold range is from 1 to the maximum value that you specified in the **type absolute max** command.

For example, to configure a threshold of 50, enter:

```
host1/Admin(config-probe-snmp-oid)# threshold 50
```

To reset the OID threshold to its default value, enter:

```
host1/Admin(config-probe-snmp)# no threshold
```

Configuring the OID Weight

You must specify an OID weight when you configure more than one OID and they need to be used in a load-balancing decision. To configure the weight for the OID, use the **weight** command in probe-snmp-oid configuration mode. The syntax of this command is as follows:

weight *integer*

The *integer* argument specifies the weight for the OID. Enter an integer from 1 to 16000. By default, an equal weight is given to each configured OID.

Note If you configure more than one OID and they are used in a load-balancing decision, you must configure a weight value.

For example, to configure the weight of 10000, enter:

```
host1/Admin(config-probe-snmp-oid)# weight 10000
```

To reset the default behavior an equal weight given to each configured OID, enter:

```
host1/Admin(config-probe-snmp)# no weight
```

Configuring a Scripted Probe

Scripted probes allow you to run a script to execute the probe that you created for health monitoring. You can author specific scripts with features not present in standard health probes. To configure a scripted probe, you need to do the following:

- Copy the script file to the ACE disk0: file system
- Load the script file
- Associate the script with the scripted probe

The ACE allows the configuration of 256 unique script files.

You can also use the Cisco-supplied scripts located in the probe: directory in the ACE. For more information about these scripts, see the "Scripts Overview" section in Appendix A, "Using TCL Scripts with the ACE".

Note The ACE can simultaneously execute only 200 scripted probe instances. When this limit is exceeded, the **show probe detail** command displays the "Out-of Resource: Max. script-instance limit reached" error message in the Last disconnect err field and the out-of-sockets counter increments.

For information about copying and loading a script file on the ACE, see Appendix A, "Using TCL Scripts with the ACE".

You can create a scripted probe and access the scripted probe configuration mode by using the **probe scripted** command. The syntax of this command is as follows:

probe scripted *name*

For the *name* argument, enter the identifier of the scripted probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a scripted probe named PROBE19 and access its mode, enter:

```
host1/Admin(config)# probe scripted PROBE19
host1/Admin(config-probe-scrptd)#
```

To configure the scripted probe attributes, see the "Associating a Script with a Probe" section.

You can also configure the general commands described in the "Configuring General Probe Attributes" section.

Associating a Script with a Probe

Scripted probes run probes from a configured script to perform health probing. You can also configure arguments that are passed to the script. Before you can associate a script file with a probe, you must copy and load the script on the ACE. For information about copying and loading a script, see Appendix A, "Using TCL Scripts with the ACE".

Use the script command to specify the name of the script file and the arguments to be passed to the script.

The syntax of this command is as follows:

script *script_name* [*script_arguments*]

The arguments are as follows:

- *script_name*—Name of the script. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters.
- *script_arguments*—(Optional) Data sent to the script. Enter a text string with a maximum of 255 alphanumeric characters including spaces and quotes. Separate each argument by a space. If a single argument contains spaces, enclose the argument string in quotes.

For example, to configure the script name of PROBE-SCRIPT and arguments of ??, enter:

```
host1/Admin(config-probe-scrptd)# script PROBE-SCRIPT ??
```

To remove the script and its arguments from the configuration, use the **no script** command. For example, enter:

```
host1/Admin(config-probe-scrptd)# no script
```

Example of a UDP Probe Load-Balancing Configuration

The following example shows a running configuration that load balances DNS traffic across multiple real servers, and transmits and receives UDP data that spans multiple packets. The configuration uses a UDP health probe. The UDP probe configuration appears in bold in the example.

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

probe udp UDP
  interval 5
  passdetect interval 10
  description THIS PROBE IS INTENDED FOR LOAD BALANCING DNS TRAFFIC
  port 53
  send-data UDP_TEST

rserver host SERVER1
  ip address 192.168.252.245
  inservice
rserver host SERVER2
  ip address 192.168.252.246
  inservice
rserver host SERVER3
  ip address 192.168.252.247
  inservice

serverfarm host SFARM1
  probe UDP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice

class-map match-all L4UDP-VIP_114:UDP_CLASS
   2 match virtual-address 192.168.120.114 udp eq 53
policy-map type loadbalance first-match L7PLBSF_UDP_POLICY
  class class-default
    serverfarm SFARM1
policy-map multi-match L4SH-Gold-VIPs_POLICY
    class L4UDP-VIP_114:UDP_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_UDP_POLICY
    loadbalance vip icmp-reply
    nat dynamic 1 vlan 120
    connection advanced-options 1SECOND-IDLE
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 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```

Configuring KAL-AP

A keepalive-appliance protocol (KAL-AP) on the ACE allows communication between the ACE and the Global Site Selector (GSS), which send KAL-AP requests, to report the server states and loads for global-server load-balancing (GSLB) decisions. The ACE uses KAL-AP through a UDP connection to calculate weights and provide information for server availability to the KAL-AP device. The ACE acts as a server and listens for KAL-AP requests. When KAL-AP is initialized on the ACE, the ACE listens on the standard 5002 port for any KAL-AP requests. You cannot configure any other port.

The ACE supports VIP-based and tag-based KAL-AP probes. For a VIP-based KAL-AP, when the ACE receives a kal-ap-by-vip request, it verifies whether the VIP addresses are active in all Layer 3 class maps that are configured with the addresses. The ACE ignores all other protocol-specific information for the VIP addresses. For each Layer 3 class map, the ACE locates the associated Layer 7 policies and associated real servers in server farms. The ACE determines the total number of servers associated with these VIPs and those servers in the Operational state.

The ACE calculates a load number from 0 to 255 and reports the server availability of the VIP to the KAL-AP device. A load value of 0 indicates that the VIP address is not available. This value is also sent in the case of any VIP lookup failures. A load value of 1 is reserved to indicate that the VIP is offline and not available for use. Valid load values are form 2 to 255. A load value of 2 indicates that the VIP is least loaded and a load value of 255 indicates that the VIP is fully loaded. For example, if the total number of servers is 10 and only 5 are operational, the load value is 127.

Note If the same real server is associated with more than one server farm, the ACE includes it twice in the calculation.

The ACE supports VIP-based and tag-based KAL-AP probes. For tag-based KAL-AP, the tag corresponds to:

- A VIP address in a policy configuration. The ACE supports a maximum of 4,096 VIP tags.
- A domain associated with a VIP address. Through a domain, you can associate multiple VIP addresses to the tag. The ACE supports a maximum of 64 KAL-AP domain tags per context.

When the ACE receives a kal-ap-by-tag request, the process is similar to VIP-based KAL-AP probes. The load calculation considers all Layer 3 class map, server farm, and real server objects of the domain or VIP. Note that other objects configured in a domain are ignored during the load calculation. For more information on domain objects, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*. The ACE gathers the server availability information for any Layer 3 VIP address. The ACE considers all of the associated server farms. If real servers are configured, the ACE adds them to the current total and then performs a division to determine their availability as tag objects. The ACE reports this final number in the KAL-AP response.

This section contains the following topics:

- Enabling KAL-AP on the ACE
- Configuring a KAL-AP VIP Address
- Configuring KAP-AP Tags per VIP Address
- Configuring KAL-AP Tags as Domains
- Configuring Secure KAL-AP
- Displaying Global-Server Load-Balancing Load Information
- Displaying Global-Server Load-Balancing Statistics

Enabling KAL-AP on the ACE

To enable KAL-AP on the ACE, you must configure a management class map and policy map, and apply it to the appropriate interface. The KAL-AP server listens on the standard 5002 port to all KAL-AP requests.

You can configure the class map for KAL-AP over UDP management access by using the **match protocol kalap-udp** command in the class map management configuration mode. The syntax of this command is as follows:

match protocol kalap-udp any | [**source-address** *ip_address subnet_mask*]

The keywords and arguments are as follows:

- **any**—Specifies any client source address for the management traffic classification.
- **source-address**—Specifies a client source host IP address and subnet mask as the network traffic matching criteria. As part of the classification, the ACE implicitly obtains the destination IP address from the interface on which you apply the policy map.
- *ip_address*—Source IP address of the client. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).
- *mask*—Subnet mask of the client entry in dotted-decimal notation (for example, 255.255.255.0).

For example, to specify a KAL-AP class map from any source IP address, enter:

```
host1/Admin(config)# class-map type management KALAP-CM
host1/Admin(config-cmap-mgmt)# match protocol kalap-udp any
host1/Admin(config-cmap-mgmt)# exit
host1/Admin(config)#
```

To remove the class map, enter:

```
host1/Admin(config-cmap-mgmt)# no match protocol kalap-udp
source-address any
```

After you create the KAL-AP class map, create a KAL-AP management policy map and apply the class map to it. To create the policy map and access policy map management configuration mode, use the **policy-map type management** command in configuration mode. For example, to create the KALAP-MGMT management policy map and apply the KALAP-CM class map to it, enter:

```
host1/Admin(config)# policy-map type management KALAP-MGMT
host1/Admin(config-pmap-mgmt)# class KALAP-CM
```

```
host1/Admin(config-cmap-mgmt)# permit
host1/Admin(config-cmap-mgmt)# exit
host1/Admin(config)#
```

To apply the policy map to an interface, use the **interface vlan** command in configuration mode. For example, to apply the KALAP-MGMT policy map to VLAN interface 10, enter:

```
host1/Admin(config)# interface vlan 10
host1/Admin(config-if)# ip address 10.1.0.1 255.255.255.0
host1/Admin(config-if)# service-policy input KALAP-MGMT
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
host1/Admin(config)#
```

Note When you modify or remove a KAL-AP policy, you must clear the existing KAL-AP connections manually.

Configuring a KAL-AP VIP Address

You can configure VIP-based KAL-AP by configuring a Layer 3/4 class map that contains a VIP address match statement. You can define a 3-tuple flow of VIP address, protocol, and port as matching criteria by using the **match virtual-address** command in class map configuration mode. You can configure multiple match criteria statements to define the VIPs for server load balancing. The syntax of this command is as follows:

> [*line_number*] **match virtual-address** *vip_address* {[*mask*] | **any** | {**tcp** | **udp** {**any** | **eq** *port_number* | **range** *port1 port2*}} | *protocol_number*}

For information on the keywords and arguments, see the "Defining VIP Address Match Criteria" section in Chapter 3, Configuring Traffic Policies for Server Load Balancing. To configure a KAL-AP tag associated with the VIP address in the match statement, see the "Configuring KAP-AP Tags per VIP Address" section.

Note For KAL-AP, the ACE verifies whether the VIP addresses are active in all Layer 3 class maps that are configured with the addresses. It ignores all other protocol-specific information for the VIP addresses.

For example, to create a class map VIP-20 that matches traffic destined to VIP address 10.10.10.10 with a wildcard value for the IP protocol value (TCP or UDP), enter:

```
host1/Admin(config)# class-map VIP-20
host1/Admin(config-cmap)# match virtual-address 10.10.10.10 any
```

To remove the VIP match statement from the class map, enter:

```
host1/Admin(config-cmap)# no match virtual-address 10.10.10.10 any
```

Configuring KAP-AP Tags per VIP Address

A keepalive-appliance protocol (KAL-AP) on the ACE allows communication between the ACE and the Global Site Selector (GSS), which sends KAL-AP requests, to report the server states and loads for global-server load-balancing (GSLB) decisions. The ACE uses KAL-AP through a UDP connection to calculate weights and provide information for server availability to the KAL-AP device. The ACE acts as a server and listens for KAL-AP requests. When KAL-AP is initialized on the ACE, the ACE listens on the standard 5002 port for any KAL-AP requests. You cannot configure any other port.

The ACE supports VIP-based and tag-based KAL-AP probes. Previous to the A2(2.0) release, the ACE supported only tag-based KAL-AP for domains associated with VIP addresses. Through the domain, you could associate multiple VIP addresses with a tag with a maximum of 64 KAL-AP domain tags per context.

The KAL-AP tags per VIP address feature allows you to associate a KAL-AP tag with a VIP address in a policy map configuration. You can configure multiple VIP addresses to a tag or a VIP address to multiple tags. The ACE supports 4,096 VIP tags.

For information on configuring a VIP KAL-AP tag and displaying its load information, see the following sections:

- Configuring the VIP Address Match Statement
- Associating a KAL-AP Tag to a VIP Class Map
- Displaying the Load Information for a VIP KAL-AP Tag

Note For the domain load calculation, the ACE module considers the Layer 3 class map, server farm, and real server objects. All other objects under the domain are ignored during the calculation. For the ACE module A2(2.0) release, the calculation of the Layer 3 class-map has changed. Previously, the calculation considered each VIP address that is configured in the class map. A VIP-based KAL-AP calculation is run on each address. Now, the calculation consider all Layer 3 rules (a Layer 3 class map within a Layer 3 policy map) defined by the class map and sums up the total number of servers and the number of servers in the Up state. After determining these sums, the ACE module multiplies them by the number of VIP addresses configured in the class map.

Configuring the VIP Address Match Statement

Before you configure the VIP KAL-AP tag, configure a Layer 3 class map that contains a VIP address match statement. You can define a 3-tuple flow of VIP address, protocol, and port as matching criteria by using the **match virtual-address** command in class map configuration mode. You can configure multiple match criteria statements to define the VIP for server load balancing. The syntax of this command is as follows:

[*line_number*] **match virtual-address** *vip_address* {[*mask*] | **any** | {**tcp** | **udp** {**any** | **eq** *port_number* | **range** *port1 port2*}} | *protocol_number*}

For detailed information on the keywords and arguments for this command, see the "Defining VIP Address Match Criteria" section on page 3-78.

Note For KAL-AP, the ACE verifies whether the VIP addresses are active in all Layer 3 class maps that are configured with the addresses. It ignores all other protocol-specific information for the VIP addresses.

For example, to create a class map VIP-20 that matches traffic destined to VIP address 10.10.10.10 with a wildcard value for the IP protocol value (TCP or UDP), enter the following command:

```
host1/Admin(config)# class-map VIP-20
host1/Admin(config-cmap)# match virtual-address 10.10.10.10 any
```

Associating a KAL-AP Tag to a VIP Class Map

After you configure a Layer 3 class map that contains a KAL-AP VIP address match statement, you can associate a KAL-AP tag with the address in the class map by using the **kal-ap-tag** command in policy map class configuration mode. The syntax for this command is as follows:

kal-ap-tag *tag_name*

The *tag_name* is the name of the KAL-AP tag. Enter the name as an unquoted text string with no spaces and a maximum of 76 alphanumeric characters.

Note the following restrictions:

- You cannot associate the same tag name to more than one Layer 3 class map.
- You cannot associate the same tag name to a domain and a Layer 3 class map.
- You cannot configure a tag name for a Layer 3 class map that already has a tag configuration as part of a different Layer 3 policy map configuration, even if it is the same tag name.

For example, to associate the VIP-20 class map with the l3_policy20 policy map by using the **class** command in policy map configuration mode and access policy class configuration mode, enter the following command:

```
host1/Admin(config)# policy-map multi-match l3_policy20
host1/Admin(config-pmap)# class VIP-20
host1/Admin(config-pmap-c)#
```

To associate the KAL-AP-TAG2 tag with the class map, enter the following command:

```
host1/Admin(config-pmap-c)# kal-ap-tag KAL-AP-TAG2
```

To remove the KAL-AP-TAG2 tag from the class map, enter the following command:

```
host1/Admin(config-pmap-c)# no kal-ap-tag
```

Displaying the Load Information for a VIP KAL-AP Tag

To display the latest load information for a VIP tag name provided to the KAL-AP request, use the **show kalap udp load** command in Exec mode. The syntax of the command to display VIP tag information is as follows:

show kalap udp load {**all** | **vip tag** *name*}

The keywords and arguments are as follows:

- **all**—Displays the latest load information for all VIP addresses, VIP tags, and domains configured on the ACE module.
- **vip tag** *name*—Displays the latest load information for the specified VIP tag name.

Table 2 lists the field and output descriptions for the **show kalap udp load all** command.

Table 2 Field Output Descriptions for the show kalap udp load all Command

Field	Description
VIP-Addr	VIP address of the KAL-AP request based on a VIP address.
VIP Tag Name	Tag name for a KAL-AP request based on a VIP tag and its associated VIP address.
Domain Name	Name of the domain for a KAL-AP request.
VIP	VIP address for the VIP tag or domain KAL-AP request.
Port	Port number for the KAL-AP request.
Load Value	Load number that the ACE calculates. The number is from 0 to 255 and reports the server availability of the VIP to the KAL-AP device. A load value of 0 indicates that the VIP address is not available. A load value of 2 indicates that the VIP is least loaded and a load value of 255 indicates that the VIP is fully loaded. A load value of 1 is reserved to indicate that the VIP is offline and not available for use.
Time Last Updated	Time when the KAL-AP request occurred.

For example, to display the latest load information for all VIP addresses, domains, and VIP tags, enter the following command:

```
host1/Admin# show kalap udp load all
```

To display the latest load information to the KAL-AP request for the VIP KAL-AP-TAG2 tag, enter the following command:

```
host1/Admin# show kalap udp load vip tag KAL-AP-TAG2
```

Configuring KAL-AP Tags as Domains

You can configure KAL-AP tags as domains by using the **domain** command in configuration mode. You can configure a maximum of 64KAL-AP tag domains per context. The syntax of this command is as follows:

domain *name*

The *name* is the name of the KAL-AP tag with a maximum of 76 characters.

Note For the domain load calculation, the ACE considers the Layer 3 class map, server farm, and real server objects. All other objects under the domain are ignored during the calculation.

For example, to configure KAL-AP-TAG1 as a domain, enter:

```
host1/Admin(config)# domain KAL-AP-TAG1
```

After you create the domain, use the **add-object class-map** command in domain configuration mode to add each class map that you want to associate with the tag domain. For example, to add the VIP-20 and VIP-71 class maps to the tag domain, enter:

```
host1/Admin(config-domain)# add-object class-map VIP-20
host1/Admin(config-domain)# add-object class-map VIP-71
```

To remove the domain, enter:

```
host1/Admin(config)# no domain KAL-AP-TAG1
```

For more information about configuring class maps, see Chapter 3, "Configuring Traffic Policies for Server Load Balancing.". For more information about configuring domains, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

Configuring Secure KAL-AP

The ACE supports secure KAL-AP for MD5 encryption of data between it and the GSS. For encryption, you must configure a shared secret as a key for authentication between the GSS and the ACE context.

To configure secure KAL-AP on the ACE, access KAL-AP UDP configuration mode through the **kalap udp** command in configuration mode. The syntax of this command is as follows:

kalap udp

For example, enter:

```
host1/Admin(config)# kalap udp
host1/Admin(config-kalap-udp)#
```

To remove the KAL-AP configuration and all VIP entries, enter the following command:

```
host1/Admin(config)# no kalap udp
```

In this mode, you enable secure KAL-AP by configuring the VIP address to the GSS and the shared secret through the **ip address** command. The syntax of this command is as follows:

ip address *ip_address* **encryption md5** *secret*

The keywords and arguments are as follows:

- *ip_address*—The VIP address for the GSS. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).
- **encryption**—Specifies the encryption method.
- **md5**—Specifies the MD5 encryption method.
- *secret*—Shared secret between the KAL-AP device and the ACE. Enter the shared secret as a case-sensitive string with no spaces and a maximum of 31 alphanumeric characters.

For example, to enable secure KAL-AP and configure the VIP address for the GSS and shared secret, enter:

```
host1/Admin(config-kalap-udp)# ip address 10.1.0.1 encryption md5
andromeda
```

To disable secure KAL-AP, use the **no** form of the **ip address** command. For example, enter:

```
host1/Admin(config-kalap-udp)# no ip address 10.1.0.1
```

Displaying Global-Server Load-Balancing Load Information

You can display the latest load information for a VIP address, domain name, or VIP tag name provided to the KAL-AP request by using the **show kalap udp load** command in Exec mode. The syntax of the command is as follows:

show kalap udp load {**all** | **domain** *name* | **vip** {*ip_address* | **tag** *name*}}

The keywords and arguments are as follows:

- **all**—Displays the latest load information for all VIP addresses, VIP-based tags, and domains.
- **domain** *name*—Displays the latest load information for the specified domain name.
- **vip** *ip_address* | **tag** *name*—Displays the latest load information for the specified VIP address or existing VIP tag name. For the *ip_address* argument, enter the IP address in dotted-decimal notation (for example, 192.168.11.1).

The output fields for the **show kalap udp load** command display the VIP address, VIP tag with its associated VIP address and port, or domain name with its associated VIP address and port, its load value, and the time stamp.

For example, to display the latest load information for all VIP addresses, VIP-based tags, and domains, enter:

```
host1/Admin# show kalap udp load all
```

For example, to display the latest load information to the KAL-AP request for VIP address 10.10.10.10, enter:

```
host1/Admin# show kalap udp load vip 10.10.10.10
```

To display the latest load information to the KAL-AP request for domain KAL-AP-TAG1, enter:

```
host1/Admin# show kalap udp load domain KAL-AP-TAG1
```

To display the latest load information to the KAL-AP request for the VIP KAL-AP-TAG2 tag, enter:

```
host1/Admin# show kalap udp load vip tag KAL-AP-TAG2
```

Displaying Global-Server Load-Balancing Statistics

You can display the global-server load-balancing statistics per context by using the **show stats kalap** command in Exec mode. The syntax of the command is as follows:

show stats kalap [**all**]

The optional **all** keyword in the admin context displays the total number of KAL-AP statistics for all contexts. These statistics are followed by the statistics for the admin context and then all other contexts.

For example, to display the statistics in a context, enter:

```
host1/Admin# show stats kalap
```

To display the all statistics for all contexts in the admin context, enter:

```
host1/Admin# show stats kalap all
```

Table 4-3 lists the output fields displayed by this command.

Table 4-3 Field Descriptions for the show stats kalap Command

Field	Description
Total bytes received	Total number of bytes received.
Total bytes sent	Total number of bytes sent.
Total requests received	Total number of requests received.
Total responses sent	Total number of responses sent.
Total requests successfully received	Total number of requests successfully received.

Table 4-3 Field Descriptions for the show stats kalap Command

Field	Description
Total queries successfully received	Number of queries that the ACE module received from the GSS. A request from the GSS may contain between 1 to 60 queries.
Total responses successfully sent	Total number of responses successfully sent.
Total secure requests received	Total number of secure requests received.
Total secure responses sent	Total number of secure responses sent.
Total requests with errors	Total number of requests with errors.
Total requests with parse errors	Total number of requests with parse errors.
Total requests dropped due to queue overflow	Number of requests that the ACE module drops when the KAL-AP request queue is full. The ACE has a maximum KAL-AP request queue size of 1024 requests.
Total response transfer errors	Total number of response transfer errors.

You can clear the global-server load-balancing statistics per context by using the **clear stats kalap** command in Exec mode. For example, enter:

```
host1/Admin# clear stats kalap
```

Displaying Probe Information

You can display configuration information and statistics for a probe by using the **show probe** command in Exec mode. The syntax of this command is as follows:

show probe [*probe_name*] [**detail**]

The argument and option are as follows:

- *probe_name*—(Optional) Information for the specified probe name.
- **detail**—(Optional) Displays detailed probe configuration and statistic information.

If you do not enter a probe name, this command shows a summary of information for all configured probes. For example, enter:

```
host1/Admin# show probe
```

You can also display configuration information for all probes by using the **show running-config probe** command.

For example, enter:

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

Table 4-4 describes the fields in the **show probe** command output including additional output provided by the **detail** option.

Table 4-4 Field Descriptions for the show probe Command

Field	Description
Probe	Name of the probe.
Type	Probe type.
State	Whether the probe is active or inactive.
Description	Configured description for the probe (**detail** option output).
Port	Port number that the probe uses. By default, the probe uses the port number based on its type.
Address	Destination address for the probe.
Addr type	Address type.
Interval	Time interval in seconds that the ACE sends probes to a server marked as passed.
Pass intvl	Time period in seconds to send a probe to a failed server.
Pass count	Consecutive number of passed probes before marking the server as passed.

Table 4-4 Field Descriptions for the show probe Command (continued)

Field	Description
Fail count	Consecutive number of failed probes before marking the server as failed.
Recv timeout	Time period in seconds to receive a server response to the probe.
DNS domain	Domain name configured for the probe (**detail** option output for a DNS probe).
HTTP method	HTTP method and URL used by the probe, GET or HEAD (**detail** option output for HTTP and HTTPS probes).
HTTP URL	URL used by the probe with the HTTP method (**detail** option output for HTTP and HTTPS probes).
RTSP method	RTSP method and URL used by the probe (**detail** option output for RTSP probes).
RTSP URL	URL used by the probe with the RTSP method (**detail** option output for RTSP probes).
IMAP mailbox	Mailbox username where the probe retrieves e-mail (**detail** option output for IMAP probes).
IMAP/POP command	Request method command for the probe (**detail** option output for IMAP and POP probes).
NAS address	Network Access Server (NAS) address for the RADIUS server (**detail** option output for RADIUS probes).
Script filename	Filename for the script (**detail** option output for scripted probes).
Conn termination	TCP connection termination type, GRACEFUL or FORCED (**detail** option output for ECHO TCP, Finger, FTP, HTTP, HTTPS, IMAP, POP, SMTP, TCP, and Telnet probes).
Expect/Search offset	Number of characters into the received message or buffer to start searching for the expect regex expression (**detail** option output for HTTP, HTTPS, RTSP, SIP, TCP, and UDP probes).

Table 4-4 Field Descriptions for the show probe Command (continued)

Field	Description
Request-method	Request method for SIP probes displayed in the **detail** option output. Currently, the OPTIONS method is the only method available for SIP probes.
Expect regex	Configured expected response data from the probe destination (**detail** option output for HTTP, HTTPS, RTSP, SIP, TCP, and UDP probes).
Open timeout	Time interval in seconds that the probe waits to open and establish the connection with the server (**detail** option output for Finger, FTP, HTTP, HTTPS, IMAP, POP, scripted, RTSP, SMTP, TCP, and Telnet probe).
Send data	ASCII data that the probe sends (**detail** option output for ECHO, Finger, HTTP, HTTPS, RTSP, TCP, and UDP probes).
Version	SNMP version in the SNMP OID query sent to the server that indicates the supported version (**detail** option output for SNMP probes).
Community	SNMP community string (**detail** option output for SNMP probes).
OID string	The configured OID (**detail** option output for SNMP probes).
Type	The OID value type, absolute or percentile, for the retrieved OID value (**detail** option output for SNMP probes).
Max value	The maximum expected load value for the OID load type (**detail** option output for SNMP probes).
Weight	The load weight for the OID (**detail** option output for SNMP probes).
Threshold	The threshold setting for the OID. When the threshold is exceeded, the OID is taken out of service (**detail** option output for SNMP probes).
probe results	
probe association	Real server association for the probe.

Table 4-4 Field Descriptions for the show probe Command (continued)

Field	Description
ip-address	Destination or source address for the probe.
probes	Total number of probes.
failed	Total number of failed probes.
passed	Total number of passed probes.
health	Health of the probe. Possible values are PASSED or FAILED.
Additional **detail** option output for scripted probes:	
Socket state	Socket state.
No. Passed states	Number of passed states.
No. Failed states	Number of failed states.
No. Probes skipped	Number of skipped probes. A skipped probe occurs when the ACE does not send out a probe because the scheduled interval to send a probe is shorter than it takes to complete the execution of the probe; the send interval is shorter than the open timeout or receive timeout interval. When a probe is skipped or an internal error is displayed by the **show probe detail** command, the state of the probe does not change. If it fails, it remains as failed.
Last status code	Last exit code (see Table A-7).
Last disconnect err	Message for the exit code for a scripted probe (see Table A-7) or an internal error.
Last probe time	Time stamp for the last probe.
Last fail time	Time stamp for the last failed probe.
Last active time	Time stamp for the last active time.
Internal error	Counter for the number of internal errors encountered.

Table 4-5 list the possible disconnect errors that can appear in the **show probe** output. For a list of disconnect messages for scripted probes, see Table A-7.

Table 4-5 ACE Probe Disconnect Errors

Probe Type	Error Message
All probe types	Unrecognized or invalid probe request.
	Connect error.
	Connection reset by server.
	Connection refused by server.
	Authentication failed.
	Unrecognized or invalid response.
	Out of memory, packets discarded.
	Server open timeout (no SYN ACK).
	Server reply timeout (no reply).
	Graceful disconnect timeout (no FIN ACK).
	Received Out-Of-Band data.
	User defined Reg-Exp was not found in host response.
	Expect status code mismatch.
	Received invalid status code.

Table 4-5 ACE Probe Disconnect Errors (continued)

Probe Type	Error Message
ICMP	ICMP Internal error.
	ICMP Internal error: Write failure.
	ICMP Internal error: Received bad FD.
	Host Unreachable, no route found to destination.
	ARP not resolved for dest-ip (destination IP address).
	Network down.
	Egress interface has no ip addr (IP address).
	ICMP Internal error: Data entry being modified.
	ICMP Internal error: No space, transmit path is full.
	ICMP Host unreachable.
	ICMP Dest unreachable.
	ICMP Time exceeded.
	ICMP Redirect.
	Received ICMP Echo Request.
	Received ICMP Stale pkt.
	Unexpected ICMP pkt type received.
	ICMP Pkt received is too short.
	ICMP Pkt received is too long.
HTTP/HTTPS	MD5 mismatch.
HTTPS	Invalid server greeting.
	Internal error: Failed to build a server query.

Table 4-5 ACE Probe Disconnect Errors (continued)

Probe Type	Error Message
SNMP	Last Disconnect Error: Sum of weights don't add up to max weight value.
	Last Disconnect Error: ASN encoding failed for the configured SNMP OID.
	Last Disconnect Error: Server load hit max value for type percentile.
	Last Disconnect Error: Server load hit max value for type absolute.
	Last Disconnect Error: Server load hit the threshold value.
	Last Disconnect Error: Failed to parse the PDU reply sent by the server.
	Last Disconnect Error: Unrecognized or invalid response.

To display the global statistics for a probe type, use the **show stats probe type** command in Exec mode. The syntax of this command is as follows:

show stats probe type *probe_type*

To view a list of probe types, enter:

```
host1/Admin# show stats probe type ?
```

For example, to view the global statistics for all DNS probes, enter:

```
host1/Admin# show stats probe type dns
```

Table 4-6 describes the fields in the **show stats probe type** command output.

Table 4-6 Field Descriptions for the show stats probe type command

Field	Description
Total probes sent	Total number of probes sent.
Total send failures	Total number of send failures. These failures are due to internal errors.
Total probes passed	Total number of passed probes.
Total probes failed	Total number of failed probes.
Total connect errors	Total number of connection errors.
Total conns refused	Total number of connections refused.
Total RST received	Total number of resets received.
Total open timeouts	Total number of open timeouts for the specified probe type.
Total receive timeouts	Total number of timeouts received.

Clearing Probe Statistics

This section describes the commands that you use to clear probe statistics, either for individual probes or for all probes in a context. It contains the following topics:

- Clearing Statistics for Individual Probes
- Clearing All Probe Statistics in a Context

Clearing Statistics for Individual Probes

You can clear the statistics displayed through the **show probe** command for a specific probe by using the **clear probe** command in Exec mode. The syntax of this command is as follows:

clear probe *name*

The *name* argument is the name of a configured probe.

For example, to clear the statistics for the DNS1 probe, enter:

```
host1/Admin# clear probe DNS1
```

Note If you have redundancy configured, then you need to explicitly clear load-balancing statistics on both the active and the standby ACEs. Clearing statistics on the active module only will leave the standby module's statistics at the old values.

Clearing All Probe Statistics in a Context

You can clear all probe statistics in the current context by using the **clear stats probe** command in Exec mode. The syntax of this command is as follows:

clear stats probe

For example, enter:

```
host1/Admin# clear stats probe
```

Note If you have redundancy configured, then you need to explicitly clear load-balancing statistics on both the active and the standby ACEs. Clearing statistics on the active module only will leave the standby module's statistics at the old values.

Where to Go Next

To learn how to use the Toolkit Command Language (TCL) to write probe scripts, see Appendix A, "Using TCL Scripts with the ACE". To configure stickiness (session persistence), see Chapter 5, Configuring Stickiness. To configure firewall load balancing (FWLB), see Chapter 6, Configuring Firewall Load Balancing.

CHAPTER 5

Configuring Stickiness

This chapter describes how to configure stickiness (sometimes referred to as session persistence) on an ACE module. It contains the following major sections:

- Stickiness Overview
- Configuration Requirements and Considerations for Configuring Stickiness
- Configuring IP Address Stickiness
- Configuring Layer 4 Payload Stickiness
- Configuring HTTP Content Stickiness
- Configuring HTTP Cookie Stickiness
- Configuring HTTP Header Stickiness
- Configuring RADIUS Attribute Stickiness
- Configuring RTSP Session Stickiness
- Configuring SIP Call-ID Stickiness
- Configuring SSL Session-ID Stickiness
- Configuring the Reverse IP Stickiness Feature
- Configuring an SLB Traffic Policy for Stickiness
- Displaying Sticky Configurations and Statistics
- Clearing Sticky Statistics
- Clearing Dynamic Sticky Database Entries
- Example of a Sticky Configuration
- Where to Go Next

Stickiness Overview

Stickiness is an ACE feature that allows the same client to maintain multiple simultaneous or subsequent TCP or IP connections with the same real server for the duration of a session. A session is defined as a series of transactions between a client and a server over some finite period of time (from several minutes to several hours). This feature is particularly useful for e-commerce applications where a client needs to maintain multiple connections with the same server while shopping online, especially while building a shopping cart and during the checkout process.

Depending on the configured SLB policy, the ACE “sticks” a client to an appropriate server after the ACE has determined which load-balancing method to use. If the ACE determines that a client is already stuck to a particular server, then the ACE sends that client request to that server, regardless of the load-balancing criteria specified by the matched policy. If the ACE determines that the client is not stuck to a particular server, it applies the normal load-balancing rules to the content request.

This section contains the following topics:

- Why Use Stickiness?
- Sticky Groups
- Sticky Methods
- Sticky Table
- Backup Server Farm Behavior with Stickiness

Why Use Stickiness?

When customers visit an e-commerce site, they usually start out by browsing the site, the Internet equivalent of window shopping. Depending on the application, the site may require that the client become “stuck” to one server as soon as the connection is established, or the application may require this action only when the client starts to build a shopping cart.

In either case, once the client adds items to the shopping cart, it is important that all of the client requests get directed to the same server so that all the items are contained in one shopping cart on one server. An instance of a customer’s shopping cart is typically local to a particular web server and is not duplicated across multiple servers.

E-commerce applications are not the only types of applications that require stickiness. Any web application that maintains client information may require stickiness, such as banking applications or online trading. Other uses include FTP and HTTP file transfers.

Sticky Groups

The ACE uses the concept of sticky groups to configure stickiness. A sticky group allows you to specify sticky attributes. After you configure a sticky group and its attributes, you associate the sticky group with a match statement or a Layer 7 policy-map action in a Layer 7 SLB policy map. You can create a maximum of 4096 sticky groups in each context. Each sticky group that you configure on the ACE contains a series of parameters that determine the following:

- Sticky method
- Timeout
- Replication
- Cookie offset and other related attributes
- HTTP, RTSP, or SIP header offset and other header-related attributes
- RADIUS attributes

Sticky Methods

Because an application must distinguish each user or group of users, the ACE needs to determine how a particular user is stuck to a specific web server. The ACE supports the following sticky methods:

- Source and/or destination IP address
- Layer 4 payload
- Hypertext Transfer Protocol (HTTP) content
- HTTP cookie
- HTTP header
- Remote Access Dial-In User Service (RADIUS) attributes
- Real-Time Streaming Protocol (RTSP) header
- Session Initiation Protocol (SIP) header
- SSL Session ID

The e-commerce application itself often dictates which of these methods is appropriate for a particular e-commerce vendor.

This section contains the following topics:

- IP Address Stickiness
- Layer 4 Payload Stickiness
- HTTP Content Stickiness
- HTTP Cookie Stickiness
- HTTP Header Stickiness
- RADIUS Attribute Stickiness
- RTSP Session Header Stickiness
- SIP Call-ID Header Stickiness
- SSL Session-ID Stickiness

IP Address Stickiness

You can use the source IP address, the destination IP address, or both to uniquely identify individual clients and their requests for stickiness purposes based on their IP netmask. However, if an enterprise or a service provider uses a megaproxy to establish client connections to the Internet, the source IP address no longer is a reliable indicator of the true source of the request. In this case, you can use cookies or one of the other sticky methods to ensure session persistence.

Layer 4 Payload Stickiness

Layer 4 payload stickiness allows you to stick a client to a server based on the data in Layer 4 frames. You can specify a beginning pattern and ending pattern, the number of bytes to parse, and an offset that specifies how many bytes to ignore from the beginning of the data.

HTTP Content Stickiness

HTTP content stickiness allows you to stick a client to a server based on the content of an HTTP packet. You can specify a beginning pattern and ending pattern, the number of bytes to parse, and an offset that specifies how many bytes to ignore from the beginning of the data.

HTTP Cookie Stickiness

Client cookies uniquely identify clients to the ACE and to the servers that provide content. A cookie is a small data structure within the HTTP header that is used by a server to deliver data to a web client and request that the client store the information. In certain applications, the client returns the information to the server to maintain the connection state or persistence between the client and the server.

When the ACE examines a request for content and determines through policy matching that the content is sticky, it examines any cookie or URL present in the content request. The ACE uses the information in the cookie or URL to direct the content request to the appropriate server. The ACE supports the following types of cookie stickiness:

- Dynamic cookie learning—You can configure the ACE to look for a specific cookie name and automatically learn its value either from the client request HTTP header or from the server Set-Cookie message in the server response. Dynamic cookie learning is useful when dealing with applications that store more than just the session ID or user ID within the same cookie. Only very specific bytes of the cookie value are relevant to stickiness.

  By default, the ACE learns the entire cookie value. You can optionally specify an offset and length to instruct the ACE to learn only a portion of the cookie value.

  Alternatively, you can specify a secondary cookie value that appears in the URL string in the HTTP request. This option instructs the ACE to search for (and eventually learn or stick to) the cookie information as part of the URL. URL learning is useful with applications that insert cookie information as part of the HTTP URL. In some cases, you can use this feature to work around clients that reject cookies.

- Cookie insert—The ACE inserts the cookie on behalf of the server upon the return request, so that the ACE can perform cookie stickiness even when the servers are not configured to set cookies. The cookie contains information that the ACE uses to ensure persistence to a specific real server.

HTTP Header Stickiness

Additionally, you can use HTTP header information to provide stickiness. With HTTP header stickiness, you can specify a header offset to provide stickiness based on a unique portion of the HTTP header.

RADIUS Attribute Stickiness

The ACE supports stickiness based on RADIUS attributes. The following attributes are supported for RADIUS sticky groups:

- Framed IP
- Framed IP and calling station ID
- Framed IP and username

RTSP Session Header Stickiness

The ACE supports stickiness based on the RTSP session header field. With RTSP header stickiness, you can specify a header offset to provide stickiness based on a unique portion of the RTSP header.

SIP Call-ID Header Stickiness

The ACE supports stickiness based on the SIP Call-ID header field. SIP header stickiness requires the entire SIP header, so you cannot specify an offset.

SSL Session-ID Stickiness

This feature allows the ACE to stick the same client to the same SSL server based on the SSL Session ID. Note that this feature supports SSLv3 only. Because the SSL Session ID is unique across multiple connections from the same client, you can use this feature to stick clients to a particular SSL server when the ACE is configured to load-balance SSL traffic, but not terminate it. To use this feature, you must configure a generic protocol-parsing policy for sticky learning. The ACE learns the SSL Session ID from the SSL server or other SSL-termination device.

Because an SSL server can reuse the same SSL Session ID for new connections from a known client, the SSL handshake time is reduced. This reduction in handshake time translates directly into lower computational requirements for the server and reduced CPU utilization, and, therefore, increased SSL transactions per second (TPS).

Sticky Table

To keep track of sticky connections, the ACE uses a sticky table. Table entries include the following items:

- Sticky groups
- Sticky methods
- Sticky connections
- Real servers

The sticky table can hold a maximum of four million entries (four million simultaneous users). When the table reaches the maximum number of entries, additional sticky connections cause the table to wrap and the first users become unstuck from their respective servers.

The ACE uses a configurable timeout mechanism to age out sticky table entries. When an entry times out, it becomes eligible for reuse. High connection rates may cause the premature aging out of sticky entries. In this case, the ACE reuses the entries that are closest to expiration first.

Sticky entries can be either dynamic or static (user configured). When you create a static sticky entry, the ACE places the entry in the sticky table immediately. Static entries remain in the sticky database until you remove them from the configuration. You can create a maximum of 4096 static sticky entries in each context.

If the ACE takes a real server out of service for whatever reason (probe failure, no inservice command, or ARP timeout), the ACE removes any sticky entries that are associated with that server from the database.

Backup Server Farm Behavior with Stickiness

When you associate a server farm with a sticky group using the **serverfarm** command in sticky configuration mode, the primary server farm inherits the stickiness of the sticky group. The ACE sends requests from the same client to the same server in the primary server farm based on the type of stickiness that you configure in the sticky group. If you also configure a backup server farm using the **backup** option of the same command, you can make the backup server farm sticky, too, by configuring the optional **sticky** keyword.

If all the servers in the primary server farm go down, the ACE sends all new requests to the backup server farm. When the primary server farm comes back up (at least one server becomes active):

- If the **sticky** option is enabled, then:
  - All new sticky connections that match existing sticky table entries for the real servers in the backup server farm are stuck to the same real servers in the backup server farm.
  - All new non-sticky connections and those sticky connections that do not have an entry in the sticky table are load balanced to the real servers in the primary server farm.
- If the **sticky** option is not enabled, then the ACE load balances all new connections to the real servers in the primary server farm.
- Existing non-sticky connections to the servers in the backup server farm are allowed to complete in the backup server farm.

Note You can fine-tune the conditions under which the primary server farm fails over and returns to service by configuring a partial server farm failover. For details about partial server farm failover, see the "Configuring a Partial Server Farm Failover" section in Chapter 2, Configuring Real Servers and Server Farms.

If you want to configure sorry servers and you want existing connections to revert to the primary server farm after it comes back up, do not use stickiness. For information about configuring backup server farms and sorry servers, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Configuration Requirements and Considerations for Configuring Stickiness

We recommend that you observe the following requirements and considerations when you configure stickiness on your ACE:

- For each context in which you configure stickiness, you must do the following:
  - Configure a resource class in the Admin context that you can associate with one or more contexts where you want to configure stickiness.

  - Allocate a minimum non-zero percentage of resources to stickiness in the resource class using the **limit-resource sticky** command.
  - Associate one or more user contexts with the resource class.

  For details about configuring resource groups and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

- You can configure the same sticky group in multiple policies or virtual servers. In that case, the sticky behavior applies to all connections to any of those policies or class maps. These connections are referred to as *buddy connections* because, if you configure both policy or class map 1 and 2 with the same sticky group, a client stuck to server A through policy or class map 1 will also be stuck to the same server A through policy or class map 2.
- If you associate the same sticky group with multiple policies, it is very important to make sure that all the policies use either the same server farm or different server farms with the same servers in them.

Note You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

Configuring IP Address Stickiness

IP address stickiness allows you to stick a client to the same server for multiple subsequent connections as needed to complete a transaction using the client source IP address, the destination IP address, or both. If the service provider or enterprise uses a megaproxy to allow clients to connect to the Internet, use cookies or one of the other sticky methods described in this chapter.

This section contains the following topics:

- IP Address Stickiness Configuration Quick Start
- Creating an IP Address Sticky Group
- Configuring a Timeout for IP Address Stickiness
- Enabling an IP Address Sticky Timeout to Override Active Connections
- Enabling the Replication of IP Address Sticky Table Entries
- Configuring Static IP Address Sticky Table Entries
- Associating a Server Farm with an IP Address Sticky Group
- Example of IP Address Sticky Configuration

IP Address Stickiness Configuration Quick Start

Table 5-1 provides a quick overview of the steps required to configure stickiness on an ACE. 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 5-1.

Table 5-1 IP Address Stickiness 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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create a sticky-IP group and enter sticky-IP configuration mode.
`host1/Admin(config)# sticky ip-netmask 255.255.255.255 address both GROUP1`
`host1/Admin(config-sticky-ip)#`

5. Configure a timeout for IP address stickiness. `host1/Admin(config-sticky-ip)# timeout 720`
6. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-ip)# timeout activeconns`
7. Enable the replication of sticky table information to the standby context in case of a switchover in a redundancy configuration. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*. `host1/Admin(config-sticky-ip)# replicate sticky`

Table 5-1 IP Address Stickiness Configuration Quick Start (continued)

Task and Command Example
8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and in the backup server farm. `host1/Admin(config-sticky-ip)# serverfarm SFARM1 backup BKUP_SFARM2 sticky`
9. (Optional) Configure static IP address sticky entries up to a maximum of 65535 static entries per context. `host1/Admin(config-sticky-ip)# static client source 192.168.12.15 destination 172.16.27.3 rserver SERVER1 2000`
10. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
11. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
12. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
13. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
14. Display your IP address sticky configuration. Make any modifications to your configuration as necessary, and then reenter the **show** command to verify your configuration changes. `host1/Admin# show running-config sticky`
15. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating an IP Address Sticky Group

Before you begin to configure an IP address sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To create a sticky group for IP address stickiness, use the **sticky ip-netmask** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is as follows:

sticky ip-netmask *netmask* **address** {**source** | **destination** | **both**} *name*

The keywords and arguments are as follows:

- **ip-netmask** *netmask*—Specifies the network mask that the ACE applies to the IP address. Enter a network mask in dotted-decimal notation (for example, 255.255.255.255).

Note If you configure a network mask other than 255.255.255.255 (/32), the ACE may populate the sticky entries only on one of its two network processors which may reduce the number of available sticky entries by 50 percent. This reduction in resources can cause problems when heavy sticky use occurs on the ACE.

- **address**—Specifies the IP address used for stickiness. Enter one of the following options after the address keyword:
  - **source**—Specifies that the ACE use the client source IP address to stick the client to a server. You typically use this keyword in web application environments.
  - **destination**—Specifies that the ACE use the destination address specified in the client request to stick the client to a server. You typically use this keyword in caching environments.
  - **both**—Specifies that the ACE use both the source IP address and the destination IP address to stick the client to a server.
- *name*—Unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a sticky group that uses IP address stickiness based on both the source IP address and the destination IP address, enter:

```
host1/Admin(config)# sticky ip-netmask 255.255.255.255 address both
GROUP1
host1/Admin(config-sticky-ip)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky ip-netmask 255.255.255.255 address both
GROUP1
```

Configuring a Timeout for IP Address Stickiness

The sticky timeout specifies the period of time that the ACE keeps (if possible) the IP address sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection or receives a new HTTP GET on an existing connection that matches that entry. High connection rates may cause the premature age-out of sticky table entries.

To configure an IP address sticky timeout, use the **timeout** *minutes* command in sticky-IP configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-ip)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-ip)# no timeout 720
```

Enabling an IP Address Sticky Timeout to Override Active Connections

By default, the ACE ages out a sticky table entry when the timeout for that entry expires and no active connections matching that entry exist. To specify that the ACE time out IP address sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky-IP configuration mode.

Note When the ACE times out a RADIUS load-balanced (RLB) sticky entry, it only uses connections for the end-user traffic towards the connection count. It does not use connections for the RADIUS traffic towards the connection count, whether or not you configure the **timeout activeconns** command. The only exception is when a connection has an outstanding RADIUS request for that sticky entry.

The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-ip)# timeout activeconns
```

To restore the behavior of the ACE to the default of not timing out IP address sticky entries if active connections exist, enter:

```
host1/Admin(config-sticky-ip)# no timeout activeconns
```

Enabling the Replication of IP Address Sticky Table Entries

If you are using redundancy, you can configure the ACE to replicate IP address sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate IP address sticky table entries on the standby ACE, use the **replicate sticky** command in sticky-IP configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of an IP address sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. Thus, the standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-ip)# replicate sticky
```

To restore the ACE default of not replicating IP address sticky table entries, enter:

```
host1/Admin(config-sticky-ip)# no replicate sticky
```

Configuring Static IP Address Sticky Table Entries

You can configure static sticky table entries based on the source IP address, destination IP address, or real server name and port. Static sticky-IP values remain constant over time and you can configure multiple static entries.

Note When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure static sticky-IP table entries, use the **static client** command in sticky-IP configuration mode. The syntax of this command varies according to the **address** option that you chose when you created the sticky group. See the "Creating an IP Address Sticky Group" section.

If you configured the sticky group with the **source** option, the syntax of this command is as follows:

static client source *ip_address* **rserver** *name* [*number*]

If you configured the sticky group with the **destination** option, the syntax of this command is as follows:

static client destination *ip_address* **rserver** *name* [*number*]

If you configured the sticky group with the **both** option, the syntax of this command is as follows:

static client source *ip_address* [**destination** *ip_address*]{**rserver** *name* [*number*]}

The keywords, arguments, and options are as follows:

- **source** *ip_address*—Specifies that the static entry be based on the source IP address. Enter an IP address in dotted decimal notation (for example, 192.168.12.15).
- **destination** *ip_address*—Specifies that the static entry be based on the destination IP address. Enter an IP address in dotted-decimal notation (for example, 172.16.27.3).
- **rserver** *name*—Specifies that the static entry be based on the real server name. Enter the name of an existing real server as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, to configure a static sticky entry based on the source IP address, the destination IP address, and the server name and port number, enter:

```
host1/Admin(config-sticky-ip)# static client source 192.168.12.15
destination 172.16.27.3 rserver SERVER1 2000
```

To remove the static entry from the sticky table, enter:

```
host1/Admin(config-sticky-ip)# no static client source 192.168.12.15
destination 172.16.27.3 rserver SERVER1 2000
```

Associating a Server Farm with an IP Address Sticky Group

To complete a sticky group configuration, you must configure a server-farm entry for the group. To configure a server-farm entry for a sticky group, use the **serverfarm** command in sticky-IP configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. If you configure the **sticky** option with the backup server farm, the clients remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.

  Note If all servers in the server farm fail and you did not configure a backup server farm, the ACE sends a reset (RST) to a client in response to a content request. If you do configure a backup server farm, by default, the ACE takes into account the state of all the real servers in the backup server farm before taking the VIP out of service. If all the real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

- **sticky**—(Optional) Specifies that the backup server farm is sticky.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-ip)# serverfarm SFARM1 backup BKUP_SFARM2
sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-ip)# no serverfarm
```

Example of IP Address Sticky Configuration

The following example illustrates a running configuration that defines IP address stickiness.The IP address stickiness configuration appears in bold in the example.

In this configuration, the ACE uses IP address stickiness to stick a client to the same server for multiple subsequent connections as needed to complete a transaction using the client source IP address, the destination IP address, or both.

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

probe icmp ICMP
  interval 2
  faildetect 2
  passdetect interval 2

rserver host SERVER1
  ip address 192.168.252.240
  inservice
rserver host SERVER2
  ip address 192.168.252.241
  inservice
rserver host SERVER3
  ip address 192.168.252.242
  inservice

serverfarm host SFARM1
  probe ICMP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice

sticky ip-netmask 255.255.255.255 address both SGROUP1
  timeout 20
  replicate sticky
  serverfarm SFARM1

class-map match-all L4STICKY-IP_115:ANY_CLASS
  2 match virtual-address 192.168.120.115 any
policy-map type loadbalance first-match L7PLBSF_STICKY-NETMASK_POLICY
  class class-default
    sticky-serverfarm SGROUP1
```

```
policy-map multi-match L4SH-Gold-VIPs_POLICY
  class L4STICKY-IP_115:ANY_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_STICKY-NETMASK_POLICY
    loadbalance vip icmp-reply
    nat dynamic 1 VLAN 120

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 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```

Configuring Layer 4 Payload Stickiness

This section describes how to configure stickiness based on a string in the data of a TCP stream or UDP packet. You can configure a class map and a policy map to match generic protocols (those that are not explicitly supported by the ACE) and then stick a client to a specific server based on a string in the data (payload) portion of the protocol packet, such as a user ID. You define the string as a regular expression (regex) and its location in the payload as an offset and length in the sticky configuration. For more information, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

To avoid using a large amount of memory with regular expressions, we recommend the following guidelines when you configure Layer 4 payload stickiness:

- Use only one generic rule per VIP.
- Use the same offset for all generic rules on the same VIP.
- Use the smallest possible offset that will work for your application.
- Avoid deploying Layer 4 payload stickiness and Layer 4 payload matching (see Chapter 3, Configuring Traffic Policies for Server Load Balancing) simultaneously, when possible.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- Layer 4 Payload Stickiness Configuration Quick Start
- Creating a Layer 4 Payload Sticky Group
- Configuring a Layer 4 Payload Sticky Timeout
- Enabling a Layer 4 Payload Timeout to Override Active Connections
- Enabling the Replication of Layer 4 Payload Sticky Entries
- Configuring Layer 4 Payload Sticky Parameters
- Configuring a Static Layer 4 Payload Sticky Entry
- Associating a Server Farm with a Layer 4 Payload Sticky Group

Layer 4 Payload Stickiness Configuration Quick Start

Table 5-4 provides a quick overview of the steps required to configure stickiness on an ACE. 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 5-4.

Table 5-2 Layer 4 Payload Stickiness 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, change to, or directly log in to, the correct context.

   ```
   host1/Admin# changeto C1
   host1/C1#
   ```

   The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create a Layer 4 payload sticky group and enter sticky Layer 4 configuration mode.

   ```
   host1/Admin(config)# sticky layer4-payload L4_PAYLOAD_GROUP
   host1/Admin(config-sticky-l4payloa)#
   ```

5. Configure a timeout for Layer 4 payload stickiness.

   ```
   host1/Admin(config-sticky-l4payloa)# timeout 720
   ```

6. (Optional) Enable the timeout to override active connections.

   ```
   host1/Admin(config-sticky-l4payloa)# timeout activeconns
   ```

7. Enable the replication of sticky table information to the standby context in case of a switchover in a redundancy configuration. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*.

   ```
   host1/Admin(config-sticky-l4payloa)# replicate sticky
   ```

Table 5-2 Layer 4 Payload Stickiness Configuration Quick Start

Task and Command Example
8. Enable sticky learning for responses from the server. `host1/Admin(config-sticky-l4payloa)# response sticky`
9. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and in the backup server farm. `host1/Admin(config-sticky-l4payloa)# serverfarm SFARM1 backup BKUP_SFARM2 sticky`
10. (Optional) Configure the Layer 4 payload offset and length to instruct the ACE to parse only a portion of the data for stickiness. `host1/Admin(config-sticky-l4payloa)# layer4-payload offset 250 length 750 begin-pattern abc123`
11. (Optional) Configure one or more static sticky payload entries. `host1/Admin(config-sticky-l4payloa)# static layer4-payload stingray rserver RS1 4000`
12. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
13. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
14. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
15. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
16. Display your Layer 4 payload sticky configuration. Make any modifications to your configuration as necessary, and then reenter the **show** command to verify your configuration changes. `host1/Admin# show running-config sticky`
17. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating a Layer 4 Payload Sticky Group

Before you begin to configure a Layer 4 payload sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To create a Layer 4 payload sticky group, use the **sticky layer4-payload** command in configuration mode. The syntax of this command is as follows:

sticky layer4-payload *name*

The *name* argument is the unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config)# sticky layer4-payload L4_PAYLOAD_GROUP
host1/Admin(config-sticky-l4payloa)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky layer4-payload L4_PAYLOAD_GROUP
```

Configuring a Layer 4 Payload Sticky Timeout

The sticky timeout specifies the period of time that the ACE keeps the Layer 4 payload sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** *minutes* command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-l4payloa)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-l4payloa)# no timeout 720
```

Enabling a Layer 4 Payload Timeout to Override Active Connections

To specify that the ACE time out Layer 4 payload sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-l4payloa)# timeout activeconns
```

To restore the behavior of the ACE to the default of not timing out Layer 4 payload sticky entries if active connections exist for those entries, enter:

```
host1/Admin(config-sticky-l4payloa)# no timeout activeconns
```

Enabling the Replication of Layer 4 Payload Sticky Entries

If you are using redundancy, you can configure the ACE to replicate Layer 4 payload sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate sticky table entries on the standby ACE, use the **replicate sticky** command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of a sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. The standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-l4payloa)# replicate sticky
```

To restore the ACE default of not replicating sticky table entries, enter:

```
host1/Admin(config-sticky-l4payloa)# no replicate sticky
```

Enabling Sticky Learning for Server Responses

With Layer 4 payload sticky, the ACE parses client requests based on the offset, length, and pattern that you specify. See the "Configuring Layer 4 Payload Sticky Parameters" section.

To enable the ACE to parse server responses and perform sticky learning, use the **response sticky** command in sticky Layer 4 payload configuration mode. The ACE uses a hash of the server response bytes to populate the sticky database. The next time that the ACE receives a client request with those same bytes, it sticks the client to the same server. The syntax of this command is as follows:

response sticky

For example, to enable the ACE to parse the response bytes from a server and perform sticky learning, enter:

```
host1/Admin(config-sticky-l4payloa)# response sticky
```

To reset the behavior of the ACE to the default of not parsing server responses and performing sticky learning, enter:

```
host1/Admin(config-sticky-l4payloa)# no response sticky
```

Configuring Layer 4 Payload Sticky Parameters

A Layer 4 payload may change over time with only a portion remaining constant throughout a transaction between the client and a server. You can configure the ACE to use the constant portion of a payload to make persistent connections to a specific server. To define the portion of the payload that you want the ACE to use, you specify payload offset and length values. The ACE stores these values in the sticky table.

You can also specify a beginning and end pattern based on a regular expression that the ACE uses to stick a client to a particular server. For information about regular expressions, see Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing.

To configure the payload offset, length, beginning pattern, and end pattern, use the **layer4-payload** command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

> **layer4-payload** [**offset** *number1*] [**length** *number2*] [**begin-pattern** *expression1*] [**end-pattern** *expression2*]

The keywords, arguments, and options are as follows:

- **offset** *number1*—Specifies the portion of the payload that the ACE uses to stick the client on a particular server by indicating the bytes to ignore starting with the first byte of the payload. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the payload.
- **length** *number2*—Specifies the length of the portion of the payload (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is the entire payload.

  For a TCP connection, the ACE stops parsing only if the **max-parse length** value is equal to or less than the portion of the packet remaining after the **offset** value. If the **max-parse length** value is larger than the remaining packet size, the ACE waits continuously to receive more data from the client.

  For UDP, the ACE stops parsing when it reaches the end of the packet.

  Note You cannot specify both the **length** and the **end-pattern** options in the same **layer4-payload** command.

- **begin-pattern** *expression1*—Specifies the beginning pattern of the Layer 4 payload and the pattern string to match before hashing. If you do not specify a beginning pattern, the ACE begins parsing immediately after the offset byte. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). The ACE supports the use of regexes for matching string expressions. Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing, lists the supported characters that you can use for matching string expressions.

Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

- **end-pattern** *expression2*—Specifies the pattern that marks the end of hashing. If you do not specify an end pattern or a length, the ACE continues to parse the data until it reaches the end of the field or packet, or until it reaches the maximum body parse length. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.)

Note You cannot specify both the **length** and the **end-pattern** options in the same **layer4-payload** command.

Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). The ACE supports the use of regexes for matching string expressions. Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing, lists the supported characters that you can use for matching string expressions.

For example, enter:

```
host1/Admin(config-sticky-l4payloa)# layer4-payload offset 250 length 
750 begin-pattern abc123
```

To remove the payload offset and length from the configuration, enter:

```
host1/Admin(config-sticky-l4payloa)# no layer4-payload
```

Configuring a Static Layer 4 Payload Sticky Entry

You can configure the ACE to use static sticky entries from entries based on Layer 4 payloads and, optionally, real server names and ports. Static payload values remain constant over time. You can configure multiple static payload entries, but only one unique real-server name can exist for a given static payload value.

Note When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure a static Layer 4 payload sticky entry, use the **static layer4-payload** command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

static layer4-payload *value* **rserver** *name* [*number*]

The keywords, arguments, and options are as follows:

- *value*—Payload string value. 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 string in quotation marks (“).
- **rserver** *name*—Specifies the hostname of an existing real server.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, enter:

```
host1/Admin(config-sticky-l4payloa)# static layer4-payload STINGRAY 
rserver SERVER1 4000
```

To remove a static payload entry from the configuration, enter:

```
host1/Admin(config-sticky-l4payloa)# no static layer4-payload STINGRAY
rserver SERVER1 4000
```

Associating a Server Farm with a Layer 4 Payload Sticky Group

To complete a sticky group configuration, you must configure a server farm entry for the group. To configure a server farm entry for a sticky group, use the **serverfarm** command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**] [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. Once clients are stuck to a backup server farm, they remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.
- **sticky**—(Optional) Specifies that the backup server farm is sticky.
- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-l4payloa)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-l4payloa)# no serverfarm
```

Configuring HTTP Content Stickiness

This section describes how to configure stickiness based on the content (data, not the header) of HTTP packets. You define a string in the HTTP content as a regular expression with a beginning and ending pattern. You further define its location in the packet data as an offset and length.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- HTTP Content Stickiness Configuration Quick Start
- Creating an HTTP Content Sticky Group
- Configuring an HTTP Content Sticky Timeout
- Enabling a Sticky Content Timeout to Override Active Connections
- Enabling the Replication of Sticky Content Entries
- Configuring HTTP Content Sticky Parameters
- Configuring Static HTTP Content
- Associating a Server Farm with an HTTP Content Sticky Group

HTTP Content Stickiness Configuration Quick Start

Table 5-3 provides a quick overview of the steps required to configure HTTP content stickiness on an ACE. 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 5-3.

Table 5-3 HTTP Content Stickiness 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, change to, or directly log in to, the correct context.

```
host1/Admin# changeto C1
host1/C1#
```

The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create an HTTP content sticky group and enter sticky-content configuration mode.

```
host1/Admin(config)# sticky http-content HTTP_CONTENT_GROUP
host1/Admin(config-sticky-content)#
```

5. Configure a timeout for HTTP content stickiness.

```
host1/Admin(config-sticky-content)# timeout 720
```

Table 5-3 HTTP Content Stickiness Configuration Quick Start (continued)

Task and Command Example
6. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-content)# timeout activeconns`
7. Enable the replication of sticky table information to the standby context in case of switchover in a redundancy configuration. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*. `host1/Admin(config-sticky-content)# replicate sticky`

8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and in the backup server farm.
`host1/Admin(config-sticky-content)# serverfarm SFARM1 backup`
`BKUP_SFARM2 sticky`

9. (Optional) Configure the sticky content beginning pattern, ending pattern, offset, and length to instruct the ACE to use only a portion of the content (that part of the content that remains constant) for stickiness.
`host1/Admin(config-sticky-content)# content begin-pattern abc123*`
`end-pattern *xyz890 offset 3000 length 1000`

10. (Optional) Configure one or more static sticky content entries.
`host1/Admin(config-sticky-content)# static content stingray`
`rserver RS1 4000`

11. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
12. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
13. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
14. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Table 5-3 HTTP Content Stickiness Configuration Quick Start (continued)

Task and Command Example
15. Display your HTTP content sticky configuration. Make any modifications to your configuration as necessary, then reenter the **show** command to verify your configuration changes. `host1/Admin# show running-config sticky`
16. Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating an HTTP Content Sticky Group

Before you begin to configure an HTTP content sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To configure the ACE to use HTTP content for stickiness, use the **sticky http-content** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is as follows:

sticky http-content *name*

The *name* argument is the unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a sticky group for content stickiness, enter:

```
host1/Admin(config)# sticky http-content HTTP_CONTENT_GROUP
host1/Admin(config-sticky-content)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky http-content HTTP_CONTENT_GROUP
```

Configuring an HTTP Content Sticky Timeout

The sticky timeout specifies the period of time that the ACE keeps the HTTP content sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** *minutes* command in sticky-content configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-content)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-content)# no timeout 720
```

Enabling a Sticky Content Timeout to Override Active Connections

To specify that the ACE time out HTTP content sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky-content configuration mode. The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-content)# timeout activeconns
```

To restore the ACE default to not time out HTTP content sticky entries if active connections exist for those entries, enter:

```
host1/Admin(config-sticky-content)# no timeout activeconns
```

Enabling the Replication of Sticky Content Entries

If you are using redundancy, you can configure the ACE to replicate HTTP content sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate sticky table entries on the standby ACE, use the **replicate sticky** command in sticky-content configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of a sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. Thus, the standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-content)# replicate sticky
```

To restore the default behavior of the ACE to not replicate sticky table entries, enter:

```
host1/Admin(config-sticky-content)# no replicate sticky
```

Configuring HTTP Content Sticky Parameters

HTTP content may change over time with only a portion remaining constant throughout a transaction between the client and a server. You can configure the ACE to use the constant portion of the content to make persistent connections to a specific server. To define the portion of the content that you want the ACE to use, you specify the beginning pattern, the ending pattern, the offset, and the length values. The ACE stores these values in the sticky table.

To configure the HTTP content sticky parameters, use the **content** command in sticky-content configuration mode. The syntax of this command is as follows:

content [**offset** *number1*] [**length** *number2*] [**begin-pattern** *expression1*] [**end-pattern** *expression2*]

The keywords, arguments, and options are as follows:

- **offset** *number1*—(Optional) Specifies the portion of the content that the ACE uses to stick the client on a particular server by indicating the bytes to ignore starting with the first byte of the payload. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the content.

  The offset and length can vary from 0 to 1000 bytes. If the content string is longer than the offset but shorter than the offset plus the length of the string, the ACE sticks the connection based on that portion of the content starting with the byte after the offset value and ending with the byte specified by the offset plus the length. The total of the offset and the length cannot exceed 1000.

- **length** *number2*—(Optional) Specifies the length of the portion of the content (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is the entire payload.

  Note You cannot specify both the **length** and the **end-pattern** options in the same **content** command.

- **begin-pattern** *expression1*—(Optional) Specifies the beginning pattern of the HTTP content payload and the pattern string to match before hashing. If you do not specify a beginning pattern, the ACE begins parsing immediately after the offset byte. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.)

  Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). The ACE supports the use of regexes for matching string expressions. See Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing, for a list of the supported characters that you can use for matching string expressions.

Note When matching data strings, note that the period (.) and the question mark (?) characters do not have a literal meaning in regular expressions. Use brackets ([]) 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 (?).

- **end-pattern** *expression2*—(Optional) Specifies the pattern that marks the end of hashing. If you do not specify an end pattern or a length, the ACE continues to parse the data until it reaches the end of the field or packet, or until it reaches the maximum body parse length. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See Chapter 3, Configuring Traffic Policies for Server Load Balancing.)

  Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“). The ACE supports the use of regexes for matching string expressions. See Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing, for a list of the supported characters that you can use for matching string expressions.

  Note You cannot specify both the **length** and the **end-pattern** options in the same **content** command.

For example, enter:

```
host1/Admin(config-sticky-content)# content begin-pattern abc123*
end-pattern *xyz890 offset 500
```

To remove the content parameters from the configuration, enter:

```
host1/Admin(config-sticky-content)# no content offset
```

Configuring Static HTTP Content

You can configure the ACE to use static content entries and, optionally, real server names and ports. Static content entries remain constant over time. You can configure multiple static content entries, but only one unique real-server name can exist for a given static content string.

Note When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure a static content entry, use the **static content** command in sticky-content configuration mode. The syntax of this command is as follows:

static content *value* **rserver** *name* [*number*]

The keywords, arguments, and options are as follows:

- *value*—Content string value. 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 string in quotation marks (“).
- **rserver** *name*—Specifies the hostname of an existing real server.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, to create a static content entry, enter:

```
host1/Admin(config-sticky-content)# static content STINGRAY rserver
SERVER1 4000
```

To remove a static content entry from the configuration, enter:

```
host1/Admin(config-sticky-content)# no static content STINGRAY rserver
SERVER1 4000
```

Associating a Server Farm with an HTTP Content Sticky Group

To complete a sticky group configuration, you must configure a server farm entry for the group. To configure a server farm entry for a sticky group, use the **serverfarm** command in sticky-content configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**] [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. Once clients are stuck to a backup server farm, they remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.
- **sticky**—(Optional) Specifies that the backup server farm is sticky.
- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-content)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-content)# no serverfarm
```

Configuring HTTP Cookie Stickiness

This section describes how to configure stickiness based on HTTP cookies. The ACE learns cookie values from the following:

- HTTP header in the client request
- Set-Cookie message sent by the server to the client
- URL for a web page

When a client makes an HTTP request to a server, the server typically sends a cookie in the Set-Cookie message in the response to the client. In most cases, the client returns the same cookie value in a subsequent HTTP request. The ACE sticks the client to the same server based on that matching value. This scenario is typical on the web with traditional web clients.

However, in some environments, clients may be unable to support cookies in their browser, which makes this type of cookie sticky connection impossible. To circumvent this problem, the ACE can extract the cookie name and value embedded in the URL string. This feature works only if the server embeds the cookie into the URL link on the web page.

Depending on client and server behavior and the sequence of frames, the same cookie value may appear in the standard HTTP cookie that is present in the HTTP header, Set-Cookie message, or cookie embedded in a URL. The actual name of the cookie may differ depending on whether the cookie is embedded in a URL or appears in an HTTP header. The use of a different name for the cookie and the URL occurs because these two parameters are configurable on the server and are often set differently. For example, the Set-Cookie name may be as follows:

```
Set-Cookie: session_cookie = 123
```

The URL may be as follows:

```
http://www.example.com/?session-id=123
```

If the client request does not contain a cookie, the ACE looks for the session-ID string (?session-id=) configured on the ACE. The value associated with this string is the session-ID number that the ACE looks for in the cache. The ACE matches the session ID with the server where the requested information resides and the ACE sends the client request to that server.

The *name* argument in the **sticky** command is the cookie name that appears in the HTTP header. The name argument in the **cookie secondary** command specifies the cookie name that appears in the URL.

By default, the maximum number of bytes that the ACE parses to check for a cookie, HTTP header, or URL is 4096. If a cookie, HTTP header, or URL exceeds the default value, the ACE drops the packet and sends a RST (reset) to the client browser. You can increase the number of bytes that the ACE parses using the **set header-maxparse-length** command in HTTP parameter-map configuration mode. For details about setting the maximum parse length, see Chapter 3, "Configuring Traffic Policies for Server Load Balancing."

You can also change the default behavior of the ACE when a cookie, header, or URL exceeds the maximum parse length using the **length-exceed** command in HTTP parameter-map configuration mode. For details, see Chapter 3, "Configuring Traffic Policies for Server Load Balancing."

Note You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- HTTP Cookie Stickiness Configuration Quick Start
- Creating an HTTP Cookie Sticky Group
- Configuring a Cookie Sticky Timeout
- Enabling a Sticky Cookie Timeout to Override Active Connections
- Enabling the Replication of Cookie Sticky Entries
- Enabling Cookie Insertion
- Configuring the Offset and Length of an HTTP Cookie
- Configuring a Secondary Cookie
- Configuring a Static Cookie

- Associating a Server Farm with an HTTP Cookie Sticky Group
- Example of HTTP Cookie Stickiness Configuration

HTTP Cookie Stickiness Configuration Quick Start

Table 5-4 provides a quick overview of the steps required to configure stickiness on an ACE. 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 5-4.

Table 5-4 HTTP Cookie Stickiness 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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create an HTTP cookie sticky group and enter sticky-cookie configuration mode.
`host1/Admin(config)# sticky http-cookie cisco.com GROUP2`
`host1/Admin(config-sticky-cookie)#`

Table 5-4 HTTP Cookie Stickiness Configuration Quick Start (continued)

Task and Command Example
5. Configure a timeout for HTTP cookie stickiness. `host1/Admin(config-sticky-cookie)# timeout 720`
6. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-cookie)# timeout activeconns`
7. Enable the replication of sticky table information to the standby context in case of a switchover in a redundancy configuration. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*. `host1/Admin(config-sticky-cookie)# replicate sticky`

8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with tall the real servers in the primary server farm and in the backup server farm.
`host1/Admin(config-sticky-cookie)# serverfarm SFARM1 backup`
`BKUP_SFARM2 sticky`

9. (Optional) Enable cookie insertion to allow the ACE to insert a cookie in the Set-Cookie header of the response from the server to the client. Use this command when the server is not setting the appropriate cookie. `host1/Admin(config-sticky-cookie)# cookie insert browser-expire`
10. (Optional) Configure the cookie sticky offset and length to instruct the ACE to use only a portion of the cookie (that part of the cookie that remains constant) for stickiness. `host1/Admin(config-sticky-cookie)# cookie offset 3000 length 1000`

11. (Optional) Configure the ACE to use an alternative cookie that appears in the URL string of the HTTP request from the client.
`host1/Admin(config-sticky-cookie)# cookie secondary`
`arrowpoint.com`

12. (Optional) Configure one or more static sticky cookie entries.
`host1/Admin(config-sticky-cookie)# static cookie-value corvette`
`rserver SERVER1 4000`

13. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Table 5-4 HTTP Cookie Stickiness Configuration Quick Start (continued)

Task and Command Example
14. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
15. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
16. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
17. Display your HTTP cookie sticky configuration. Make any modifications to your configuration as necessary, and then reenter the **show** command to verify your configuration changes. `host1/Admin# show running-config sticky`
18. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating an HTTP Cookie Sticky Group

Before you begin to configure an HTTP cookie sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

You can configure the ACE to learn a cookie from either the HTTP header of a client request or the Set-Cookie message sent by the server to a client. The ACE then uses the learned cookie to provide stickiness between a client and a server for the duration of a transaction.

To configure the ACE to use HTTP cookies for stickiness, use the **sticky http-cookie** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is as follows:

sticky http-cookie *name1 name2*

The keywords and arguments are as follows:

- **http-cookie** *name1*—Specifies that the ACE learn the cookie value from the HTTP header of the client request or from the Set-Cookie message from the server. Enter a unique identifier for the cookie as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *name2*—Unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a sticky group for cookie stickiness, enter:

```
host1/Admin(config)# sticky http-cookie cisco.com GROUP3
host1/Admin(config-sticky-cookie)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky http-cookie cisco.com GROUP3
```

Configuring a Cookie Sticky Timeout

The sticky timeout specifies the period of time that the ACE keeps the HTTP cookie sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection that matches that entry.

To configure a sticky timeout, use the **timeout** *minutes* command in sticky-cookie configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-cookie)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-cookie)# no timeout 720
```

Enabling a Sticky Cookie Timeout to Override Active Connections

To specify that the ACE time out HTTP cookie sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky-cookie configuration mode. The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-cookie)# timeout activeconns
```

To restore the ACE default of not timing out HTTP cookie sticky entries if active connections exist for those entries, enter:

```
host1/Admin(config-sticky-cookie)# no timeout activeconns
```

Enabling the Replication of Cookie Sticky Entries

If you are using redundancy, you can configure the ACE to replicate HTTP cookie sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate sticky table entries on the standby ACE, use the **replicate sticky** command in sticky-cookie configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of a sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. Thus, the standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-cookie)# replicate sticky
```

To restore the ACE default of not replicating sticky table entries, enter:

```
host1/Admin(config-sticky-cookie)# no replicate sticky
```

Enabling Cookie Insertion

Use cookie insertion when you want to use a session cookie for persistence if the server is not currently setting the appropriate cookie. With this feature enabled, the ACE inserts the cookie in the Set-Cookie header of the response from the server to the client. The ACE selects a cookie value that identifies the original server from which the client received a response. For subsequent connections of the same transaction, the client uses the cookie to stick to the same server.

Note With either TCP server reuse or persistence rebalance enabled, the ACE inserts a cookie in every client request. For information about TCP server reuse, see the "Configuring TCP Server Reuse" section in Chapter 3, Configuring Traffic Policies for Server Load Balancing. For information about persistence rebalance, see "Configuring HTTP Persistence Rebalance" in Chapter 3, Configuring Traffic Policies for Server Load Balancing.

To enable cookie insertion, use the **cookie insert** command in sticky-cookie configuration mode. The syntax of this command is as follows:

cookie insert [**browser-expire**]

The optional **browser-expire** keyword allows the client's browser to expire a cookie when the session ends.

For example, to enable cookie insertion and allow a browser to expire the cookie, enter:

```
host1/Admin(config-sticky-cookie)# cookie insert browser-expire
```

To disable cookie insertion, enter:

```
host1/Admin(config-sticky-cookie)# no cookie insert browser-expire
```

Configuring the Offset and Length of an HTTP Cookie

An HTTP cookie value may change over time with only a portion remaining constant throughout a transaction between the client and a server. You can configure the ACE to use the constant portion of a cookie to make persistent connections to a specific server. To define the portion of the cookie that you want the ACE to use, you specify cookie offset and length values. The ACE stores these values in the sticky table.

To configure the cookie offset and length, use the **cookie offset** command in sticky-cookie configuration mode. The syntax of this command is as follows:

cookie offset *number1* **length** *number2*

The keywords and arguments are as follows:

- **offset** *number1*—Specifies the portion of the cookie that the ACE uses to stick the client on a particular server by indicating the bytes to ignore starting with the first byte of the cookie. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the cookie.
- **length** *number2*—Specifies the length of the portion of the cookie (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is 1000.

For example, enter:

```
host1/Admin(config-sticky-cookie)# cookie offset 500 length 1000
```

To remove the cookie offset and length from the configuration, enter:

```
host1/Admin(config-sticky-cookie)# no cookie offset
```

Configuring a Secondary Cookie

You can configure an alternative cookie name that appears in the URL string of the web page on the server. The ACE uses this cookie to maintain a sticky connection between a client and a server and adds a secondary entry in the sticky table. To configure a secondary cookie, use the **cookie secondary** command in sticky-cookie configuration mode. The syntax of this command is as follows:

cookie secondary *name*

Enter a cookie name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin(config-sticky-cookie)# cookie secondary mysite.com
```

To remove a secondary cookie from the configuration, enter:

```
host1/Admin(config-sticky-cookie)# no cookie secondary mysite.com
```

Configuring a Static Cookie

You can configure the ACE to use static cookies from entries based on cookie values and, optionally, real server names and ports. Static cookie values remain constant over time. You can configure multiple static cookie entries, but only one unique real-server name can exist for a given static cookie value.

Note When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure a static cookie, use the s**tatic cookie-value** command in sticky-cookie configuration mode. The syntax of this command is as follows:

static cookie-value *value* **rserver** *name* [*number*]

The keywords, arguments, and options are as follows:

- *value*—Cookie string value. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. Alternatively, you can enter a text string with spaces provided that you enclose the string in quotation marks (“).
- **rserver** *name*—Specifies the hostname of an existing real server.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, enter:

```
host1/Admin(config-sticky-cookie)# static cookie-value CORVETTE
rserver SERVER1 4000
```

To remove a static cookie form the configuration, enter:

```
host1/Admin(config-sticky-cookie)# no static cookie-value CORVETTE
rserver SERVER1 4000
```

Associating a Server Farm with an HTTP Cookie Sticky Group

To complete a sticky group configuration, you must configure a server farm entry for the group. To configure a server farm entry for a sticky group, use the **serverfarm** command in sticky-cookie configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**] [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. Once clients are stuck to a backup server farm, they remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.
- **sticky**—(Optional) Specifies that the backup server farm is sticky.
- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-cookie)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-cookie)# no serverfarm
```

Example of HTTP Cookie Stickiness Configuration

The following example shows a running configuration that defines HTTP cookie stickiness. The HTTP cookie stickiness configuration appears in bold in the example.

In this configuration, the ACE uses HTTP cookie stickiness to allow a particular connection to be maintained with a specific server of a server farm for its duration.

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

probe icmp ICMP
  interval 2
  faildetect 2
  passdetect interval 2

rserver host SERVER1
  ip address 192.168.252.240
  inservice
rserver host SERVER2
  ip address 192.168.252.241
  inservice
rserver host SERVER3
  ip address 192.168.252.242
  inservice
serverfarm host SFARM1
  probe ICMP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice

sticky http-cookie COOKIE_TEST COOKIE-GROUP
  serverfarm SFARM1

class-map match-all L4STICKY-COOKIE-VIP_127:80_CLASS
  2 match virtual-address 192.168.120.127 tcp eq www
policy-map type loadbalance first-match L7PLBSF_STICKY-COOKIE_POLICY
  class class-default
    sticky-serverfarm COOKIE-GROUP
```

```
policy-map multi-match L4SH-Gold-VIPs_POLICY
  class L4STICKY-COOKIE-VIP_127:80_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_STICKY-COOKIE_POLICY
    loadbalance vip icmp-reply
    nat dynamic 1 vlan 120
    appl-parameter http advanced-options PERSIST-REBALANCE

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 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```

Configuring HTTP Header Stickiness

When a client requests a service from a server, the client sends a request to the ACE. The request contains an HTTP header that contains fields that the ACE can use to provide stickiness. This process ensures that connections from the same client that match the same SLB policy use the same server for subsequent connections based on the HTTP header fields. For HTTP, you can specify any supported protocol header name or select one of the standard protocol headers.

By default, the ACE CLI is case sensitive. For example, the ACE treats the sticky group names http_sticky_group1 and HTTP_STICKY_GROUP1 as two different group names. You can configure the CLI to be case insensitive for all HTTP-related parameters only by entering the **case-insensitive** command in an HTTP parameter map and then associating the parameter map with a Layer 3 and Layer 4 SLB policy map. For details, see the "Configuring an HTTP Parameter Map" section in Chapter 3, Configuring Traffic Policies for Server Load Balancing.

By default, the maximum number of bytes that the ACE parses to check for a cookie, HTTP header, or URL is 4096. If a cookie, HTTP header, or URL exceeds the default value, the ACE drops the packet and sends a RST (reset) to the client browser. You can increase the number of bytes that the ACE parses using the **set header-maxparse-length** command in HTTP parameter-map configuration mode. For details about setting the maximum parse length, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

You can also change the default behavior of the ACE when a cookie, header, or URL exceeds the maximum parse length using the **length-exceed** command in HTTP parameter-map configuration mode. For details, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- HTTP Header Stickiness Configuration Quick Start
- Creating an HTTP Header Sticky Group
- Configuring a Timeout for HTTP Header Stickiness
- Enabling an HTTP Header Sticky Timeout to Override Active Connections
- Enabling the Replication of HTTP Header Sticky Entries
- Configuring the Offset and Length of the HTTP Header
- Configuring a Static HTTP Header Sticky Entry
- Associating a Server Farm with an HTTP Header Sticky Group
- Example of HTTP Header Stickiness Configuration

HTTP Header Stickiness Configuration Quick Start

Table 5-5 provides a quick overview of the steps required to configure header stickiness on an ACE. 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 5-5.

Table 5-5 HTTP Header Stickiness 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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create an HTTP header sticky group and enter sticky-header configuration mode.
`host1/Admin(config)# sticky http-header Host HTTP_GROUP`
`host1/Admin(config-sticky-header)#`

5. Configure a timeout for header stickiness. `host1/Admin(config-sticky-header)# timeout 720`

Table 5-5 HTTP Header Stickiness Configuration Quick Start (continued)

Task and Command Example
6. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-header)# timeout activeconns`
7. Enable the replication of header sticky table information to the standby context in case of a switchover. Use this command with redundancy. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*. `host1/Admin(config-sticky-header)# replicate sticky`
8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and in the backup server farm. `host1/Admin(config-sticky-ssl)# serverfarm SFARM1 backup BKUP_SFARM2 sticky`
9. (Optional) Configure the header sticky offset and length to instruct the ACE to use only a portion of the header (that part of the header that remains constant) for stickiness. `host1/Admin(config-sticky-header)# header offset 3000 length 1000`
10. (Optional) Configure one or more static header sticky entries. `host1/Admin(config-sticky-header)# static header Host rserver SERVER1 4000`
11. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
12. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
13. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
14. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Table 5-5 HTTP Header Stickiness Configuration Quick Start (continued)

Task and Command Example
15. Display your header sticky configuration. Make any modifications to your configuration as necessary, and then reenter the **show** command to verify your configuration changes. `host1/Admin# show running-config sticky`
16. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating an HTTP Header Sticky Group

Before you begin to configure a header sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To create a header sticky group to enable the ACE to stick client connections to the same real server based on an HTTP header field, use the **sticky http-header** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is as follows:

sticky http-header *name1 name2*

The keywords and arguments are as follows:

- **http-header** *name1*—Specifies stickiness based on the HTTP header. Enter an HTTP header name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. Alternatively, you can select one of the standard HTTP headers listed in Table 5-6.

Table 5-6 Standard HTTP Header Fields

Field Name	Description
Accept	Semicolon-separated list of representation schemes (content type metainformation values) that will be accepted in the response to the request.
Accept-Charset	Character sets that are acceptable for the response. This field allows clients capable of understanding more comprehensive or special-purpose character sets to signal that capability to a server that can represent documents in those character sets.
Accept-Encoding	Restricts the content encoding that a user will accept from the server.
Accept-Language	ISO code for the language in which the document is written. The language code is an ISO 3316 language code with an optional ISO639 country code to specify a national variant.
Authorization	Directives that the user agent wants to authenticate itself with a server, usually after receiving a 401 response.
Cache-Control	Directives that must be obeyed by all caching mechanisms along the request-response chain. The directives specify behavior intended to prevent caches from adversely interfering with the request or response.
Connection	Directives that allows the sender to specify connection options.
Content-MD5	MD5 digest of the entity-body that provides an end-to-end integrity check. Only a client or an origin server can generate this header field.
Expect	Used by a client to inform the server about what behaviors the client requires.
From	E-mail address of the person that controls the requesting user agent.

Table 5-6 Standard HTTP Header Fields (continued)

Field Name	Description
Host	Internet host and port number of the resource being requested, as obtained from the original URI given by the user or referring resource. The Host field value must represent the naming authority of the origin server or gateway given by the original URL.
If-Match	Used with a method to make it conditional. A client that has one or more entities previously obtained from the resource can verify that one of those entities is current by including a list of their associated entity tags in the If-Match header field. This feature allows efficient updates of cached information with a minimum amount of transaction overhead. It is also used on updating requests to prevent inadvertent modification of the wrong version of a resource. As a special case, the value "*" matches any current entity of the resource.
Pragma	Pragma directives understood by servers to whom the directives are relevant. The syntax is the same as for other multiple-value fields in HTTP, for example, the **accept** field, a comma-separated list of entries, for which the optional parameters are separated by semicolons.
Referer	Address (URI) of the resource from which the URI in the request was obtained.
Transfer-Encoding	What (if any) type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient.

Table 5-6 Standard HTTP Header Fields (continued)

Field Name	Description
User-Agent	Information about the user agent, for example, a software program that originates the request. This information is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for customizing responses to avoid particular user agent limitations.
Via	Used by gateways and proxies to indicate the intermediate protocols and recipients between the user agent and the server on requests, and between the origin server and the client on responses.

- *name2*—Unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a group for HTTP header stickiness, enter:

```
host1/Admin(config-sticky-header)# sticky http-header Host HTTP_GROUP
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config-sticky-header)# no sticky http-header Host
HTTP_GROUP
```

Configuring a Timeout for HTTP Header Stickiness

The sticky timeout specifies the period of time that the ACE keeps the HTTP header sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** command in sticky-header configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-header)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-header)# no timeout 720
```

Enabling an HTTP Header Sticky Timeout to Override Active Connections

To specify that the ACE time out HTTP header sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky-header configuration mode. The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-header)# timeout activeconns
```

To restore the ACE default of not timing out header sticky entries if active connections exist for those entries, enter:

```
host1/Admin(config-sticky-header)# no timeout activeconns
```

Enabling the Replication of HTTP Header Sticky Entries

If you are using redundancy, you can configure the ACE to replicate HTTP header sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate header sticky table entries on the standby ACE, use the **replicate sticky** command in sticky-header configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of a sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. Thus, the standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-header)# replicate sticky
```

To restore the ACE default of not replicating HTTP header sticky table entries, enter:

```
host1/Admin(config-sticky-header)# no replicate sticky
```

Configuring the Offset and Length of the HTTP Header

You can configure the ACE to use a portion of an HTTP header to make persistent connections to a specific server. To define the portion of the header that you want the ACE to use, you specify header offset and length values. The ACE stores these values in the sticky table.

To configure the header offset and length, use the **header** command in sticky-header configuration mode. The syntax of this command is as follows:

header offset *number1* **length** *number2*

The keywords and arguments are as follows:

- **offset** *number1*—Specifies the portion of the header that the ACE uses to stick the client to a particular server by indicating the bytes to ignore starting with the first byte of the header. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the header.
- **length** *number2*—Specifies the length of the portion of the header (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is 1000.

For example, enter:

```
host1/Admin(config-sticky-header)# header offset 500 length 1000
```

To remove the header offset and length values from the configuration, enter:

```
host1/Admin(config-sticky-header)# no header offset
```

Configuring a Static HTTP Header Sticky Entry

You can configure static header sticky entries based on HTTP header values and, optionally, real server names and ports. Static sticky values remain constant over time. You can configure multiple header static entries, but only one unique real-server name can exist for a given static header sticky value.

Note When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure a static HTTP header, use the **static header-value** command in sticky-header configuration mode. The syntax of this command is as follows:

static header-value *value* **rserver** *name* [*number*]

The keywords, arguments, and options are as follows:

- *value*—Header value. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. Alternatively, you can enter a text string with spaces provided that you enclose the string in quotation marks (“).
- **rserver** *name*—Specifies the hostname of an existing real server.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, enter:

```
host1/Admin(config-sticky-header)# static header-value 12345678
rserver SERVER1 3000
```

To remove the static header entry from the sticky table, enter:

```
host1/Admin(config-sticky-header)# no static header-value 12345678
rserver SERVER1 3000
```

Associating a Server Farm with an HTTP Header Sticky Group

To complete an HTTP header sticky group configuration, you must configure a server farm entry for the group. To configure a server farm entry for a sticky group, use the **serverfarm** command in sticky-header configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**] [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. Once clients are stuck to a backup server farm, they remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.
- **sticky**—(Optional) Specifies that the backup server farm is sticky.
- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-header)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-header)# no serverfarm
```

Example of HTTP Header Stickiness Configuration

The following example shows a running configuration that defines HTTP header stickiness. The HTTP header stickiness configuration appears in bold in the example.

In this configuration, the ACE uses HTTP header stickiness to allow a particular connection to be maintained with a specific server of a server farm for its duration.

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

probe http HTTP
  interval 5
  passdetect interval 10
  receive 5
  expect status 200 200
  open 3

rserver host SERVER1
  ip address 192.168.252.240
  inservice
rserver host SERVER2
  ip address 192.168.252.241
  inservice
rserver host SERVER3
  ip address 192.168.252.242
  inservice
rserver host SERVER4
  ip address 192.168.252.243
  inservice
rserver host SERVER5
  ip address 192.168.252.244
  inservice
rserver host SERVER6
  ip address 192.168.252.245
  inservice
rserver host SERVER7
  ip address 192.168.252.246
  inservice
rserver host SERVER8
  ip address 192.168.252.247
  inservice
rserver host SERVER9
  ip address 192.168.252.248
  inservice
```

```
serverfarm host SFARM1
  probe HTTP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice
serverfarm host SFARM2
  probe HTTP
  rserver SERVER4
    inservice
  rserver SERVER5
    inservice
  rserver SERVER6
    inservice
serverfarm host DEFAULT
  probe ICMP
  rserver SERVER7
    inservice
  rserver SERVER8
    inservice
  rserver SERVER9
    inservice

sticky http-header MSISDN HEADER-GROUP1
  timeout 30
  serverfarm SFARM1
sticky http-header TestHeader HEADER-GROUP2
  header offset 15 length 7
  timeout 30
  serverfarm SFARM2

class-map match-all L4STICKY-HEADER_129:80_CLASS
  2 match virtual-address 192.168.120.129 tcp eq www
class-map type http loadbalance match-all L7MSISDN_CLASS
  2 match http header MSISDN header-value ".*"
class-map type http loadbalance match-all L7TESTHEADER_CLASS
  2 match http header TestHeader header-value ".*"
policy-map type loadbalance first-match L7PLBSF_STICKY-HEADER_POLICY
  class L7MSISDN_CLASS
    sticky-serverfarm HEADER-GROUP1
  class L7TESTHEADER_CLASS
    sticky-serverfarm HEADER-GROUP2
  class class-default
    serverfarm DEFAULT
```

```
policy-map multi-match L4SH-Gold-VIPs_POLICY
  class L4STICKY-HEADER_129:80_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_STICKY-HEADER_POLICY
    loadbalance vip icmp-reply active
    nat dynamic 1 VLAN 120
    appl-parameter http advanced-options PERSIST-REBALANCE

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 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```

Configuring RADIUS Attribute Stickiness

This section describes how to configure RADIUS-attribute stickiness on an ACE.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- RADIUS-Attribute Stickiness Configuration Quick Start
- Creating a RADIUS-Attribute Sticky Group

- Configuring a Timeout for RADIUS-Attribute Stickiness
- Enabling a RADIUS-Attribute Sticky Timeout to Override Active Connections
- Enabling the Replication of RADIUS-Attribute Sticky Entries
- Associating a Server Farm with a RADIUS Attribute Sticky Group

RADIUS-Attribute Stickiness Configuration Quick Start

Table 5-7 provides a quick overview of the steps required to configure RADIUS-attribute stickiness on an ACE. 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 5-7.

Table 5-7 RADIUS-Attribute Stickiness 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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

Table 5-7 RADIUS-Attribute Stickiness Configuration Quick Start

Task and Command Example

4. Create a RADIUS-attribute sticky group and enter sticky-header configuration mode.
`host1/Admin(config)# sticky radius framed-ip RADIUS_GROUP`
`host1/Admin(config-sticky-radius)#`

5. Configure a timeout for RADIUS-attribute stickiness. `host1/Admin(config-sticky-radius)# timeout 720`
6. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-radius)# timeout activeconns`
7. Enable the replication of RADIUS-attribute sticky table information to the standby context in case of a switchover. Use this command with redundancy. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*. `host1/Admin(config-sticky-radius)# replicate sticky`

8. Associate a server farm with the RADIUS-attribute sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and in the backup server farm.
`host1/Admin(config-sticky-radius)# serverfarm SFARM1 backup`
`BKUP_SFARM2 sticky`

9. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
10. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
11. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
12. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Table 5-7 RADIUS-Attribute Stickiness Configuration Quick Start

Task and Command Example
13. Display your RADIUS-attribute sticky configuration. Make any modifications to your configuration as necessary, then reenter the **show** command to verify your configuration changes. `host1/Admin# show running-config sticky`
14. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating a RADIUS-Attribute Sticky Group

Before you begin to configure a RADIUS-attribute sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To create a RADIUS-attribute sticky group to enable the ACE to stick client connections to the same real server based on a RADIUS attribute, use the **sticky radius framed-ip** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE.

The syntax of this command is as follows:

sticky radius framed-ip [**calling-station-id** | **username**] *name*

The keywords, arguments, and options are as follows:

- **calling-station-id**—(Optional) Specifies stickiness based on the RADIUS framed IP attribute and the calling station ID attribute.
- **username**—(Optional) Specifies stickiness based on the RADIUS framed IP attribute and the username attribute.
- *name*—Unique identifier of the RADIUS sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a group for RADIUS-attribute stickiness, enter:

```
host1/Admin(config-sticky-radius)# sticky radius framed-ip
calling-station-id RADIUS_GROUP
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config-sticky-radius)# no sticky radius framed-ip
calling-station-id RADIUS_GROUP
```

Configuring a Timeout for RADIUS-Attribute Stickiness

The sticky timeout specifies the period of time that the ACE keeps the RADIUS-attribute sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** command in sticky radius configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-radius)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-radius)# no timeout 720
```

Enabling a RADIUS-Attribute Sticky Timeout to Override Active Connections

To specify that the ACE time out RADIUS-attribute sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky radius configuration mode.

The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-radius)# timeout activeconns
```

To restore the ACE default of not timing out RADIUS-attribute sticky entries if active connections exist for those entries, enter:

```
host1/Admin(config-sticky-radius)# no timeout activeconns
```

Enabling the Replication of RADIUS-Attribute Sticky Entries

If you are using redundancy, you can configure the ACE to replicate RADIUS-attribute sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate RADIUS-attribute sticky table entries on the standby ACE, use the **replicate sticky** command in sticky radius configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of a sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. Thus, the standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-radius)# replicate sticky
```

To restore the ACE default of not replicating RADIUS-attribute sticky table entries, enter:

```
host1/Admin(config-sticky-radius)# no replicate sticky
```

Associating a Server Farm with a RADIUS Attribute Sticky Group

To complete a RADIUS-attribute sticky group configuration, you must configure a server farm entry for the group. To configure a server farm entry for a RADIUS-attribute sticky group, use the **serverfarm** command in sticky radius configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**] [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the RADIUS-attribute sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. Once clients are stuck to a backup server farm, they remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.
- **sticky**—(Optional) Specifies that the backup server farm is sticky.
- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-radius)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-radius)# no serverfarm
```

Configuring RTSP Session Stickiness

This section describes how to configure RTSP Session stickiness on an ACE. The ACE supports only the Session header field for stickiness.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- RTSP Session Stickiness Configuration Quick Start
- Creating an RTSP Header Sticky Group
- Configuring a Timeout for RTSP Session Stickiness
- Enabling an RTSP Header Sticky Timeout to Override Active Connections
- Enabling the Replication of RTSP Header Sticky Entries
- Configuring the Offset and Length of the RTSP Header
- Configuring a Static RTSP Header Sticky Entry
- Associating a Server Farm with an RTSP Header Sticky Group

RTSP Session Stickiness Configuration Quick Start

Table 5-8 provides a quick overview of the steps required to configure RTSP Session stickiness on an ACE. 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 5-8.

Table 5-8 RTSP Session Stickiness 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, change to, or directly log in to, the correct context.
`host1/Admin# `**`changeto C1`**
`host1/C1#`
The rest of the examples in this table use the Admin context for illustration purposes, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create an RTSP header sticky group and enter sticky-header configuration mode.
`host1/Admin(config)# `**`sticky rtsp-header Session RTSP_GROUP`**
`host1/Admin(config-sticky-header)#`

5. Configure a timeout for RTSP header stickiness. `host1/Admin(config-sticky-header)# `**`timeout 720`**

Table 5-8 RTSP Session Stickiness Configuration Quick Start (continued)

Task and Command Example
6. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-header)# timeout activeconns`
7. Enable the replication of RTSP header sticky table information to the standby context in case of a switchover. Use this command with redundancy. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*. `host1/Admin(config-sticky-header)# replicate sticky`
8. Associate a server farm with the RTSP header sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and in the backup server farm. `host1/Admin(config-sticky-header)# serverfarm SFARM1 backup BKUP_SFARM2 sticky`
9. (Optional) Configure one or more static RTSP header sticky entries. `host1/Admin(config-sticky-header)# static header Session rserver SERVER1 4000`
10. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
11. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
12. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
13. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Table 5-8 RTSP Session Stickiness Configuration Quick Start (continued)

Task and Command Example
14. Display your RTSP header sticky configuration. Make any modifications to your configuration as necessary, and then reenter the **show** command to verify your configuration changes. `host1/Admin# show running-config sticky`
15. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Creating an RTSP Header Sticky Group

Before you begin to configure an RTSP header sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To create an RTSP header sticky group to enable the ACE to stick client connections to the same real server based on the RTSP Session header field, use the **sticky rtsp-header** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is as follows:

sticky rtsp-header Session *name*

The keywords and arguments are as follows:

- **Session**—Specifies stickiness based on the RTSP Session header field.
- *name*—Unique identifier of the RTSP sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a group for RTSP header stickiness, enter:

```
host1/Admin(config)# sticky rtsp-header Session RTSP_GROUP
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky rtsp-header Session RTSP_GROUP
```

Configuring a Timeout for RTSP Session Stickiness

The sticky timeout specifies the period of time that the ACE keeps the RTSP header sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** command in sticky-header configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-header)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-header)# no timeout 720
```

Enabling an RTSP Header Sticky Timeout to Override Active Connections

To specify that the ACE time out RTSP header sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky-header configuration mode. The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-header)# timeout activeconns
```

To restore the ACE default of not timing out header sticky entries if active connections exist for those entries, enter:

```
host1/Admin(config-sticky-header)# no timeout activeconns
```

Enabling the Replication of RTSP Header Sticky Entries

If you are using redundancy, you can configure the ACE to replicate RTSP header sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate header sticky table entries on the standby ACE, use the **replicate sticky** command in sticky-header configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of a sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. Thus, the standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-header)# replicate sticky
```

To restore the ACE default of not replicating RTSP header sticky table entries, enter:

```
host1/Admin(config-sticky-header)# no replicate sticky
```

Configuring the Offset and Length of the RTSP Header

You can configure the ACE to use a portion of the RTSP Session header to make persistent connections to a specific server. To define the portion of the Session header that you want the ACE to use, you specify header offset and length values. The ACE stores these values in the sticky table.

To configure the RTSP Session header offset and length, use the **header** command in sticky-header configuration mode. The syntax of this command is as follows:

header offset *number1* **length** *number2*

The keywords and arguments are as follows:

- **offset** *number1*—Specifies the portion of the header that the ACE uses to stick the client to a particular server by indicating the bytes to ignore starting with the first byte of the header. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the header.
- **length** *number2*—Specifies the length of the portion of the header (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is 1000.

For example, enter:

```
host1/Admin(config-sticky-header)# header offset 500 length 1000
```

To remove the header offset and length values from the configuration, enter:

```
host1/Admin(config-sticky-header)# no header offset
```

Configuring a Static RTSP Header Sticky Entry

You can configure static RTSP header sticky entries based on header values and, optionally, real server names and ports. Static sticky values remain constant over time. You can configure multiple header static entries, but only one unique real-server name can exist for a given static header sticky value.

Note When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure a static header, use the **static header-value** command in sticky-header configuration mode. The syntax of this command is as follows:

static header-value *value* **rserver** *name* [*number*]

The keywords, arguments, and options are as follows:

- *value*—Header value. 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 string in quotation marks (“).
- **rserver** *name*—Specifies the hostname of an existing real server.

- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, enter:

```
host1/Admin(config-sticky-header)# static header-value 12345678 
rserver SERVER1 3000
```

To remove the static RTSP header entry from the sticky table, enter:

```
host1/Admin(config-sticky-header)# no static header-value 12345678 
rserver SERVER1 3000
```

Associating a Server Farm with an RTSP Header Sticky Group

To complete an RTSP header sticky group configuration, you must configure a server farm entry for the group. To configure a server farm entry for an RTSP sticky group, use the **serverfarm** command in sticky-header configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**] [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the RTSP header sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. Once clients are stuck to a backup server farm, they remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.
- **sticky**—(Optional) Specifies that the backup server farm is sticky.
- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of

service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-header)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-header)# no serverfarm
```

Configuring SIP Call-ID Stickiness

This section describes how to configure SIP Call-ID stickiness on an ACE. The ACE supports only the SIP Call-ID header field for stickiness.

Note You can associate a maximum of 1024 instances of the same type of regex with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- SIP Call-ID Stickiness Configuration Quick Start
- Creating a SIP Header Sticky Group
- Configuring a Timeout for SIP Call-ID Stickiness
- Enabling a SIP Header Sticky Timeout to Override Active Connections

- Enabling the Replication of SIP Header Sticky Entries
- Configuring a Static SIP Header Sticky Entry
- Associating a Server Farm with a SIP Header Sticky Group

SIP Call-ID Stickiness Configuration Quick Start

Table 5-9 provides a quick overview of the steps required to configure SIP Call-ID stickiness on an ACE. 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 5-9.

Table 5-9 SIP Call-ID Stickiness 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, change to, or directly log in to, the correct context.
`host1/Admin#` **`changeto C1`**
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section. For details about configuring a resource class and allocating resources, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

Table 5-9 SIP Call-ID Stickiness Configuration Quick Start (continued)

Task and Command Example

4. Create a SIP-header sticky group and enter sticky-header configuration mode.

```
host1/Admin(config)# sticky sip-header Call-ID SIP_GROUP
host1/Admin(config-sticky-header)#
```

5. Configure a timeout for SIP header stickiness.

```
host1/Admin(config-sticky-header)# timeout 720
```

6. (Optional) Enable the timeout to override active connections.

```
host1/Admin(config-sticky-header)# timeout activeconns
```

7. Enable the replication of header sticky table information to the standby context in case of a switchover. Use this command with redundancy. For details about configuring redundancy on an ACE, see the *Cisco Application Control Engine Module Administration Guide*.

```
host1/Admin(config-sticky-header)# replicate sticky
```

8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and in the backup server farm.

```
host1/Admin(config-sticky-header)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

9. (Optional) Configure one or more static header sticky entries.

```
host1/Admin(config-sticky-header)# static header Call-ID rserver
SERVER1 4000
```

10. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

11. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

12. Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

13. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Table 5-9 SIP Call-ID Stickiness Configuration Quick Start (continued)

Task and Command Example
14. Display your header sticky configuration. Make any modifications to your configuration as necessary, and then reenter the **show** command to verify your configuration changes. `host1/Admin# `**`show running-config sticky`**
15. (Optional) Save your configuration changes to flash memory. `host1/Admin# `**`copy running-config startup-config`**

Creating a SIP Header Sticky Group

Before you begin to configure a SIP header sticky group, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To create a SIP header sticky group to enable the ACE to stick client connections to the same real server based on the SIP Call-ID header field, use the **sticky sip-header** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is as follows:

sticky sip-header Call-ID *name*

The keywords and arguments are as follows:

- **sip-header Call-ID**—Specifies stickiness based on the SIP Call-ID header field.
- *name*—Unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a group for SIP-header stickiness, enter:

```
host1/Admin(config-sticky-header)# sticky sip-header Call-ID SIP_GROUP
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config-sticky-header)# no sticky sip-header Call-ID
SIP_GROUP
```

Configuring a Timeout for SIP Call-ID Stickiness

The sticky timeout specifies the period of time that the ACE keeps the SIP header sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** command in sticky-header configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-header)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-header)# no timeout 720
```

Enabling a SIP Header Sticky Timeout to Override Active Connections

To specify that the ACE time out SIP header sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky-header configuration mode. The syntax of this command is as follows:

timeout activeconns

For example, enter:

```
host1/Admin(config-sticky-header)# timeout activeconns
```

To restore the ACE default of not timing out SIP header sticky entries if active connections exist for those entries, enter:

```
host1/Admin(config-sticky-header)# no timeout activeconns
```

Enabling the Replication of SIP Header Sticky Entries

If you are using redundancy, you can configure the ACE to replicate SIP header sticky table entries on the standby ACE so if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate SIP header sticky table entries on the standby ACE, use the **replicate sticky** command in sticky-header configuration mode. The syntax of this command is as follows:

replicate sticky

Note The timer of a sticky table entry on the standby ACE is reset every time the entry is synchronized with the active ACE entry. Thus, the standby sticky entry may have a lifetime up to twice as long as the active entry. However, if the entry expires on the active ACE or a new real server is selected and a new entry is created, the old entry on the standby ACE is replaced.

For example, enter:

```
host1/Admin(config-sticky-header)# replicate sticky
```

To restore the ACE default of not replicating SIP header sticky table entries, enter:

```
host1/Admin(config-sticky-header)# no replicate sticky
```

Configuring a Static SIP Header Sticky Entry

You can configure static header sticky entries based on header values and, optionally, real server names and ports. Static sticky values remain constant over time. You can configure multiple SIP header static entries, but only one unique real-server name can exist for a given static SIP header sticky value.

Note When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure a static SIP header, use the **static header-value** command in sticky-header configuration mode. The syntax of this command is as follows:

static header-value *value* **rserver** *name* [*number*]

The keywords, arguments, and options are as follows:

- *value*— SIP header value. 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 string in quotation marks (“).
- **rserver** *name*—Specifies the hostname of an existing real server.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, enter:

```
host1/Admin(config-sticky-header)# static header-value 12345678
rserver SERVER1 3000
```

To remove the static header entry from the sticky table, enter:

```
host1/Admin(config-sticky-header)# no static header-value 12345678
rserver SERVER1 3000
```

Associating a Server Farm with a SIP Header Sticky Group

To complete a SIP header sticky group configuration, you must configure a server farm entry for the group. To configure a server farm entry for a sticky group, use the **serverfarm** command in sticky-header configuration mode. The syntax of this command is as follows:

serverfarm *name1* [**backup** *name2* [**sticky**] [**aggregate-state**]]

The keywords, arguments, and options are as follows:

- *name1*—Identifier of an existing server farm that you want to associate with the sticky group. You can associate one server farm with each sticky group.
- **backup** *name2*—(Optional) Specifies an identifier of an existing server farm that you want the ACE to use as a backup server farm. If the primary server farm goes down, the ACE uses the configured backup server farm. Once clients are stuck to a backup server farm, they remain stuck to the backup even if the primary server farm becomes active again. For more information about backup server farms, see the "Backup Server Farm Behavior with Stickiness" section.
- **sticky**—(Optional) Specifies that the backup server farm is sticky.
- **aggregate-state**—This option has been deprecated and no longer has an effect on the state of the VIP. By default, the ACE takes into account the state of all real servers in the backup server farm before taking the VIP out of service. If all real servers in the primary server farm fail, but there is at least one real server in the backup server farm that is operational, the ACE keeps the VIP in service.

For example, to associate a server farm with a sticky group and specify a sticky backup server farm, enter:

```
host1/Admin(config-sticky-header)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-header)# no serverfarm
```

Configuring SSL Session-ID Stickiness

This section describes how to configure SSL Session-ID stickiness on the ACE. This feature allows the ACE to stick the same client to the same SSL server based on the unique SSL Session ID for the duration of an SSL session.

When a client requests a new SSL connection with an SSL server, the initial TCP connection has an SSL Session ID of zero. This zero value tells the server that it needs to set up a new SSL session and to generate an SSL Session ID. The server sends the new SSL Session ID in its response to the client as part of the SSL handshake. The ACE learns the new SSL Session ID from the server and enters the value in the sticky table. For subsequent SSL connections from the same client, the ACE uses the SSL Session-ID value in the sticky table to stick the client to the same SSL server for the duration of the SSL session, provided that the SSL Session ID is not renegotiated.

SSL Session-ID stickiness uses a specific configuration of the generic protocol parsing (GPP) feature of the ACE. The syntax and descriptions of some of the commands used in the following sections have been simplified to include only those details necessary to configure SSL Session-ID stickiness. For a complete description of all stickiness-related commands referenced in this section, see the "Configuring Layer 4 Payload Stickiness" section.

Note You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
- Inline match statements in Layer 7 policy maps
- Layer 7 hash predictors for server farms
- Layer 7 sticky expressions in sticky groups
- Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists

This section contains the following topics:

- Configuration Requirements and Considerations
- SSL Session-ID Stickiness Configuration Quick Start
- Creating a Layer 4 Payload Sticky Group
- Configuring a Layer 4 Payload Sticky Timeout
- Associating a Server Farm with a Layer 4 Payload Sticky Group
- Enabling SSL Session-ID Learning from the SSL Server
- Configuring the Offset, Length, and Beginning Pattern for the SSL Session ID
- Example of an SSL Session-ID Sticky Configuration for a 32-Byte SSL Session ID

Configuration Requirements and Considerations

When you configure SSL Session-ID stickiness, keep in mind the following configuration requirements and considerations:

- Ensure that you configure a resource class, allocate a minimum percentage of resources to stickiness, and associate one or more contexts where you want to configure SSL Session-ID stickiness with the resource class. See the "Configuration Requirements and Considerations for Configuring Stickiness" section.
- This feature supports SSLv3 and TLS1 only. SSLv2 places the SSL ID within the encrypted data, which makes it impossible for the ACE or any other load balancer to inspect it.
- The ACE supports 1- to 32-byte SSL Session IDs.
- SSL Session-ID stickiness is necessary typically only when the ACE does not terminate SSL traffic.
- You must configure a generic protocol parsing (GPP) policy map.
- Configure a generic parameter map to specify the maximum number of bytes in the TCP payload that you want the ACE to parse. The value of the maximum parse length should always be 70.

SSL Session-ID Stickiness Configuration Quick Start

Table 5-10 provides a quick overview of the steps required to configure 32-byte SSL Session-ID stickiness on an ACE. Each step includes the CLI command or a reference to the procedure that is required to complete the task. For a complete description of each task and details associated with the CLI commands, see the sections following Table 5-10.

Table 5-10 32-Byte SSL Session-ID Stickiness 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, change to, or directly log in to, the correct context.
`host1/Admin# `**`changeto C1`**
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

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

3. Configure real servers for the SSL servers and associate them with an SSL server farm. See Chapter 2, Configuring Real Servers and Server Farms.

4. Create a Layer 4 payload sticky group for 32-byte (the default length) SSL IDs and enter sticky Layer 4 payload configuration mode.
`host1/Admin(config)# `**`sticky layer4-payload SSL_GROUP`**
`host1/Admin(config-sticky-l4payloa)#`

5. Configure a timeout for SSL Session-ID stickiness. `host1/Admin(config-sticky-l4payloa)# `**`timeout 600`**
6. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of all the real servers in the primary server farm and all the real servers in the backup server farm. `host1/Admin(config-sticky-l4payloa)# `**`serverfarm SSL_SFARM1`**
7. Configure sticky learning so that the ACE can learn the SSL Session ID from the server response. `host1/Admin(config-sticky-l4payloa)# `**`response sticky`**

Table 5-10 32-Byte SSL Session-ID Stickiness Configuration Quick Start (continued)

Task and Command Example

8. Configure the Layer 4 payload offset, length, and beginning pattern to instruct the ACE to parse only that portion of the payload that contains the 32-byte SSL ID.

```
host1/Admin(config-sticky-l4payloa)# layer4-payload offset 43
length 32 begin-pattern “\x20”
host1/Admin(config-sticky-l4payloa)# exit
```

9. Configure a generic parameter map to specify the maximum number of bytes in the TCP payload that you want the ACE to parse.

```
host1/Admin(config)# parameter-map type generic SSLID_PARAMMAP
host1/Admin(config-parammap-generi)# set max-parse-length 70
host1/Admin(config)# exit
```

10. Configure a Layer 7 generic policy map.

```
host1/Admin(config)# policy-map type loadbalance generic
first-match SSLID_32_POLICY
host1/Admin(config-pmap-lb-generic)# class class-default
host1/Admin(config-pmap-lb-generic-c)# sticky serverfarm
SSL_GROUP
host1/Admin(config-pmap-lb-generic-c)# exit
host1/Admin(config-pmap-lb-generic)# exit
host1/Admin(config)#
```

11. Configure a Layer 3 and Layer 4 server load-balancing traffic policy. Associate the Layer 7 SLB traffic policy and the generic parameter map with the Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

12. Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.

13. Display your SSL Session-ID sticky configuration. Make any modifications to your configuration as necessary, and then reenter the **show** command to verify your configuration changes.

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

Creating a Layer 4 Payload Sticky Group

Before you begin to configure a Layer 4 payload sticky group for SSL Session-ID stickiness, be sure that you have allocated resources to stickiness as described in the "Configuration Requirements and Considerations for Configuring Stickiness" section.

To create one Layer 4 payload sticky group using the **sticky layer4-payload** command in configuration mode. The syntax of this command is as follows:

sticky layer4-payload *name*

The *name* argument is the unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to configure an SSL ID sticky group, enter:

```
host1/Admin(config)# sticky layer4-payload SSL_GROUP
host1/Admin(config-sticky-l4payloa)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky layer4-payload SSL_GROUP
```

Configuring a Layer 4 Payload Sticky Timeout

The sticky timeout specifies the period of time that the ACE keeps the SSL Session-ID stickiness information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific SSL ID sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** *minutes* command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

timeout *minutes*

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-l4payloa)# timeout 600
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-l4payloa)# no timeout 600
```

Associating a Server Farm with a Layer 4 Payload Sticky Group

For each sticky group, you must associate a server farm with the group. To associate a server farm with a sticky group, use the **serverfarm** command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

serverfarm *name1*

The *name1* argument specifies the identifier of an existing server farm that you want to associate with the sticky group. You can associate one server farm with each sticky group.

For example, to associate a server farm with a sticky group, enter:

```
host1/Admin(config-sticky-l4payloa)# serverfarm SSL_SFARM1
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-l4payloa)# no serverfarm
```

Enabling SSL Session-ID Learning from the SSL Server

To enable the ACE to parse SSL server responses and learn the SSL Session ID, use the **response sticky** command in sticky Layer 4 payload configuration mode. The ACE uses a hash of the SSL server response bytes to populate the sticky database. The next time that the ACE receives a client request with the same SSL Session ID, it sticks the client to the same SSL server. The syntax of this command is as follows:

response sticky

For example, to enable the ACE to parse the response bytes from an SSL server and learn the SSL Session ID, enter:

```
host1/Admin(config-sticky-l4payloa)# response sticky
```

To reset the behavior of the ACE to the default of not parsing SSL server responses and performing sticky learning, enter the following command:

```
host1/Admin(config-sticky-l4payloa)# no response sticky
```

Configuring the Offset, Length, and Beginning Pattern for the SSL Session ID

For SSLv3/TLS1, the SSL Session ID always appears at the same location in the TCP packet payload. You can configure the ACE to use the constant portion of the payload to make persistent connections to a specific SSL server. To define the portion of the payload that you want the ACE to use, you specify payload offset and length values. The ACE stores these values in the sticky table.

You can also specify a beginning pattern based on a regex that the ACE uses to stick a client to a particular SSL server. For information about regular expressions, see Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Note The inclusion of the “\xST”(STop) metacharacter aids the ACE in properly load-balancing SSL session-ID packets. Without the “\xST” metacharacter in regexes, certain SSL session-ID packets may get stuck in the ACE HTTP engine and eventually time out the connection. For information on the use of the “\xST” metacharacter for regular expressions, see the “Using the “\xST” Metacharacter in Regular Expressions for Layer 4 Generic Data Parsing” section.

To configure the payload offset, length, and beginning pattern, use the **layer4-payload** command in sticky Layer 4 payload configuration mode. The syntax of this command is as follows:

layer4-payload offset *number1* **length** *number2* **begin-pattern** *expression*

The keywords, arguments, and options are as follows:

- **offset** *number1*—Specifies the portion of the payload that the ACE uses to stick the client to a particular SSL server by indicating the bytes to ignore starting with the first byte of the payload. Enter **43**, the offset at which the SSL ID always appears for SSLv3.

- **length** *number2*—Specifies the length of the portion of the payload (starting with the byte after the offset value) that the ACE uses for sticking the client to the SSL server. Enter **32** for a 32-byte SSL ID.
- **begin-pattern** *expression*—Specifies the beginning pattern of the payload and the pattern string to match before hashing. If you do not specify a beginning pattern, the ACE begins parsing immediately after the offset byte. For a 32-byte SSL ID, enter **“\x20|\x00\xST)”**. The ACE supports the use of regexes for matching string expressions. Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing, lists the supported characters that you can use for matching string expressions.

For example, to specify a 32-byte SSL Session ID, enter:

```
host1/Admin(config-sticky-l4payloa)# layer4-payload offset 43 length
32 begin-pattern “\x20|\x00\xST)”
```

To remove the payload offset, length, and beginning pattern from the configuration, enter:

```
host1/Admin(config-sticky-l4payloa)# no layer4-payload
```

Example of an SSL Session-ID Sticky Configuration for a 32-Byte SSL Session ID

The following configuration example shows how to configure SSL Session-ID stickiness for 32-byte SSL IDs. It includes a regular load-balancing policy in addition to the SSL Session-ID sticky commands. The SSL Session-ID sticky-specific commands and related commands are in bold text.

```
resource-class RC1
  limit-resource sticky minimum 10.00 maximum equal-to-min

access-list ACL1 line 10 extended permit ip any any

parameter-map type generic SSLID_PARAMMAP
  set max-parse-length 70

rserver SSL_SERVER1
  ip address 192.168.12.2
  inservice
rserver SSL_SERVER2
  ip address 192.168.12.3
  inservice

serverfarm SSL_SFARM1
  rserver SSL_SERVER1
    inservice
  rserver SSL_SERVER2
    inservice

sticky layer4-payload SSL_GROUP
  timeout 600
  serverfarm SSL_SFARM1
  response sticky
  layer4-payload offset 43 length 32 begin-pattern “\x20|\x00\xST)”

class-map match-any L4_CLASS
  match virtual-address 172.27.16.2 tcp eq any
```

```
policy-map type loadbalance generic first-match SSLID_32_POLICY
  class class-default
  sticky-serverfarm SSL_GROUP

policy-map multi-match L4_POLICY
  class L4_CLASS
    loadbalance vip advertise active
    loadbalance vip inservice
    loadbalance vip icmp-reply active
    loadbalance policy SSLID_32_POLICY
    appl-parameter generic advanced-options SSLID-PARAMMAP

interface vlan 200
  ip address 172.27.16.1 255.255.255.0
  access-group input ACL1
  service-policy input L4_POLICY

interface vlan 300
  ip address 192.168.12.1 255.255.255.0

context Admin
  member RC1
```

Note For SSL Session-ID stickiness for different lengths of Session IDs, you can configure as many class maps as necessary.

Configuring the Reverse IP Stickiness Feature

This section describes the reverse IP stickiness feature that is used primarily in firewall load balancing (FWLB) to ensure that applications with separate control and data channels use the same firewall for ingress and egress flows for a given connection. It contains the following subsections:

- Overview of Reverse IP Stickiness
- Configuration Requirements and Restrictions
- Configuring Reverse IP Stickiness
- Displaying Reverse IP Sticky Status and Statistics
- Reverse IP Stickiness Configuration Examples

Overview of Reverse IP Stickiness

Reverse IP stickiness is an enhancement to regular stickiness and is used mainly in FWLB. It ensures that multiple distinct connections that are opened by hosts at both ends (client and server) are load-balanced and stuck to the same firewall. Reverse stickiness applies to such protocols as FTP, RTSP, SIP, and so on where there are separate control channels and data channels opened by the client and the server, respectively.

You configure reverse IP stickiness as an action under a Layer 7 load-balancing policy map by associating an existing IP address sticky group with the policy using the **reverse-sticky** command. Then you associate the Layer 7 policy map with a Layer 4 multi-match policy map and apply the Layer 4 policy map as a service policy on the ACE interface between the firewalls and the ACE. When incoming traffic matches the policy, the ACE verifies that a reverse IP sticky group is associated with the policy. If the association exists, the ACE creates a sticky entry in the sticky table that maps the opposite IP address (for example, the destination IP address if source IP sticky is configured) to the real server ID, which is the ID of the firewall. To obtain the real ID of the firewall, the ACE uses the encapsulation (encap) ID from the traffic coming from the firewall as a lookup key into the list of real servers in the server farm.

Note The ACE sticky table, which holds a maximum of 4 million entries, is shared across all sticky types, including reverse IP stickiness.

This section contains the following topics:

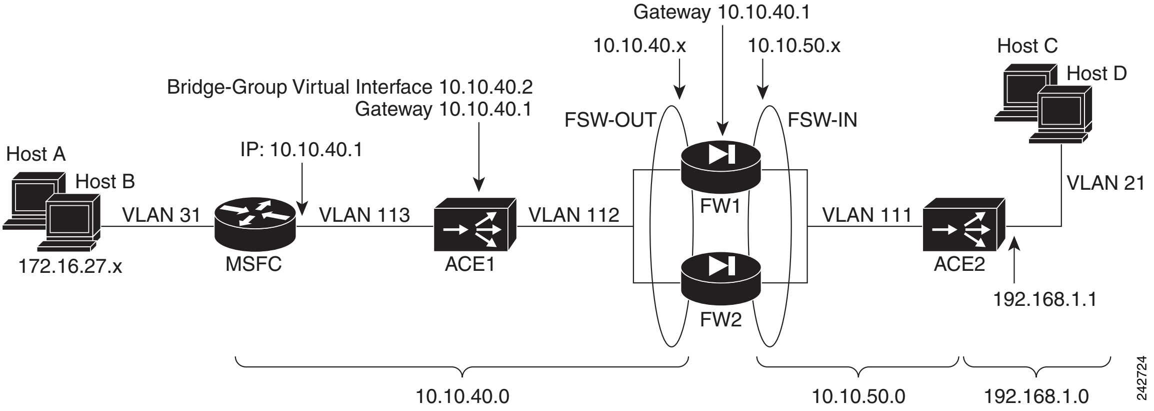
- Symmetric Topology
- Asymmetric Topology

Symmetric Topology

A typical firewall load-balancing topology (symmetric) includes two dedicated ACEs with the firewalls positioned between the ACEs. In this scenario, the ACEs are used exclusively for FWLB and simply forward traffic through their host interfaces in either direction. See Figure 1.

The hosts in either VLAN 31 or VLAN 21 can initiate the first connection and the hosts on both sides of the connection can "see" each other directly. Therefore, only catch-all VIPs (with an IP address of 0.0.0.0 and a netmask of 0.0.0.0) are configured on the ACE interfaces.

Figure 1 Typical Symmetric Firewall Load-Balancing Topology for Reverse IP Stickiness



For the network diagram shown in Figure 1, the following steps describe a possible connection scenario with reverse IP stickiness:

Step 1 Host A (a client) initiates an FTP control channel connection to the IP address of Host C (an FTP server).

Step 2 ACE 1 load balances the connection to one of the two firewalls (FW1 or FW2) in the FWS-OUT server farm. ACE 1 is configured with a source IP sticky group that is associated with a policy map, which is applied to interface VLAN 113. This configuration ensures that all connections coming from the same host (or directed to the same host) are load balanced to the same firewall. The ACE creates a sticky entry that maps the IP address of Host A to one of the firewalls.

Step 3 The firewall that receives the packets from ACE 1 forwards them to ACE 2.

Step 4 Assume that a sticky group that is based on the destination IP address is associated with a policy map and is applied to interface VLAN 21. The same sticky group is associated as a reverse sticky group with the policy that is applied to VLAN 111. When it receives the packets, ACE 2 creates a sticky entry in the sticky database based on the source IP address (because the sticky group is based on the destination IP address in this case), which maps the Host A IP address to the firewall in the FWS-IN server farm from which the traffic was received. Then, ACE 2 forwards the packets to the FTP server (Host C) in the server farm.

Step 5 If you have enabled the **mac-sticky** command on the VLAN 111 interface, ACE 2 forwards return traffic from the same connection to the same firewall from which the incoming traffic was received. The firewall routes the return traffic through ACE 1, which in turn forwards it to the MSFC and from there to the client.

Step 6 Now suppose that Host C (an FTP server) opens a new connection (for example, the corresponding FTP data channel of the previously opened FTP control channel) to the IP address of Host A. Because a sticky group based on destination IP is associated with the policy applied to interface VLAN 21, ACE 2 performs a sticky lookup and finds a valid sticky entry (the one created in Step 4) in the sticky database that allows ACE 2 to load balance the packets to the same firewall that the control connection traversed.

Step 7 The firewall routes the packets through ACE 1, which in turn forwards them to the MSFC and from there to the client (Host A).

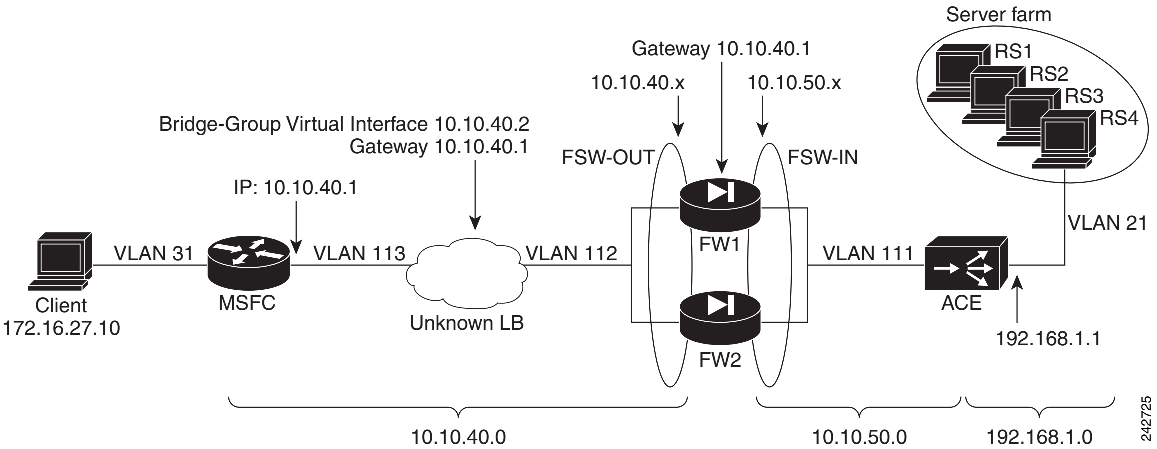
Follow these guidelines and observations when you configure reverse IP stickiness:

- When reverse IP sticky is enabled, the sticky entry is populated in one direction (for incoming traffic) and looked up in the opposite direction (for outgoing traffic), allowing traffic to flow through the same firewall in both directions.
- The example that is described in the steps above is symmetric because it does not matter on which side of the connections that the clients and servers reside. Everything would work in a similar manner if Host C was a client opening the FTP control channel and Host A was a server opening the FTP data channel, assuming that a reverse sticky group was also configured on the ACE 1 VLAN 112 interface. To make reverse IP stickiness work symmetrically, you must apply a reverse sticky group to the ACE interfaces that are associated with the firewall server farm (in this example, VLAN 112 and VLAN 111) and apply the same sticky group as a regular sticky group to the ACE interfaces associated with the hosts (in this example, VLAN 113 and VLAN 21).
- In this example, the assumption is to have a regular sticky group based on the source IP associated with the VLAN 113 interface of the ACE 1 module and another sticky group based on the destination IP associated with the VLAN 21 interface of the ACE 2 module (the reverse sticky groups on VLAN 112 and VLAN 111 would be based on the opposite IPs). Everything would work correctly if the regular sticky groups were reversed, that is, the sticky group on VLAN 113 was based on the destination IP and the one on VLAN 21 was based on the source IP, or if both regular sticky groups were based on both the source and the destination IP.

Asymmetric Topology

The following scenario is asymmetric because it cannot work equally in both directions as in the previous scenario. In this setup, one of the load balancers is unknown (Unknown LB) so that it is uncertain whether the load balancer supports reverse sticky. The clients must be on one side of the connection and the servers must be on the other side with the clients opening the first connection to the servers. See Figure 2. In this scenario, the ACE performs only FWLB and forwards traffic to the real servers in the server farm.

Figure 2 Asymmetric Firewall Load Balancing Topology for Reverse IP Stickiness



For the network diagram shown in Figure 2, the following steps describe the sequence of events for establishing a connection with reverse IP stickiness:

Step 1 A client initiates a connection (for example, an FTP control channel connection) to the IP address of one of the servers in the server farm.

Step 2 The Unknown LB load balances the connection to one of the two firewalls in the FWS-OUT server farm. The Unknown LB should, at a minimum, support load balancing based on the source or destination IP address hash predictor. These predictors ensure that all connections coming from the same client (or destined to the same server) are load balanced to the same firewall. Assume in this example that a predictor based on source IP hash is configured in the Unknown LB, so that all traffic coming from the same client will be directed to the same firewall.

Step 3 The firewall that receives the packet forwards it to the ACE.

Step 4 Assume that a sticky group that is based on the destination IP address is associated with a policy map that is applied to interface VLAN 21 using a service policy. The same sticky group is associated as a reverse sticky group with the policy that is applied to VLAN 111. When it receives the packets, the ACE creates a sticky entry in the sticky database based on the source IP address (because the sticky group is based on the destination IP in this case), which maps the Host A IP address to the firewall in the FWS-IN server farm from which the traffic was received. Then, the ACE forwards the packets to the FTP server (Host C) in the server farm.

Step 5 If you have enabled the **mac-sticky** command on VLAN 111, the ACE forwards the return traffic for the same connection to the same firewall from which the incoming traffic was received. The firewall routes the return traffic through the Unknown-LB, which in turn forwards it to the MSFC and then to the client.

Step 6 Now suppose that the FTP server opens a new connection (for example, the corresponding FTP data channel of the previously opened FTP control channel) to the IP address of the client. Because a sticky group based on the destination IP address is associated with the policy applied to interface VLAN 21, the ACE performs a sticky lookup and finds a valid sticky entry (the one created in Step 4) in the sticky database that allows the ACE to load balance the packets to the same firewall that the control connection traversed.

Step 7 The firewall routes the packet through the Unknown LB, which in turn forwards it to the MSFC and then to the client.

In this scenario, reverse sticky would also work properly under the following conditions:

- The sticky group is associated with the policy map as a regular sticky group based on source the IP and applied to the VLAN 21 interface.
- The sticky group is associated with the policy map as a reverse sticky group (based on the destination IP address) and applied to the VLAN 111 interface.
- The Unknown LB has a predictor based on the hash of the destination IP.

For more information about configuring firewall load balancing, see the *Cisco Application Control Engine Module Server Load-Balancing Configuration Guide*.

Configuration Requirements and Restrictions

Before attempting to configure reverse IP stickiness, be sure that you have met the following configuration requirements and restrictions:

- A sticky group of type IP netmask based on source IP, destination IP, or both must be present in your configuration.
- The sticky group cannot be a static sticky group.
- Once you have associated reverse IP stickiness with a sticky group, you cannot change that sticky group to a static sticky group.

- For firewall load balancing, configure the **mac-sticky** command on the ACE interface that is connected to the firewall.

Configuring Reverse IP Stickiness

To configure reverse IP stickiness, use the **reverse-sticky** command in policy map loadbalance class configuration mode. The syntax of this command is as follows:

reverse-sticky *name*

The *name* argument specifies the unique identifier of an existing IP address sticky group. Enter the name of an existing IP address sticky group as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to configure reverse IP stickiness for a sticky group called DEST_IP_STICKY, enter the following sequence of commands:

```
host1/Admin(config)# sticky ip-netmask 255.255.255.255 address
destination DEST_IP_STICKY
host1/Admin(config-sticky-ip)# serverfarm FWS-IN

host1/Admin(config)# policy-map type loadbalance first-match
L7PMAP_TO_REALS
host1/Admin(config-pmap-lb)# class class-default
host1/Admin(config-pmap-lb-c)# forward
host1/Admin(config-pmap-lb-c)# reverse-sticky DEST_IP_STICKY
```

Displaying Reverse IP Sticky Status and Statistics

Use the following **show** commands to display the state of the reverse-sticky command and reverse sticky statistics:

- **show sticky database detail**—Provides the reverse entry field that indicates the state (TRUE or FALSE) of reverse IP stickiness for each configured sticky group.
- **show stats sticky**—Provides the Total active reverse sticky entries field that displays the total number of active reverse IP sticky entries in the sticky database.
- **show service-policy route detail**—Provides the reverse sticky group field that displays the name of the sticky group configured for reverse IP stickiness.

Reverse IP Stickiness Configuration Examples

This section contains configuration examples that show how to configure reverse IP stickiness with a symmetric firewall load balancing configuration. These configuration examples correspond with the network diagram in Figure 1. The examples are as follows:

- ACE 1 Configuration
- ACE 2 Configuration

ACE 1 Configuration

```
access-list acl1 line 8 extended permit ip any any

rserver host FW1
  ip address 10.10.40.10
  inservice
rserver host FW2
  ip address 10.10.40.20
  inservice

serverfarm host FWS-OUT
  transparent
  rserver FW1
    inservice
  rserver FW2
    inservice

sticky ip-netmask 255.255.255.255 address source SOURCE_IP_STICKY
  serverfarm FWS-OUT

class-map match-all CATCH-ALL-VIP
  2 match virtual-address 0.0.0.0 0.0.0.0 any

policy-map type management first-match MGMT-POLICY
  class class-default
    permit

policy-map type loadbalance first-match LB_PMAP_TO_REALS
  class class-default
    sticky-serverfarm SOURCE_IP_STICKY
policy-map type loadbalance first-match ROUTE_PMAP
  class class-default
    forward
    reverse-sticky SOURCE_IP_STICKY
```

```
policy-map multi-match LB
  class CATCH-ALL-VIP
    loadbalance vip inservice
    loadbalance policy LB_PMAP_TO_REALS
policy-map multi-match ROUTE
  class CATCH-ALL-VIP
    loadbalance vip inservice
    loadbalance policy ROUTE_PMAP

service-policy input mgmt-policy

interface vlan 112
  description outside FW vlan
  bridge-group 15
  mac-sticky enable
  access-group input acl1
  service-policy input ROUTE
  no shutdown
interface vlan 113
  description client vlan
  bridge-group 15
  access-group input acl1
  service-policy input LB
  no shutdown

interface bvi 15
  ip address 10.10.40.2 255.255.255.0
  alias 10.10.40.3 255.255.255.0
  no shutdown

ip route 0.0.0.0 0.0.0.0 10.10.40.1
```

ACE 2 Configuration

```
access-list acl1 line 8 extended permit ip any any

rserver host FW1
  ip address 10.10.50.10
  inservice
rserver host FW2
  ip address 10.10.50.20
  inservice

serverfarm host FWS-IN
  transparent
  rserver FW1
    inservice
  rserver FW2
    inservice

sticky ip-netmask 255.255.255.255 address destination DEST_IP_STICKY
  serverfarm FWS-IN

class-map match-all CATCH_ALL_VIP
  2 match virtual-address 0.0.0.0 0.0.0.0 any

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

policy-map type loadbalance first-match L7PMAP_TO_FWS
  class class-default
    sticky-serverfarm DEST_IP_STICKY
policy-map type loadbalance first-match L7PMAP_TO_REALS
  class class-default
    forward
    reverse-sticky DEST_IP_STICKY

policy-map multi-match L4_TO_FWS
  class CATCH_ALL_VIP
    loadbalance vip inservice
    loadbalance policy L7PMAP_TO_FWS
policy-map multi-match L4_TO_REALS
  class CATCH_ALL_VIP
    loadbalance vip inservice
    loadbalance policy L7PMAP_TO_REALS

service-policy input mgmt-policy

interface vlan 21
  ip address 21.1.1.1 255.255.255.0
  access-group input acl1
  service-policy input L4_TO_FWS
  no shutdown
interface vlan 111
  description inside FW vlan
  ip address 10.10.50.1 255.255.255.0
  mac-sticky enable
  access-group input acl1
  service-policy input L4_TO_REALS
  no shutdown
```

Configuring an SLB Traffic Policy for Stickiness

After you configure the parameters that control a specific type of stickiness in a sticky group, you must create a Layer 3 and Layer 4 traffic policy and a Layer 7 traffic policy. For example, to configure an SLB policy for HTTP header stickiness, perform the following steps:

Step 1 Configure a Layer 7 class map and Layer 7 policy map, and then associate the class map with the policy map.

```
host1/Admin(config)# class-map type http loadbalance match-any
L7SLBCLASS
host1/Admin(config-cmap-http-lb)# match http 1 -value field=stream
host1/Admin(config-cmap-http-lb)# exit
host1/Admin(config)# policy-map type loadbalance first-match
L7SLBPOLICY
host1/Admin(config-pmap-lb)# class L7SLBCLASS
host1/Admin(config-pmap-lb-c)#
```

Step 2 Associate the sticky group as an action in the Layer 7 policy map.

```
host1/Admin(config-pmap-lb-c)# sticky-serverfarm STICKY_GROUP1
```

Step 3 Configure a Layer 7 HTTP parameter map. Configure parameters as necessary for your application. For details about configuring an HTTP parameter map, see the "Configuring an HTTP Parameter Map" section.

```
host1/Admin(config)# parameter-map type http HTTP_PARAM_MAP
host1/Admin(config-parammap-http)# set header-maxparse-length 8192
host1/Admin(config-parammap-http)# length-exceed continue
host1/Admin(config-parammap-http)# persistence-rebalance
host1/Admin(config-parammap-http)# exit
```

Step 4 Configure a Layer 3 and Layer 4 class map and policy map, and then associate the class map with the policy map.

```
host1/Admin(config)# class-map L4VIPCLASS
host1/Admin(config-cmap)# match virtual-address 192.168.1.10 tcp eq 80
host1/Admin(config-cmap) exit
host1/Admin(config)# policy-map multi-match L4POLICY
host1/Admin(config-pmap)# class L4VIPCLASS
host1/Admin(config-pmap-c)#
```

Step 5 Associate the Layer 7 policy map with the Layer 3 and Layer 4 policy map.

```
host1/Admin(config-pmap-c)# loadbalance policy L7SLBPOLICY
host1/Admin(config-pmap-c)# loadbalance vip inservice
```

Step 6 Associate the HTTP parameter map with the Layer 3 and Layer 4 policy map.

```
host1/Admin(config-pmap-c)# appl-parameter http advanced-options
HTTP_PARAM_MAP
host1/Admin(config-pmap-c)# exit
```

Step 7 Apply the Layer 3 and Layer 4 policy map to an interface using a service policy or globally to all interfaces in the current context.

```
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# service-policy input L4POLICY
```

or

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

For details about configuring an SLB traffic policy, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

Displaying Sticky Configurations and Statistics

This section describes the **show** commands that you can use to display sticky configurations and statistic. It contains the following topics:

- Displaying a Sticky Configuration
- Displaying Sticky Database Entries
- Displaying Sticky Statistics

Displaying a Sticky Configuration

You can display the current sticky configuration by using the **show running-config** command in EXEC mode. The syntax of this command is as follows:

show running-config sticky

The output of this command displays configured sticky groups and their attributes.

Displaying Inserted Cookie Information

To display inserted cookie information for a specified sticky group, use the **show sticky cookie-insert** command in Exec mode. This information correlates the inserted cookie, the sticky entry, and the final destination for the cookie insert configuration. The syntax of this command is:

show sticky cookie-insert group *sticky_group_name*

For the **group** *sticky_group_name* argument, enter the name of the configured sticky group.

For example, to display the inserted cookie information for the sticky group called GROUP1, enter:

```
host1/Admin# show sticky cookie-insert group GROUP1
```

Table 5-11 describes the fields in the **show sticky cookie-insert** command output.

Table 5-11 Field Descriptions for the show sticky cookie-insert Command Output

Field	Description
Cookie	Cookie-insert hash string for each real server in the associated server farm.
HashKey	64-bit hash value associated with the cookie.
rserver-instance	String containing the server-farm name, real-server name, and real-server port in the following format: *server_farm_name*/*real_server_name*:*rserver_port*

Displaying Sticky Database Entries

The ACE stores sticky entries in a sticky database based on the following categories of information:

- Client IP address
- Sticky group ID
- HTTP content value
- HTTP cookie value
- HTTP header value
- Layer 4 payload
- Real-server name
- Static sticky database entries
- Sticky group type

To display sticky database entry information, use the **show sticky database** command in EXEC mode. The syntax of this command is as follows:

show sticky database [**static**] [**client** *ip_address* | **group** *name1* | **http-content** *value1* | **http-cookie** *value2* | **http-header** *value3* | **layer4-payload** *value4* | **rserver** *name2* | **rtsp-header** *value5* | **sip-header** *value6* | **type** {**http-cookie** | **http-header** | **ip-netmask** | **radius**}]

The keywords, arguments, and options are as follows:

- **static**—(Optional) Displays static sticky database entries for one of the static sticky entry types that follow.
- **client** *ip_address*—(Optional) Displays sticky database entries for the source IP address of a client that you specify.
- **group** *name1*—(Optional) Displays sticky database entries for the sticky group name that you specify.
- **http-content** *value1*—(Optional) Displays sticky database entries for the HTTP content value that you specify.
- **http-cookie** *value2*—(Optional) Displays sticky database entries for the HTTP cookie value that you specify.
- **http-header** *value3*—(Optional) Displays sticky database entries for the HTTP header value that you specify.
- **layer4-payload** *value4*—(Optional) Displays sticky database entries for the Layer 4 payload value that you specify.
- **rserver** *name2*—(Optional) Displays sticky database entries for the real-server name that you specify.
- **rtsp-header** *value5*—(Optional) Displays sticky database entries for the RTSP header value that you specify.
- **sip-header** *value6*—(Optional) Displays sticky database entries for the SIP header value that you specify.
- **type**—(Optional) Displays sticky database entries for one of the following sticky group types:
  - **http-cookie**
  - **http-header**
  - **ip-netmask**
  - **radius**

For example, enter:

```
host1/Admin# show sticky database
sticky group : src-ip
type         : IP
timeout      : 1440         timeout-activeconns : FALSE
  sticky-entry          rserver-instance time-to-expire flags
  ---------------------+----------------------+----------+-------+
  3232236541            rs1:0                   86399      -
```

Table 5-12 describes the fields in the **show sticky database** command output.

Table 5-12 Field Descriptions for the show sticky database Command Output

Field	Description
Sticky Group	Name of the sticky group.
Type	Type of sticky group (for example, HTTP-HEADER).
Timeout	Timeout (in minutes) for the entry in the sticky table.
Timeout-Activeconns	Indication whether the **timeout activeconns** command is enabled or disabled. When enabled, this command times out sticky connections even when active connections exist. Possible values are TRUE (enabled) or FALSE (disabled).
Sticky-Entry	Hashed value of the sticky entry in the database.
Rserver-instance	Name and, optionally, port of a real server associated with the sticky group (for example, rs1:81). If no port is configured for the real server in the server farm, the port displays as 0 (for example, rs1:0).
Time-To-Expire	Time (in seconds) remaining for the sticky timeout. For sticky entries that have no expiration, the value is “never.” Static sticky entries always have a value of “never.”
Flags	For future use.
Sticky Replicate	Indication whether the ACE replicates sticky entries to the peer ACE in a redundancy configuration.

Displaying Sticky Statistics

You can display global sticky statistics for the current context by using the **show stats sticky** command in Exec mode. The syntax of this command is as follows:

show stats sticky

Table 5-13 describes the fields in the **show stats sticky** command output.

Table 5-13 Field Descriptions for the show stats sticky Command Output

Field	Description
Total Sticky Entries Reused Prior to Expiry	Total number of older sticky entries in the sticky database that the ACE needed to clear because the database was full and new sticky connections were received, even though the entries had not expired.
Total Active Sticky Entries	Total number of entries in the sticky database that currently have flows mapped to them.
Total Active Sticky Conns	Total number of sticky connections that are currently active.
Total Static Sticky Entries	Total number of configured static entries that are in the sticky database.

Clearing Sticky Statistics

You can clear all sticky statistics for the current context by using the **clear stats sticky** command in Exec mode. The syntax of this command is as follows:

clear stats sticky

For example, to clear all sticky statistics for the Admin context, enter:

```
host1/Admin# clear stats sticky
```

Note If you have redundancy configured, you need to explicitly clear sticky statistics on both the active and the standby ACEs. Clearing statistics on the active module only leaves the standby module’s statistics at the old values.

Clearing Dynamic Sticky Database Entries

You can clear dynamic sticky database entries by using the **clear sticky database** command in Exec mode. The syntax of this command is as follows:

clear sticky database {**all** | **group** *group_name*}

The keywords and arguments are as follows:

- **all**—Clears all dynamic sticky database entries in the context.
- **group** *group_name*—Clears all dynamic sticky database entries for the specified sticky group.

Note This command does not clear static sticky database entries. To clear static sticky database entries, use the **no** form of the **static** command.

For example, to clear all dynamic sticky database entries for the sticky group named GROUP1, enter:

```
host1/Admin# clear sticky database GROUP1
```

Example of a Sticky Configuration

This section provides a sample sticky configuration.

```
resource-class RC1
  limit-resource all minimum 0.00 maximum unlimited
  limit-resource sticky minimum 10.00 maximum unlimited

context Admin
  member RC1

rserver SERVER1
  address 192.168.12.15
  probe PROBE1
  inservice

rserver SERVER2
  address 192.168.12.16
  probe PROBE2
  inservice

serverfarm SFARM1
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice

sticky http-header Host GROUP4
  serverfarm SFARM1
  timeout 720
  timeout activeconns
  replicate sticky
  header offset 3000 length 1000
  static header Host rserver SERVER1 4000

class-map match-any L4CLASS
  10 match virtual-address 192.168.12.15 netmask 255.255.255.0 tcp
   port eq 80

class-map type http loadbalance match-any L7CLASS
  10 match http header Host header-value .*cisco.com

policy-map multi-match L4POLICY
  sequence interval 10
  class L4CLASS
    loadbalance policy L7POLICY
```

```
policy-map type loadbalance first-match L7POLICY
  class L7CLASS
    sticky-serverfarm GROUP4
    insert-http Host header-value *.cisco.com

interface vlan 193
  ip address 192.168.3.13 255.255.255.0
  service-policy input L4POLICY
  no shutdown

context Admin
  member RC1
```

Where to Go Next

If you want to configure firewall load balancing (FWLB), see Chapter 6, Configuring Firewall Load Balancing.

CHAPTER 6

Configuring Firewall Load Balancing

This chapter describes how to configure firewall load balancing on your ACE. Firewall load balancing allows you to scale firewall protection by distributing traffic across multiple firewalls on a per-connection basis. All packets belonging to a particular connection must go through the same firewall. The firewall then allows or denies transmission of individual packets across its interfaces.

This chapter contains the following major sections:

- Firewall Overview
- Configuring Standard Firewall Load Balancing
- Configuring Stealth Firewall Load Balancing
- Displaying FWLB Configurations
- Firewall Load-Balancing Configuration Examples
- Where to Go Next

Firewall Overview

A firewall forms a physical barrier between two parts of a network, for example, the Internet and an intranet. When a firewall accepts a packet from one side (the Internet), it sends the packet through to the other side (the intranet). A firewall can modify a packet before passing it through or send it through unaltered. When a firewall rejects a packet, it typically discards the packet and logs the discarded packet as an event.

After a session is established and a flow of packets begins, a firewall can monitor each packet in the flow or allow the flow to continue unmonitored, depending on the policies that you configure on that firewall.

This section contains the following topics:

- Firewall Types
- How the ACE Distributes Traffic to Firewalls
- Supported Firewall Configurations

Firewall Types

The two basic types of firewalls are as follows:

- Standard firewalls
- Stealth firewalls

Standard firewalls have a presence on the network. You assign IP addresses to the firewalls, which allows other devices on the network to see and address them as devices. Each firewall has an IP address on the VLANs configured on both sides of the firewall.

Stealth firewalls have no presence on the network. You do not assign IP addresses to the firewalls, which prevents other devices on the network from seeing or addressing them. Instead, you configure alias IP addresses on the VLANs on both sides of the firewall. To the network, a stealth firewall is part of the wire.

Both firewall types do the following tasks:

- Examine traffic moving in both directions (between the protected and the unprotected sides of the network)
- Accept or reject packets based on user-defined policies

How the ACE Distributes Traffic to Firewalls

The ACE load balances traffic to devices configured in server farms. These devices can be firewalls, caches, servers, or any IP-addressable object, including an alias IP address. For more information about server farms, see the "Configuring a Server Farm" section in Chapter 2, Configuring Real Servers and Server Farms. When the ACE load balances traffic to firewalls, it performs the same function that it performs when it load balances Layer 3 traffic to real servers in a server farm.

The ACE uses load-balancing algorithms or predictors to determine how to balance the traffic among the devices configured in the server farms, independent of the device type. For FWLB, we recommend that you use only the hash address source and the hash address destination predictors. Using any other predictor with FWLB may fail and block traffic, especially for applications that have separate control and data channels, for example, FTP.

For more information about load-balancing predictor methods, see the "Configuring the Server Farm Predictor Method" section in Chapter 2, Configuring Real Servers and Server Farms.

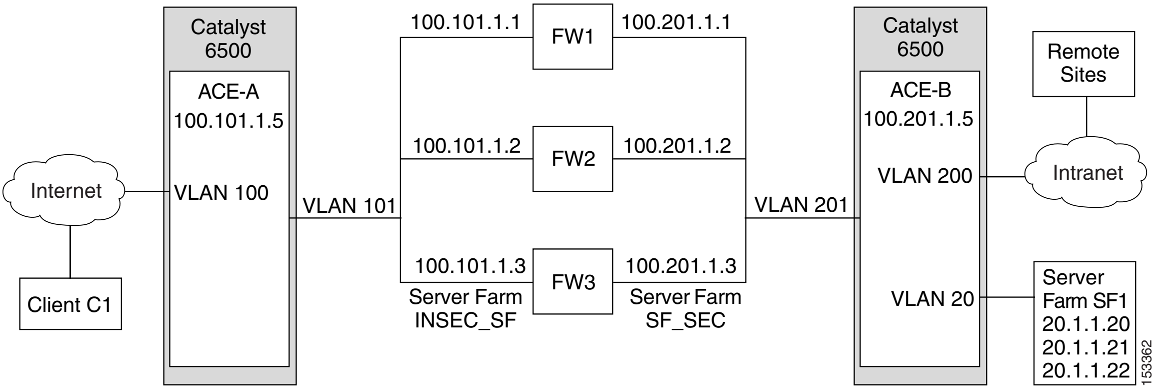
Supported Firewall Configurations

The ACE can load balance traffic to both standard and stealth firewalls.

For standard firewalls, a single ACE or a pair of ACEs in two different Catalyst 6500 series switches or two different Cisco 7600 series routers load balances traffic among firewalls that contain unique IP addresses in a manner similar to how the ACE load balances traffic among servers in a server farm (see Figure 6-1).

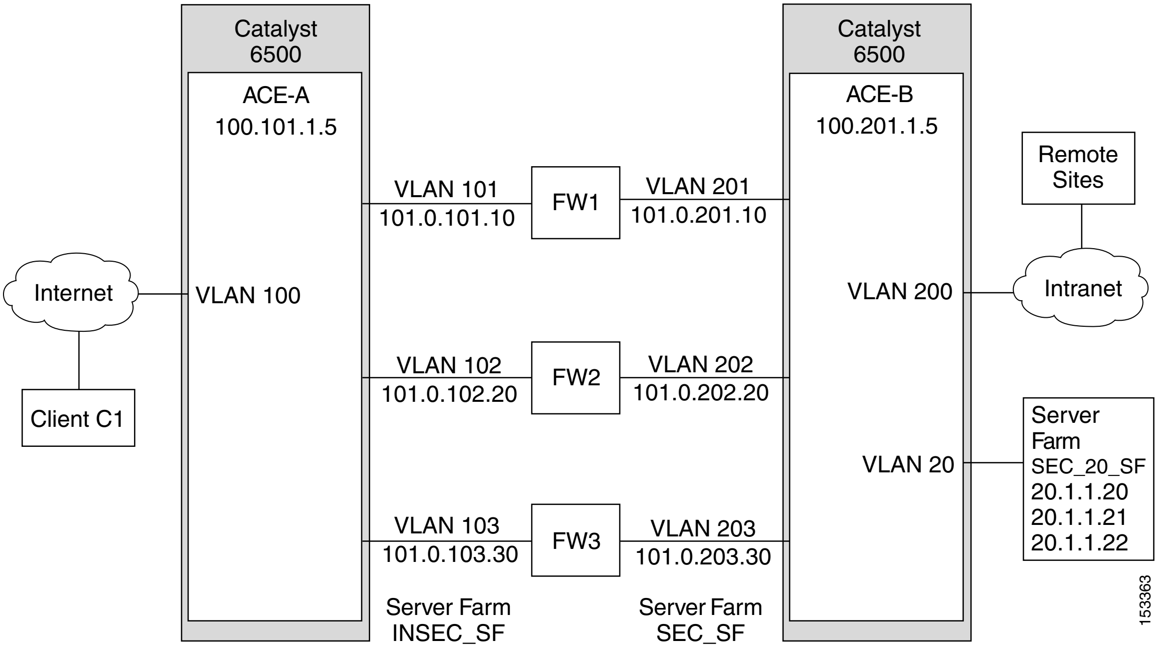
In Figure 6-2, traffic moves through the firewalls and the firewalls filter the traffic in both directions. For traffic that originates on the Internet, ACE A load balances the traffic to the firewalls in the SF_INSEC server farm. For traffic that originates on the intranet, ACE B load balances the traffic to the firewalls in server farm SF_SEC. You configure the firewalls so that the return traffic flows through the same firewall as the original traffic.

Figure 6-1 Standard Firewall Configuration



For stealth firewalls, an ACE load balances traffic among interfaces with unique alias IP addresses in different ACEs that provides paths through the firewalls (see Figure 6-2). You configure a stealth firewall so that all traffic moving in both directions across a particular VLAN moves through the same firewall.

Figure 6-2 Stealth Firewall Configuration (Dual ACEs Only)



In Figure 6-2, traffic flows through the firewalls and the firewalls filter the traffic in both directions. On the path to the intranet, ACE A balances traffic across VLANs 101, 102, and 103 through the firewalls to ACE B. On the path to the Internet, ACE B balances traffic across VLANs 201, 202, and 203 through the firewalls to ACE A. Each ACE uses the alias IP addresses configured on the other ACE as targets for the load-balancing process.

Configuring Standard Firewall Load Balancing

This section describes how to configure firewall load balancing for standard firewalls. It contains the following topics:

- Standard FWLB Configuration Overview
- Standard FWLB Configuration Quick Starts

Note For information about configuring the firewall devices in your network, refer to the documentation included with your firewall product.

Standard FWLB Configuration Overview

In this standard FWLB configuration example (see Figure 6-2), you configure three firewalls (FW1, FW2, and FW3) between two ACEs (ACE A and ACE B). (You can also configure standard FWLB using a single ACE.) Traffic enters and exits the firewalls through shared VLANs on either side of the firewalls (VLAN 101 on the insecure side and VLAN 201 on the secure side). You assign unique IP addresses to each firewall configured as a real server in a server farm on each shared VLAN.

Other VLANs provide connectivity to the following:

- Internet (VLAN 100)
- Internal network (VLAN 200)
- Internal server farm (VLAN 20)

Standard FWLB Configuration Quick Starts

This section provides quick start tables that include step-by-step instructions for configuring standard FWLB on two ACEs in separate Catalyst 6500 series switches. You can also configure standard FWLB on a single ACE. This section includes the following topics:

- Standard FWLB Configuration Quick Start for ACE A
- Standard FWLB Configuration Quick Start for ACE B

Standard FWLB Configuration Quick Start for ACE A

Table 6-1 provides a quick overview of the steps required to configure standard FWLB on ACE A (see Figure 6-2). Each step includes the CLI command required to complete the task.

Table 6-1 Standard FWLB Configuration Quick Start for ACE A

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, change to, or directly log in to, the correct context.

```
host1/Admin# changeto C1
host1/C1#
```

The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.

```
host1/Admin# config
Enter configuration commands, one per line. End with CNTL/Z
host1/Admin(config)#
```

3. Configure an access control list (ACL) to allow traffic. You can modify the ACL to suit your application needs. For more information about configuring ACLs, see the *Cisco Application Control Engine Module Security Configuration Guide*.

```
host1/Admin(config)# access-list ACL1 line 10 extended permit ip
any any
host1/Admin(config-acl)# exit
```

4. Configure three real servers to represent the insecure side of the firewalls on VLAN 101. For more information about configuring real servers, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# rserver FW_INSEC_1
host1/Admin(config-rserver-host)# ip address 100.101.1.1
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_INSEC_2
host1/Admin(config-rserver-host)# ip address 100.101.1.2
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_INSEC_3
host1/Admin(config-rserver-host)# ip address 100.101.1.3
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
```

Table 6-1 Standard FWLB Configuration Quick Start for ACE A

Task and Command Example

5. Configure a server farm to handle connections originating from the insecure side of the firewalls (Internet). The ACE selects a firewall based on source IP address using the hash address source predictor. For more information about configuring server farms, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# serverfarm SF_INSEC
host1/Admin(config-sfarm-host)# transparent
host1/Admin(config-sfarm-host)# predictor hash address source
255.255.255.255
host1/Admin(config-sfarm-host)# rserver FW_INSEC_1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver FW_INSEC_2
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver FW_INSEC_3
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# exit
```

6. Configure a Layer 7 load-balancing policy map to balance requests to server farm SF-INSEC. Associate the default class map and the SF-INSEC server farm with the policy map. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# policy-map type loadbalance first-match
LB_FW_INSEC
host1/Admin(config-pmap-lb)# class class-default
host1/Admin(config-pmap-lb-c)# serverfarm SF_INSEC
host1/Admin(config-pmap-lb-c)# exit
host1/Admin(config-pmap-lb)# exit
```

7. Configure a Layer 3 class map to classify traffic from the Internet that matches VIP address 200.1.1.1 on VLAN 100 on the insecure side of the firewalls. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# class-map match-any FW_VIP
host1/Admin(config-cmap)# match virtual-address 200.1.1.1
255.255.0.0 any
host1/Admin(config-cmap)# exit
```

Table 6-1 Standard FWLB Configuration Quick Start for ACE A

Task and Command Example

8. Configure a Layer 3 policy map and associate the Layer 3 class map and the Layer 7 policy map with it to complete the traffic policy configuration. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# policy-map multi-match POL_INSEC
host1/Admin(config-pmap)# class FW_VIP
host1/Admin(config-pmap-c)# loadbalance vip inservice
host1/Admin(config-pmap-c)# loadbalance policy LB_FW_INSEC
host1/Admin(config-pmap-c)# exit
host1/Admin(config-pmap)# exit
```

9. Configure an interface that the ACE uses to receive traffic from the Internet and to send traffic that originates from the intranet to the Internet. Apply the ACL (ACL1) and the Layer 3 policy (POL_INSEC) to the interface. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# ip address 100.100.1.100 255.255.0.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# service-policy input POL_INSEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

10. Configure an interface on the insecure side of the firewalls. The ACE uses this interface to load balance traffic to the firewalls and to receive traffic that originates from the intranet. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 101
host1/Admin(config-if)# ip address 100.101.1.101 255.255.0.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# mac-sticky enable
host1/Admin(config-if)# service-policy input POL_INSEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# Ctrl-z
```

Table 6-1 Standard FWLB Configuration Quick Start for ACE A

Task and Command Example

11. Use the following **show** commands to verify your FWLB configuration:
`host1/Admin# show running-config access-list`
`host1/Admin# show running-config class-map`
`host1/Admin# show running-config interface`
`host1/Admin# show running-config policy-map`
`host1/Admin# show running-config rserver`
`host1/Admin# show running-config serverfarm`

12. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Standard FWLB Configuration Quick Start for ACE B

Table 6-2 provides a quick overview of the steps required to configure standard FWLB on ACE B (see Figure 6-2). Each step includes the CLI command and a reference to the procedure required to complete the task.

Table 6-2 Standard FWLB Configuration Quick Start for ACE B

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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.
`host1/Admin# config`
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

Table 6-2 Standard FWLB Configuration Quick Start for ACE B

Task and Command Example

3. Configure an ACL to allow traffic. You can modify the ACL to suit your application needs. For more information about configuring ACLs, see the *Cisco Application Control Engine Module Security Configuration Guide*.

```
host1/Admin(config)# access-list ACL1 line 10 extended permit ip
any any
host1/Admin(config-acl)# exit
```

4. Configure three real servers to represent the secure side of the firewalls on VLAN 201. For more information about configuring real servers, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# rserver FW_SEC_1
host1/Admin(config-rserver-host)# ip address 100.201.1.1
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_SEC_2
host1/Admin(config-rserver-host)# ip address 100.201.1.2
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_SEC_3
host1/Admin(config-rserver-host)# ip address 100.201.1.3
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
```

Table 6-2 Standard FWLB Configuration Quick Start for ACE B

Task and Command Example

5. Configure a server farm to handle connections that originate from the secure side of the firewall (intranet). In this case, the ACE selects a firewall based on the destination IP address using the hash address destination predictor. This predictor allows the ACE to select the same firewall for return flows and buddy connections. For example, you want both the FTP control and data channels to pass through the same firewall. For more information about configuring server farms, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# serverfarm SF_SEC
host1/Admin(config-sfarm-host)# transparent
host1/Admin(config-sfarm-host)# predictor hash address
destination 255.255.255.255
host1/Admin(config-sfarm-host)# rserver FW_SEC_1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver FW_SEC_2
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver FW_SEC_3
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host)# exit
```

6. Configure two real servers to load balance content on VLAN 20 on the secure side of the firewall. For more information about configuring server farms, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# rserver REAL1
host1/Admin(config-rserver-host)# ip address 20.1.1.1
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver REAL2
host1/Admin(config-rserver-host)# ip address 20.1.1.2
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver REAL3
host1/Admin(config-rserver-host)# ip address 20.1.1.3
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
```

Table 6-2 Standard FWLB Configuration Quick Start for ACE B

Task and Command Example

7. Configure a standard server farm of HTTP servers. For more information about configuring server farms, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# serverfarm SEC_20_SF
host1/Admin(config-sfarm-host)# rserver REAL1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver REAL2
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver REAL3
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# exit
```

8. Configure a Layer 7 policy map that load balances traffic to the HTTP server farm on VLAN 20 using the default class map. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# policy-map type loadbalance first-match
SEC_20_LB
host1/Admin(config-pmap-lb)# class class-default
host1/Admin(config-pmap-lb-c)# serverfarm SEC_20_SF
host1/Admin(config-pmap-lb-c)# exit
host1/Admin(config-pmap-lb)# exit
```

9. Configure a Layer 3 class map to classify traffic destined to the virtual IP address 200.1.1.1 configured on VLAN 201. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# class-map match-any SEC_20_VS
host1/Admin(config-cmap)# match virtual-address 200.1.1.1
255.255.0.0 any
host1/Admin(config-cmap)# exit
```

Table 6-2 Standard FWLB Configuration Quick Start for ACE B

Task and Command Example

10. Configure a Layer 3 policy map and associate the Layer 3 class map (SEC_20_VS) and the Layer 7 policy map (SEC_20_LB) with it. This step completes the policy that load balances traffic to the HTTP servers on VLAN 20. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.
`host1/Admin(config)# policy-map multi-match POL_SEC_20`
`host1/Admin(config-pmap)# class SEC_20_VS`
`host1/Admin(config-pmap-c)# loadbalance vip inservice`
`host1/Admin(config-pmap-c)# loadbalance policy SEC_20_LB`

11. Configure a Layer 7 policy map to load balance traffic that originates from either VLAN 200 or VLAN 20 and is destined for the Internet to the secure side of the firewalls on VLAN 201. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.
`host1/Admin(config)# policy-map type loadbalance first-match`
`LB_FW_SEC`
`host1/Admin(config-pmap-lb)# class class-default`
`host1/Admin(config-pmap-lb-c)# serverfarm SF_SEC`
`host1/Admin(config-pmap-lb-c)# exit`
`host1/Admin(config-pmap-lb)# exit`

12. Configure a Layer 3 class map to classify all traffic that originates on the secure side of the firewalls and is destined for the Internet. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.
`host1/Admin(config)# class-map match-any FW_SEC_VIP`
`host1/Admin(config-cmap)# match virtual-address 0.0.0.0 0.0.0.0`
`any`
`host1/Admin(config-cmap)# exit`

Table 6-2 Standard FWLB Configuration Quick Start for ACE B

Task and Command Example

13. Configure a Layer 3 policy map and associate the Layer 7 policy map (LB_FW_SEC) and the Layer 3 class map (FW_SEC_VIP) with it. Enable the VIP for load balancing. This step completes the policy that load balances any request that originates on the secure side of the firewalls and is destined for the Internet. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# policy-map multi-match POL_SEC
host1/Admin(config-pmap)# class FW_SEC_VIP
host1/Admin(config-pmap-c)# loadbalance vip inservice
host1/Admin(config-pmap-c)# loadbalance LB_FW_SEC
host1/Admin(config-pmap-c)# exit
host1/Admin(config-pmap)# exit
```

14. Configure an interface on the secure side of the firewalls for traffic that originates from the Internet and is passing through the firewalls. The ACE uses this interface to catch traffic from the firewalls, load balance it to the HTTP server farm, and route it to the remote host. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 201
host1/Admin(config-if)# ip address 100.201.1.201 255.255.0.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# mac-sticky enable
host1/Admin(config-if)# service-policy input POL_SEC_20
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

15. Configure an interface on the secure side of the firewalls for traffic that originates from the HTTP server farm on VLAN 20. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 20
host1/Admin(config-if)# ip address 20.1.1.20 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# service-policy input POL_SEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

Table 6-2 Standard FWLB Configuration Quick Start for ACE B

Task and Command Example

16. Configure an interface on the secure side of the firewalls for traffic that originates from the remote host on VLAN 200. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 200
host1/Admin(config-if)# ip address 200.1.1.200 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# service-policy input POL_SEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# Ctrl-z
```

17. Use the following **show** commands to verify your FWLB configuration:

```
host1/Admin# show running-config access-list
host1/Admin# show running-config class-map
host1/Admin# show running-config interface
host1/Admin# show running-config policy-map
host1/Admin# show running-config rserver
host1/Admin# show running-config serverfarm
```

18. (Optional) Save your configuration changes to flash memory.

```
host1/Admin# copy running-config startup-config
```

Configuring Stealth Firewall Load Balancing

This section describes how to configure stealth FWLB. It contains the following topics:

- Stealth Firewall Load-Balancing Configuration Overview
- Stealth Firewall Load-Balancing Configuration Quick Starts

Note For information about configuring the firewall devices in your network, refer to the documentation included with your firewall product.

Stealth Firewall Load-Balancing Configuration Overview

Note In a stealth FWLB configuration, you must configure two ACEs, each in a separate Catalyst 6500 series switch or each in a separate Cisco 7600 series router.

In this stealth FWLB configuration example (see Figure 6-2), ACE A and ACE B load balance traffic through three firewalls. Each firewall configured as a real server in a server farm connects to two different VLANs, one on the insecure side and one on the secure side of the firewall. Stealth firewalls do not have IP addresses on VLANs. Instead, you configure alias IP addresses on each ACE interface to which a firewall connects. The ACEs use the alias IP addresses to direct traffic to the correct firewall.

On the path from the Internet to the intranet, traffic enters the insecure side of the firewalls through separate VLANs (VLAN 101,VLAN 102, and VLAN 103) and exits the secure side of the firewalls through separate VLANs (VLAN 201, VLAN 202, and VLAN 203). On the path from the intranet to the Internet, the flow is reversed. Other VLANs provide connectivity to the following locations:

- Internet (VLAN 100)
- Remote host (VLAN 200)
- Intranet server farm (VLAN 20)

Stealth Firewall Load-Balancing Configuration Quick Starts

This section provides quick start tables that include step-by-step instructions about how to configure stealth FWLB on two separate ACE modules. This section includes the following topics:

- Stealth FWLB Configuration Quick Start for ACE A
- Stealth FWLB Configuration Quick Start for ACE B

Stealth FWLB Configuration Quick Start for ACE A

Table 6-3 provides a quick overview of the steps required to configure stealth FWLB on ACE A (insecure side). Each step includes the CLI command required to complete the task.

Table 6-3 Stealth FWLB Configuration Quick Start for ACE A

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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.
`host1/Admin# config`
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

3. Configure an ACL to allow traffic to the ACE. You can modify the ACL to suit your application needs. For more information about configuring ACLs, see the *Cisco Application Control Engine Module Security Configuration Guide*
`host1/Admin(config)# access-list ACL1 line 10 extended permit ip any any`
`host1/Admin(config-acl)# exit`

Table 6-3 Stealth FWLB Configuration Quick Start for ACE A (continued)

Task and Command Example

4. Configure three real servers to represent the insecure side of the firewalls on VLANs 101, 102, and 103. For more information about configuring real servers, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# rserver FW_INSEC_1
host1/Admin(config-rserver-host)# ip address 101.0.201.100
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_INSEC_2
host1/Admin(config-rserver-host)# ip address 101.0.202.100
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_INSEC_3
host1/Admin(config-rserver-host)# ip address 101.0.203.100
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
```

5. Configure a server farm to handle connections originating from the insecure side of the firewalls (Internet). The ACE selects a firewall based on source IP address using the hash address source predictor. For more information about configuring server farms, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# serverfarm SF_INSEC
host1/Admin(config-sfarm-host)# transparent
host1/Admin(config-sfarm-host)# predictor hash address source
255.255.255.255
host1/Admin(config-sfarm-host)# rserver FW_INSEC_1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver FW_INSEC_2
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver FW_INSEC_3
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# exit
```

Table 6-3 Stealth FWLB Configuration Quick Start for ACE A (continued)

Task and Command Example

6. Configure a Layer 7 load-balancing policy map to forward packets received from the firewall to the Internet. Associate the default class map with the policy map. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# policy-map type loadbalance first-match FORWARD_FW_INSEC
host1/Admin(config-pmap-lb)# class class-default
host1/Admin(config-pmap-lb-c)# forward
host1/Admin(config-pmap-lb-c)# exit
host1/Admin(config-pmap-lb)# exit
```

7. Configure a Layer 3 class map to classify traffic from the firewalls that matches any VIP address, netmask, and protocol on VLANs 101, 102, and 103 on the insecure side of the firewalls.

```
host1/Admin(config)# class-map match-any FORWARD_VIP
host1/Admin(config-cmap)# match virtual-address 0.0.0.0 0.0.0.0 any
host1/Admin(config-cmap)# exit
```

8. Configure a Layer 3 policy map and associate the Layer 3 forwarding class map (FORWARD_VIP) and the Layer 7 forwarding policy map (FORWARD_FW_INSEC) with it to complete the forwarding policy configuration.

```
host1/Admin(config)# policy-map multi-match FORWARD_INSEC
host1/Admin(config-pmap)# class FORWARD_VIP
host1/Admin(config-pmap-c)# loadbalance vip inservice
host1/Admin(config-pmap-c)# loadbalance policy FORWARD_FW_INSEC
host1/Admin(config-pmap-c)# exit
host1/Admin(config-pmap)# exit
```

9. Configure a Layer 7 load-balancing policy map to balance requests from the Internet to server farm SF-INSEC. Associate the default class map and the SF-INSEC server farm with the policy map.

```
host1/Admin(config)# policy-map type loadbalance first-match LB-FW-INSEC
host1/Admin(config-pmap-lb)# class class-default
host1/Admin(config-pmap-lb-c)# serverfarm SF_INSEC
host1/Admin(config-pmap-lb-c)# exit
host1/Admin(config-pmap-lb)# exit
```

Table 6-3 Stealth FWLB Configuration Quick Start for ACE A (continued)

Task and Command Example

10. Configure a Layer 3 class map to classify traffic from the Internet that matches VIP address 200.1.1.1, netmask 255.255.0.0, and any protocol on VLAN 100 on the insecure side of the firewalls.

```
host1/Admin(config)# class-map match-any FW_VIP
host1/Admin(config-cmap)# match virtual-address 200.1.1.1
255.255.0.0 any
host1/Admin(config-cmap)# exit
```

11. Configure a Layer 3 policy map and associate the Layer 3 class map (FW-VIP) and the Layer 7 policy map (LB_FW_INSEC) with it to complete the load-balancing policy configuration.

```
host1/Admin(config)# policy-map multi-match POL_INSEC
host1/Admin(config-pmap)# class FW_VIP
host1/Admin(config-pmap-c)# loadbalance vip inservice
host1/Admin(config-pmap-c)# loadbalance policy LB_FW_INSEC
host1/Admin(config-pmap-c)# exit
host1/Admin(config-pmap)# exit
```

12. Configure an interface that the ACE uses to receive traffic from the Internet and load balance the traffic to the insecure side of the firewall. Apply the ACL (ACL1) and the Layer 3 policy (POL_INSEC) to the interface. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 100
host1/Admin(config-if)# ip address 100.100.1.100 255.255.0.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# service-policy input POL_INSEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

Table 6-3 Stealth FWLB Configuration Quick Start for ACE A (continued)

Task and Command Example

13. Configure an interface on the insecure side of the firewalls that ACE A uses to load balance traffic to FW1 and to receive traffic that originates from the intranet. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 101
host1/Admin(config-if)# ip address 101.0.101.10 255.255.255.0
host1/Admin(config-if)# alias 101.0.101.100 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# mac-sticky enable
host1/Admin(config-if)# service-policy input FORWARD_INSEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

14. Configure an interface on the insecure side of the firewalls that ACE A uses to load balance traffic to FW2 and to receive traffic that originates from the intranet. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 102
host1/Admin(config-if)# ip address 101.0.102.20 255.255.255.0
host1/Admin(config-if)# alias 101.0.102.100 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# mac-sticky enable
host1/Admin(config-if)# service-policy input FORWARD_INSEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

15. Configure an interface on the insecure side of the firewalls that ACE A uses to load balance traffic to the FW3 and to receive traffic that originates from the intranet. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 103
host1/Admin(config-if)# ip address 101.0.103.30 255.255.255.0
host1/Admin(config-if)# alias 101.0.103.100 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# mac-sticky enable
host1/Admin(config-if)# service-policy input FORWARD_INSEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# Ctrl-z
```

Table 6-3 Stealth FWLB Configuration Quick Start for ACE A (continued)

Task and Command Example

16. Use the following **show** commands to verify your FWLB configuration:
`host1/Admin# show running-config access-list`
`host1/Admin# show running-config class-map`
`host1/Admin# show running-config interface`
`host1/Admin# show running-config policy-map`
`host1/Admin# show running-config rserver`
`host1/Admin# show running-config serverfarm`

17. (Optional) Save your configuration changes to flash memory. `host1/Admin# copy running-config startup-config`

Stealth FWLB Configuration Quick Start for ACE B

Table 6-4 provides a quick overview of the steps required to configure stealth FWLB on ACE B (secure side). Each step includes the CLI command required to complete the task.

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B

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, change to, or directly log in to, the correct context.
`host1/Admin# changeto C1`
`host1/C1#`
The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.
`host1/Admin# config`
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B (continued)

Task and Command Example

3. Configure an ACL to allow traffic to the ACE. You can modify the ACL to suit your application needs. For more information about configuring ACLs, see the *Cisco Application Control Engine Module Security Configuration Guide*.

```
host1/Admin(config)# access-list ACL1 line 10 extended permit ip
any any
host1/Admin(config-acl)# exit
```

4. Configure three real servers to represent the secure side of the firewalls on VLANs 201, 202, and 203. For more information about configuring real servers, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# rserver FW_SEC_1
host1/Admin(config-rserver-host)# ip address 101.0.101.100
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_SEC_2
host1/Admin(config-rserver-host)# ip address 101.0.102.100
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver FW_SEC_3
host1/Admin(config-rserver-host)# ip address 101.0.103.100
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
```

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B (continued)

Task and Command Example

5. Configure a server farm to handle connections that originate from the secure side of the firewall (intranet). In this case, the ACE selects a firewall based on the destination IP address using the hash address destination predictor. This predictor allows the ACE to select the same firewall for return flows and buddy connections. For example, you want both the FTP control and data channels to pass through the same firewall. For more information about configuring server farms, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# serverfarm SF_SEC
host1/Admin(config-sfarm-host)# transparent
host1/Admin(config-sfarm-host)# predictor hash address
destination 255.255.255.255
host1/Admin(config-sfarm-host)# rserver FW_SEC_1
host1/Admin(config-sfarm-host)# inservice
host1/Admin(config-sfarm-host)# rserver FW_SEC_2
host1/Admin(config-sfarm-host)# inservice
host1/Admin(config-sfarm-host)# rserver FW_SEC_3
host1/Admin(config-sfarm-host)# inservice
host1/Admin(config-sfarm-host)# exit
```

6. Configure three real servers to load balance the content on VLAN 20 on the secure side of the firewall. For more information about configuring real servers, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# rserver REAL1
host1/Admin(config-rserver-host)# ip address 20.1.1.1
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver REAL2
host1/Admin(config-rserver-host)# ip address 20.1.1.2
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit

host1/Admin(config)# rserver REAL3
host1/Admin(config-rserver-host)# ip address 20.1.1.3
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
```

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B (continued)

Task and Command Example

7. Configure a standard server farm of HTTP servers to load balance requests to the HTTP servers on VLAN 20. For more information about configuring server farms, see Chapter 2, Configuring Real Servers and Server Farms.

```
host1/Admin(config)# serverfarm SEC_20_SF
host1/Admin(config-sfarm-host)# rserver REAL1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver REAL2
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver REAL3
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# exit
```

8. Configure a Layer 7 policy map that load balances traffic to the HTTP server farm on VLAN 20 using the default class map. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# policy-map type loadbalance first-match
SEC_20_LB
host1/Admin(config-pmap-lb)# class class-default
host1/Admin(config-pmap-lb-c)# serverfarm SEC_20_SF
host1/Admin(config-pmap-lb-c)# exit
host1/Admin(config-pmap-lb)# exit
```

9. Configure a Layer 3 class map to classify traffic destined to the virtual IP address 200.1.1.1 on VLANs 201, 202, and 203. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.

```
host1/Admin(config)# class-map match-any SEC_20_VS
host1/Admin(config-cmap)# match virtual-address 200.1.1.1
255.255.0.0 any
host1/Admin(config-cmap)# exit
```

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B (continued)

Task and Command Example

10. Configure a Layer 3 policy map and associate the Layer 3 class map (SEC_20_VS) and the Layer 7 policy map (SEC_20_LB) with it. This step completes the policy that load balances traffic to the HTTP servers on VLAN 20. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.
`host1/Admin(config)# policy-map multi-match POL_SEC_20`
`host1/Admin(config-pmap)# class SEC_20_VS`
`host1/Admin(config-pmap-c)# loadbalance vip inservice`
`host1/Admin(config-pmap-c)# loadbalance policy SEC_20_LB`
`host1/Admin(config-pmap-c)# exit`
`host1/Admin(config-pmap)# exit`

11. Configure a Layer 7 policy map to load balance requests that originate from either VLAN 200 or VLAN 20 and are destined for the Internet to the secure side of the firewalls on VLAN 201. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.
`host1/Admin(config)# policy-map type loadbalance first-match LB_FW_SEC`
`host1/Admin(config-pmap-lb)# class class-default`
`host1/Admin(config-pmap-lb-c)# serverfarm SF_SEC`
`host1/Admin(config-pmap-lb-c)# exit`
`host1/Admin(config-pmap-lb)# exit`

12. Configure a Layer 3 class map to classify all traffic with any IP address, netmask, and protocol originating on the secure side of the firewalls. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.
`host1/Admin(config)# class-map match-any FW_SEC_VIP`
`host1/Admin(config-cmap)# match virtual-address 0.0.0.0 0.0.0.0 any`
`host1/Admin(config-cmap)# exit`

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B (continued)

Task and Command Example

13. Configure a Layer 3 policy map and associate the Layer 7 policy map (LB_FW_SEC) and the Layer 3 class map (FW_SEC_VIP) with it. Enable the VIP for load balancing. This step completes the policy that load balances any request that originates on the secure side of the firewalls and destined for the Internet. For more information about configuring traffic policies for SLB, see Chapter 3, Configuring Traffic Policies for Server Load Balancing.
`host1/Admin(config)# policy-map multi-match POL_SEC`
`host1/Admin(config-pmap)# class FW_SEC_VIP`
`host1/Admin(config-pmap-c)# loadbalance vip inservice`
`host1/Admin(config-pmap-c)# loadbalance policy LB_FW_SEC`
`host1/Admin(config-pmap-c)# exit`
`host1/Admin(config-pmap)# exit`

14. Configure an interface on the secure side of the firewalls that the ACE uses to send traffic to FW1 from the intranet and to receive traffic that originates from the Internet and passing through the firewall. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.
`host1/Admin(config)# interface vlan 201`
`host1/Admin(config-if)# ip address 101.0.201.10 255.255.255.0`
`host1/Admin(config-if)# alias 101.0.201.100 255.255.255.0`
`host1/Admin(config-if)# access-group input ACL1`
`host1/Admin(config-if)# mac-sticky enable`
`host1/Admin(config-if)# service-policy input POL_SEC_20`
`host1/Admin(config-if)# no shutdown`
`host1/Admin(config-if)# exit`

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B (continued)

Task and Command Example

15. Configure an interface on the secure side of the firewalls that the ACE uses to send traffic to FW2 from the intranet and to receive traffic that originates from the Internet and is passing through the firewall. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 202
host1/Admin(config-if)# ip address 101.0.202.20 255.255.255.0
host1/Admin(config-if)# alias 101.0.202.100 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# mac-sticky enable
host1/Admin(config-if)# service-policy input POL_SEC_20
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

16. Configure an interface on the insecure side of the firewall that the ACE uses to send traffic to FW3 from the intranet and to receive traffic that originates from the Internet and is passing through the firewall. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 203
host1/Admin(config-if)# ip address 101.0.203.30 255.255.255.0
host1/Admin(config-if)# alias 101.0.203.100 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# mac-sticky enable
host1/Admin(config-if)# service-policy input POL_SEC_20
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

17. Configure an interface that the ACE uses to receive traffic that originates from the remote host on VLAN 200 and is destined to the Internet. Apply the ACL (ACL1) and the Layer 3 policy map (POL_SEC) to the interface. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 200
host1/Admin(config-if)# ip address 200.1.1.200 255.255.255.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# service-policy input POL_SEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
```

Table 6-4 Stealth FWLB Configuration Quick Start for ACE B (continued)

Task and Command Example

18. Configure an interface that the ACE uses to receive traffic that originates from the HTTP server farm on VLAN 20 and is destined to the Internet. Apply the ACL (ACL1) and the Layer 3 policy map (POL_SEC) to the interface. For more information about configuring interfaces, see the *Cisco Application Control Engine Module Routing and Bridging Configuration Guide*.

```
host1/Admin(config)# interface vlan 20
host1/Admin(config-if)# ip address 20.100.1.100 255.255.0.0
host1/Admin(config-if)# access-group input ACL1
host1/Admin(config-if)# service-policy input POL_SEC
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# Ctrl-z
```

19. Use the following **show** commands to verify your FWLB configuration:

```
host1/Admin# show running-config access-list
host1/Admin# show running-config class-map
host1/Admin# show running-config interface
host1/Admin# show running-config policy-map
host1/Admin# show running-config rserver
host1/Admin# show running-config serverfarm
```

20. (Optional) Save your configuration changes to flash memory.

```
host1/Admin# copy running-config startup-config
```

Displaying FWLB Configurations

You can display your entire running configuration by using the **show running-config** command in Exec mode. The syntax of this command is as follows:

show running-config

To display sections of the running-config that pertain to FWLB, use the following commands in Exec mode:

- **show running-config access-list**
- **show running-config class-map**
- **show running-config interface**
- **show running-config policy-map**
- **show running-config rserver**
- **show running-config serverfarm**

Firewall Load-Balancing Configuration Examples

This section provides examples of standard and stealth FWLB configurations. It contains the following topics:

- Example of a Standard Firewall Load-Balancing Configuration
- Example of a Stealth Firewall Configuration

Example of a Standard Firewall Load-Balancing Configuration

The following example shows those portions of the running configuration that pertain to standard FWLB. The configuration is based on two ACE modules each in a separate Catalyst 6500 series switch with the firewalls situated between them (see Figure 6-2). You can also configure standard FWLB using a single ACE.

ACE A Configuration—Standard Firewall Load Balancing

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

```
rserver host FW_INSEC_1
  ip address 100.101.1.1
  inservice
rserver host FW_INSEC_2
  ip address 100.101.1.2
  inservice
rserver host FW_INSEC_3
  ip address 100.101.1.3
  inservice

serverfarm INSEC_SF
  transparent
  predictor hash address source 255.255.255.255
  rserver FW_INSEC_1
    inservice
  rserver FW_INSEC_2
    inservice
  rserver FW_INSEC_3
    inservice

class-map match-any FW_VIP
  10 match virtual-address 200.1.1.1 255.255.0.0 any
policy-map type loadbalance first-match LB_FW_INSEC
  class class-default
    serverfarm INSEC_SF
policy-map multi-match POL_INSEC
  class FW_VIP
    loadbalance vip inservice
    loadbalance policy LB_FW_INSEC

interface vlan 100
  ip addr 100.100.1.100 255.255.0.0
  access-group input ACL1
  service-policy input POL_INSEC
  no shutdown
interface vlan 101
  ip addr 100.101.1.101 255.255.0.0
  access-group input ACL1
  mac-sticky enable
  service-policy input POL_INSEC
  no shutdown
```

ACE B Configuration—Standard Firewall Load Balancing

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

```
rserver FW_SEC_1
  ip address 100.201.1.1
  inservice
rserver FW_SEC_2
  ip address 100.201.1.2
  inservice
rserver FW_SEC_3
  ip address  100.201.1.3
  inservice

rserver REAL1
  ip address 20.1.1.1
  inservice
rserver REAL2
  ip address 20.1.1.2
  inservice
rserver REAL3
  ip address 20.1.1.3
  inservice

serverfarm SEC_SF
  predictor hash address destination 255.255.255.255
  transparent
  rserver FW_SEC_1
    inservice
  rserver FW_SEC_2
    inservice
  rserver FW_SEC_3
    inservice

serverfarm SEC_20_SF
  rserver REAL1
    inservice
  rserver REAL2
    inservice
  rserver REAL3
    inservice

class-map match-any SEC_20_VS
  10 match virtual-address 200.1.1.1 255.255.0.0 any
class-map match any FW_SEC_VIP
  10 match virtual-address 0.0.0.0 0.0.0.0 any

policy-map type loadbalance first-match SEC_20_LB
  class class-default
    serverfarm SEC_20_SF
policy-map multi-match POL_SEC_20
  class SEC_20_VS
```

```
    loadbalance vip inservice
    loadbalance policy SEC_20_LB

policy-map type loadbalance first-match LB_FW_SEC
  class class-default
    serverfarm SEC_SF
policy-map multi-match POL_SEC
  class FW_SEC_VIP
    loadbalance vip inservice
    loadbalance policy LB_FW_SEC

interface vlan 201
  ip address 100.201.1.201 255.255.0.0
  access-group input ACL1
  mac-sticky enable
  service-policy input POL_SEC_20
  no shutdown
interface vlan 20
  ip address 20.1.1.20 255.255.255.0
  access-group input ACL1
  service-policy input POL_SEC
  no shutdown
interface vlan 200
  ip address 200.1.1.200 255.255.255.0
  access-group input ACL1
  service-policy input POL_SEC
  no shutdown
```

Example of a Stealth Firewall Configuration

The following example shows those portions of the running configuration that pertain to stealth FWLB. This configuration requires two ACE modules each residing in a different Catalyst 6500 series switch.

ACE A Configuration—Stealth Firewall Load Balancing

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

rserver FW_INSEC_1
  ip address 101.0.201.100
  inservice
rserver FW_INSEC_2
  ip address 101.0.202.100
  inservice
rserver FW_INSEC_3
  ip address 101.0.203.100
  inservice

serverfarm INSEC_SF
  transparent
  predictor hash address source 255.255.255.255
  rserver FW_INSEC_1
    inservice
  rserver FW_INSEC_2
    inservice
  rserver FW_INSEC_3
    inservice

class-map match-any FORWARD-VIP
  10 match virtual-address 0.0.0.0 0.0.0.0 any
class-map match-any FW_VIP
  10 match virtual-address 200.1.1.1 255.255.0.0 any
policy-map type loadbalance first-match FORWARD_FW_INSEC
  class class-default
    forward
policy-map type loadbalance first-match LB_FW_INSEC
  class class-default
    serverfarm INSEC_SF
policy-map multi-match FORWARD_INSEC
  class FORWARD_VIP
    loadbalance vip inservice
    loadbalance policy FORWARD_FW_INSEC
```

```
policy-map multi-match POL_INSEC
  class FW_VIP
    loadbalance vip inservice
    loadbalance policy LB_FW_INSEC

interface vlan 100
  ip address 100.100.1.10 255.255.0.0
  access-group input ACL1
  service-policy input POL_INSEC
  no shutdown
interface vlan 101
  ip address 101.0.101.10 255.255.255.0
  alias 101.0.101.100 255.255.255.0
  access-group input ACL1
  service-policy input FORWARD_INSEC
  no shutdown
interface vlan 102
  ip address 101.0.102.20 255.255.255.0
  alias 101.0.102.100 255.255.255.0
  access-group input ACL1
  service-policy input FORWARD_INSEC
  no shutdown
interface vlan 103
  ip address 101.0.103.30 255.255.0.0
  alias 101.0.103.100 255.255.255.0
  access-group input ACL1
  service-policy input FORWARD_INSEC
  no shutdown
```

ACE B Configuration—Stealth Firewall Load Balancing

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

rserver host REAL1
  ip address 20.1.1.1
  inservice
rserver host REAL2
  ip address 20.1.1.2
  inservice
rserver host REAL3
  ip address 20.1.1.3
  inservice

rserver host FW_SEC_1
  ip address 101.0.101.100
  inservice
```

```
rserver host FW_SEC_2
  ip address 101.0.102.100
  inservice
rserver host FW_SEC_3
  ip address 101.0.103.100
  inservice

serverfarm SEC_20_SF
  rserver REAL1
    inservice
  rserver REAL2
    inservice
  rserver REAL3
    inservice
serverfarm SEC_SF
  transparent
  predictor hash address destination 255.255.255.255
  rserver FW_SEC_1
    inservice
  rserver FW_SEC_2
    inservice
  rserver FW_SEC_3
    inservice

class-map match-any SEC_20_VS
  10 match virtual-address 200.1.1.1 255.255.0.0 any
class-map match-any FW_SEC_VIP
  10 match virtual-address 0.0.0.0 0.0.0.0 any

policy-map type loadbalance first-match SEC_20_LB
  class class-default
    serverfarm SEC_20_SF
policy-map type loadbalance first-match LB_FW_SEC
  class class-default
    serverfarm SEC_SF
policy-map multi-match POL_SEC_20
  class SEC_20_VS
    loadbalance vip inservice
    loadbalance policy SEC_20_LB
policy-map multi-match POL_SEC
  class FW_SEC_VIP
    loadbalance vip inservice
    loadbalance policy LB_FW_SEC
```

```
interface vlan 201
  ip address 101.0.201.10 255.255.255.0
  alias 101.0.201.100 255.255.255.0
  access-group input ACL1
  mac-sticky enable
  service-policy input POL_SEC_20
  no shutdown
interface vlan 202
  ip address 101.0.202.20 255.255.255.0
  alias 101.0.202.100 255.255.255.0
  access-group input ACL1
  mac-sticky enable
  service-policy input POL_SEC_20
  no shutdown
interface vlan 203
  ip address 101.0.203.30 255.255.0.0
  alias 101.0.203.100 255.255.255.0
  access-group input ACL1
  mac-sticky enable
  service-policy input POL_SEC_20
  no shutdown
interface vlan 20
  ip address 20.100.1.100 255.255.0.0
  access-group input ACL1
  service-policy input POL_SEC
  no shutdown
interface vlan 200
  ip address 200.1.1.200 255.255.255.0
  access-group input ACL1
  service-policy input POL_SEC
  no shutdown
```

Where to Go Next

If you want to use toolkit command language (TCL) scripts with the ACE, see Appendix A, Using TCL Scripts with the ACE.

APPENDIX A

Using TCL Scripts with the ACE

You can copy, upload, and execute Toolkit Command Language (TCL) scripts on the ACE. TCL is a widely used scripting language within the networking community. TCL also has large libraries of developed scripts that can easily be found from various sites. Using TCL scripts, you can write TCL scripts for customized health probes. You can also execute ACE CLI commands in these scripts. The ACE also supports UDP socket functions.

Note The ACE does not support custom scripts to monitor itself.

Note The ACE can simultaneously execute only 200 scripted probe instances. When this limit is exceeded, the **show probe detail** command displays the "Out-of Resource: Max. script-instance limit reached" error message in the Last disconnect err field and the out-of-sockets counter increments.

This chapter provides information on scripts and contains the following topics:

- Scripts Overview
- Probe Script Quick Start
- Copying and Loading Scripts on the ACE
- Configuring Health Probes for Scripts
- Writing Probe Scripts
- Displaying Script Information
- Debugging Probe Scripts

Scripts Overview

The ACE supports several specific types of health probes (for example HTTP, TCP, or ICMP health probes) when you need to use a diverse set of applications and health probes to administer your network. The basic health probe types supported in the current ACE software release may not support the specific probing behavior that your network requires. To support a more flexible health-probing functionality, the ACE allows you to upload and execute TCL scripts on the ACE.

The TCL interpreter code in the ACE is based on Release 8.44 of the standard TCL distribution. You can create a script to configure health probes. Script probes operate similarly to other health probes available in the ACE software. As part of a script probe, the ACE executes the script periodically, and the exit code that is returned by the executing script indicates the relative health and availability of specific real servers. For information on health probes, see Chapter 4, Configuring Health Monitoring. If the script includes commands for ACE CLI commands, these CLI commands execute when the script probe executes. For information on TCL commands to execute ACE CLI commands, see Table A-4.

Cisco Systems-Supplied Scripts

For your convenience, the following Cisco Systems-supplied sample scripts for the ACE are available to support the TCL feature and are supported by the Cisco Technical Assistance Center (TAC):

- CHECKPORT_STD_SCRIPT
- ECHO_PROBE_SCRIPT
- FINGER_PROBE_SCRIPT
- FTP_PROBE_SCRIPT
- HTTP_PROBE_SCRIPT
- HTTPCONTENT_PROBE
- HTTPHEADER_PROBE
- HTTPPROXY_PROBE
- IMAP_PROBE
- LDAP_PROBE

- MAIL_PROBE
- POP3_PROBE
- PROBENOTICE_PROBE
- RTSP_PROBE
- SSL_PROBE_SCRIPT

These scripts are located in the probe: directory and are accessible in both the Admin and user contexts. To list the contents of this directory, use the following command:

```
host1/Admin# dir probe:
```

You can use these sample scripts with probes after you load the scripts into memory and associate them with probes. When you configure a new scripted probe, the ACE looks for the script file in the disk0: directory first, then the probe: directory. If a script file with the same name resides in both the probe: directory and the disk0: directory, the ACE uses the file in the disk0: directory. Note that the script files in the probe: directory are read-only, so you cannot copy or modify them.

For information about loading scripts into memory, see the "Loading Scripts into the ACE Memory" section. For information about associating a script with a probe, see the "Associating a Script with a Probe" section in Chapter 4, Configuring Health Monitoring.

Probe Suspects

A probe suspect is a destination (IP address and port) to which the ACE sends a probe. Typically, the IP address is the address associated with the object on which the probe is configured (for example, an rserver, a serverfarm, or an rserver configured in a serverfarm). You can configure the port using the **probe scripted** command. The IP address and port for each suspect are passed to the script in the scriptprobe_env array (see the "Environment Variables" section) as realIP and realPort, respectively. If you do not specify a port in the **probe scripted** command, the health probe scripts specify a default port in the script itself. For example, the SSL_PROBE_SCRIPT file specifies a default port of 443, the standard HTTPS port. For more information about the **probe scripted** command, see the "Configuring Health Probes for Scripts" section.

Probe Script Quick Start

Before you can run a probe script, you must copy the script onto the ACE, configure a script probe, and then associate the script with the probe. Table A-1 provides steps to copy and load a script on the ACE, and configure an associated scripted probe.

Table A-1 Probe Script Quick Start

Task and Command Example
1. Copy the script into the disk0: directory on the ACE. For example, to copy a script from an FTP server to the ACE and rename it to ACETCL, enter: `host1/Admin# ` **`copy ftp://192.168.1.1/test1/ECHO_PROBE_SCRIPT disk0:ACETCL`** `Enter username:` At the prompt, you must provide a username for the server. **Note** The filename that you assign the script must be unique across the contexts. You will use this filename when you load the script into the ACE memory and configure the probe.

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

3. Enter configuration mode.
`host1/Admin# ` **`config`**
`Enter configuration commands, one per line. End with CNTL/Z`
`host1/Admin(config)#`

4. Load the script into the ACE memory. `host1/Admin(config)# ` **`script file 22 ACETCL`**

Table A-1 Probe Script Quick Start (continued)

Task and Command Example

5. Create a scripted probe.

   ```
   host1/Admin(config)# probe scripted test1
   host1/Admin(config-probe-scripted)# interval 10
   host1/Admin(config-probe-scripted)# script ACETCL
   host1/Admin(config-probe-scripted)# exit
   ```

6. Create a real server and a server farm. Associate the probe and real server with the server farm.

   ```
   host1/Admin(config)# rserver host test
   host1/Admin(config-rserver-host)# ip address 10.1.0.105
   host1/Admin(config-rserver-host)# inservice
   host1/Admin(config-rserver-host)# exit
   host1/Admin(config)# serverfarm host tests
   host1/Admin(config-sfarm-host)# probe test1
   host1/Admin(config-sfarm-host)# rserver test
   host1/Admin(config-sfarm-host-rs)# inservice
   host1/Admin(config-sfarm-host-rs)# Ctrl-z
   ```

   At this point, the script probe should be running.

7. Use the **show probe detail** command in Exec mode to ensure that the script is running.

8. Stop the script probe.

   ```
   host1/Admin# config
   host1/Admin(config)# serverfarm host test
   host1/Admin(config-sfarm-host)# no probe test1
   host1/Admin(config-sfarm-host)# exit
   ```

Copying and Loading Scripts on the ACE

You load scripts onto the ACE through script files. A script file contains only one script. The ACE supports the configuration of 256 unique script files.

When using scripts on the ACE, the following considerations apply:

- Each script is always identified by its unique name as defined when copying the script file into the ACE disk0: file system. The script name must be unique across contexts.

- During probe configuration, you can assign a script to a probe. If the script is unavailable at that time, the probe attempts to execute the script and returns an error code. If this situation occurs, a syslog message displays to indicate the probe failure and why the probe failed. If the script is unavailable due to an error when loading the script, a syslog message would indicate the script load failure. You can also use the show script command to display the exit codes. For a list of exit codes, see Table A-7.
- To change a script that is already loaded into memory, you must unload and then reload the script. For information on loading a script file, see the "Loading Scripts into the ACE Memory" section. For information on reloading a script, see the "Removing Scripts from ACE Memory" section.

  After the script is changed in memory, the ACE applies the changes automatically the next time that the script executes. The command line arguments specified during probe configuration still apply after the reloading of the script.

Note Because the ACE does not replicate probe scripts to the standby in a redundant configuration, you must copy the scripts from the probe: directory of the active ACE to the probe: directory of the standby ACE. Otherwise, configuration synchronization does not work properly.

This section contains the following topics:

- Copying Scripts to the ACE
- Unzipping and Untarring ACE Sample Scripts
- Loading Scripts into the ACE Memory
- Removing Scripts from ACE Memory
- Reloading Modified Scripts in ACE Memory

Copying Scripts to the ACE

You can copy a script from a server to the ACE disk0: file system by using the **copy** command in Exec mode. You can also copy the file from the supervisor engine to the ACE.

Because of virtualization, by default, a script file is copied into the directory for the context that you are currently accessing. A script file in one context cannot be seen from another context. For details about virtualization, see the *Cisco Application Control Engine Module Virtualization Configuration Guide*. The syntax of this command is as follows:

copy [**ftp://***server***/***path* | **tftp://***server*[**:***port*]**/***path* |
sftp://[*username*@]*server***/***path*] **disk0:***filename*

The keywords and arguments are as follows:

- **ftp:**//*server*/*path*—Specifies the File Transfer Protocol (FTP) network server and the source location of the script file including its filename.
- **tftp:** //*server*[:*port*]/*path*]—Specifies the Trivial File Transfer Protocol (TFTP) network server and the source location of the script file including its filename.
- **sftp:**[//[*username*@]*server*][/*path*]—Specifies the Secure File Transfer Protocol (SFTP) network server and the source location of the script file including its filename.
- **disk0:***filename*—Specifies the destination filename for the script on the ACE disk0: file system. If you do not enter a filename, you are prompted to enter a filename or accept the source filename. You will use this filename when you load the script into the ACE memory and configure the probe.

Note The filename that you assign to the script must be unique across the contexts.

For example, to copy a script from an FTP server to the ACE, enter:

```
host1/Admin# copy ftp://192.168.1.1/test1/FTP_PROBE_SCRIPT
disk0:ftp1.tcl
Enter username:
```

At the prompt, you must provide a username for the server.

Using the ACE Sample Scripts

Cisco Systems provides sample probe scripts that you can use to associate with a health probe. The scripts are stored in the probe: directory.

You can also copy the zipped sample scripts file for the ACE onto disk0:. After you copy the zipped file, use the **gunzip** command in Exec mode to unzip its contents. For information about using this command, see the "Unzipping and Untarring ACE Sample Scripts" section.

Unzipping and Untarring ACE Sample Scripts

Sample scripts for the ACE are available to support the TCL feature. All of these scripts are provided in a zipped file which contains a .tar file. After you copy the zip file to the ACE, you need to unzip it and then untar it. When you untar the file, the ACE automatically creates an ace_scripts directory and places all of the individual scripts in it.

Note Some browsers, such as Internet Explorer version 6.0, automatically uncompresses a .tgz file. If you download the sample script file to the ACE with a browser that uncompresses the file, you can untar the file with the **untar** command. It is unnecessary to use the **gunzip** command on it.

You can unzip the sample scripts file by using the **gunzip** command in Exec mode. The syntax for this command is as follows:

gunzip disk0:[*path/*]*filename***.tgz**

The *filename* argument is the name of the zipped scripts file.

For example, to unzip the ace_scripts.tgz scripts file, enter:

```
host1/Admin# gunzip disk0:ace_scripts.tgz
```

The ACE unzips the file and places the ace_scripts.tar file in the disk0: file system.

To untar all of the script files from the ace_scripts.tar file, use the **untar** command in Exec mode. The syntax for the command is as follows:

untar disk0:[*path/*]*filename*

The *filename* argument is the name of the .tar file in the disk0: file system. The filename must end with a .tar extension.

For example, to untar all of the script files into the ace_scripts directory in the disk0: file system, enter:

```
host1/Admin# untar disk0:ace_scripts.tar
```

To view the scripts in the ace_scripts directory, use the **dir** command in Exec mode. For example, enter:

```
host1/Admin# dir disk0:ace_scripts/
```

Before you can load a sample script into memory, you must copy the script out of the ace_scripts directory into the disk0: directory. Use the **copy disk0:** command. For example, to copy the ftp1.tcl script from the ace_scripts directory to the disk0: directory, enter:

```
host1/Admin# copy disk0:ace_scripts/ftp1.tcl disk0:ftp1.tcl
```

Loading Scripts into the ACE Memory

You can load the script into memory on the ACE and enable it for use by using the **script file** command in configuration mode. The syntax of this command is as follows:

script file *index script_name*

The arguments are as follows:

- *index*—Index number for the script file. The index number will be associated with the script name. The number must be unique across the contexts. Enter a number from 1 to 255.
- *script_name*—Name of the script in the disk0: or probe: file system.

Note To load a script into memory, the script must be in the disk0: or probe: directory. The ACE does not load script files in a disk0: or probe: subdirectory.

For example, to load a script into memory:

```
host1/Admin(config)# script file 22 ftp1.tcl
```

To run the script or create a health probe using that script, use the script name you configured; do not use the script file from which the script was loaded.

Removing Scripts from ACE Memory

After a script file has been loaded, the scripts in that file exist in the ACE independent of the file from which that script was loaded. You can remove a script from memory and the running configuration by using the **no script file** command in configuration mode. The syntax of this command is as follows:

no script file *index*

The *index* argument is an index number for the script file you want to remove from memory.

For example, to remove the script with index 22, enter:

```
host1/Admin(config)# no script file 22
```

Reloading Modified Scripts in ACE Memory

If a script file is subsequently modified, you can update the script in memory by reloading it. Reloading a script requires the following:

1. Removing the script from memory by using the **no script file** command in configuration mode. For information on removing a script from memory, see the "Removing Scripts from ACE Memory" section.
2. Reloading the modified script into memory by using the **script file** command in configuration mode. For information about loading a script into memory, see the "Loading Scripts into the ACE Memory" section.

After the script is reloaded into memory, the ACE applies the changes automatically in the next script execution. The command-line arguments specified during probe configuration still apply after the reloading of the script.

For example, to reload a script 22, enter:

```
host1/Admin(config)# no script file 22
host1/Admin(config)# script file 22 ftp1.tcl
```

Configuring Health Probes for Scripts

You can create a scripted probe that the ACE periodically executes for each real server in any server farm associated with a probe. Depending upon the exit code of a script, the real server is considered passed or failed. For more information on exit codes, see the "Exit Codes" section.

To create a scripted probe, use the **probe scripted** *probe_name* command in configuration mode. This command enters a probe configuration mode that is similar to the existing ACE health probe modes (such as HTTP, TCP, DNS, SMTP, and so on).

The probe scripted configuration mode includes the **faildetect**, **interval**, **passdetect**, **open**, **priority**, and **receive** commands.The **script** *script_name* command can process up to 80 arguments that are passed to the script when it is run as part of the health probe function. When you configure each interval of time, an internal ACE scheduler schedules the health scripts. For more information on configuring scripted probes and the associated commands, see Chapter 4, Configuring Health Monitoring.

After creating the scripted health probe, attach the probe to the server farm and the virtual server. For example, enter:

```
host1/Admin(config)# serverfarm host tests
host1/Admin(config-sfarm-host)# probe test1
host1/Admin(config-sfarm-host)# rserver test
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
```

Writing Probe Scripts

Probe scripts test the health of a real server by creating a network connection to the server, sending data to the server, and checking the response. The flexibility of this TCL scripting environment makes the available probing functions possible.

Write the script as if you intend to perform only one probe. You must declare the result of the probe using the **exit** command. Depending upon the exit code of a script, the real server is considered passed or failed. For more information about exit codes, see the "Exit Codes" section.

A health script typically performs these actions:

- Opens a socket to an IP address.

- Sends one or more requests.
- Reads the responses.
- Analyzes the responses.
- Closes the socket.
- Exits the script by using an exit code for success or failure.

This section provides information to assist you when you write a probe script. The topics are as follows:

- TCL Script Commands Supported on the ACE
- Environment Variables
- Exit Codes
- Example for Writing a Probe Script

TCL Script Commands Supported on the ACE

The ACE TCL script feature is based on the TCL 8.44 source distribution software. Table A-2 lists the TCL commands that are supported by ACE.

Table A-2 TCL Commands Supported by the ACE

Command			
Generic TCL Commands[1]			
append	array	binary	break
case	catch	concat	continue
encoding	error	eval	exit
expr	fblocked	for	foreach
format	gets	glob	global
if	incr	info	join
lappend	lindex	linsert	list
llength	lrange	lreplace	lsearch
lset	lsort	namespace	proc
regexp	regsub	rename	return

Table A-2 TCL Commands Supported by the ACE (continued)

Command			
scan	set	split	string
subst	switch	unset	uplevel
upvar	variable	while	
Time-Related Commands			
after	clock	time	
Socket Commands			
close	eof	fconfigure	fileevent
flush	read	socket	update
vwait			

[1] The **puts** command can appear in a script, however, the ACE does not display its output.

Table A-3 lists the TCL command not supported by the ACE.

Table A-3 TCL Commands Not Supported by the ACE

Generic TCL Commands			
auto_execok	auto_import	auto_load	auto_load_index
auto_qualify	cd	exec	file
fcopy	history	interp	load
open	package	pid	pwd
seek	source	tell	trace

Table A-4 lists the TCL command specific to the ACE.

Table A-4 ACE-Specific TCL Commands

Command	Definition
gset *varname value*	Allows you to preserve the state of a probe by setting a variable that is global within an instance (probe to server association). In an instance of the probe, the value is retained. Each time the probe comes back to execute the script after the firing interval expires, the probe can then access this value. The same probe associated under a different server will not be able to access this value. Variables in a probe script are only visible within one probe thread. Each time a probe exits, all variables are gone. For example, if a probe script contains a 'gset x 1 ; incr x', variable x would increase by 1 for each probe attempt. • To set the value of variable from script, set *var* or $*var.* • To reset the value of variable from script, unset *var.* The variable is freed and cannot be accessed after performing the unset. • To display the **gset** value, have the script place this value in the exit message. You can use the **show script** command to display the exit message. Make sure that the script does not overwrite the exit message with another value before it exits. For information on the **show script** command, see the "Displaying the Statistics for an Active Script" section.

Table A-4 ACE-Specific TCL Commands (continued)

Command	Definition
set sock [**socket -sslversion** *version* **-sslcipher** *cipher* **$ip $port**]	Opens a socket and configures the specified SSL version number (*version*) and cipher (*cipher*), and enables accelerated SSL connections from a TCL script. The *version* argument is the SSL version that the probe supports. Enter one of the following case-sensitive keywords. • **all**—All SSL versions (default) • **SSLv3**—SSL version 3 • **TLSv1**—TLS version 1 The *cipher* argument is the RSA cipher suite that the probe expects from the back-end server. By default, the HTTPS probe accepts any of the RSA configured cipher suites. Enter one of the following case-sensitive keywords: • **RSA_WITH_RC4_128_MD5** • **RSA_WITH_RC4_128_SHA** • **RSA_WITH_DES_CBC_SHA** • **RSA_WITH_3DES_EDE_CBC_SHA** • **RSA_EXPORT_WITH_RC4_40_MD5** • **RSA_EXPORT_WITH_DES40_CBC_SHA** • **RSA_EXPORT1024_WITH_RC4_56_MD5** • **RSA_EXPORT1024_WITH_DES_CBC_SHA** • **RSA_EXPORT1024_WITH_RC4_56_SHA** • **RSA_WITH_AES_128_CBC_SHA** • **RSA_WITH_AES_256_CBC_SHA** You must enter both version and cipher values. If you enter the incorrect version or cipher value including the wrong case (upper or lowercase), the command uses the default value. Typically, the **$ip** keyword is the IP address of the real server to which the ACE sends the probe.

Table A-4 ACE-Specific TCL Commands (continued)

Command	Definition
set sock (continued)	Typically, the **$port** keyword is the port that you define in the scripted probe. If you do not define a port value in the scripted probe, the ACE uses the port defined with the real server. Although it is not a typical usage, you can define any IP address and port in the TCL script and then use those values in the **set sock** command, regardless of what is configured in the real server or the scripted probe.

vsh_conf_cmd $cmd_string	Allows the execution of the command or set of commands specified in the preceding **set** command string (**cmd_str**) by invoking the Vegas shell (Vsh). If you specify more than one command in the command string, separate them by the **\n** characters. For example, enter:
	`set cmd_str “rserver rs \n inservice”`
	`vsh_conf_cmd $cmd_str`

vsh_show_cmd $cmd_string	Allows the executing of the **show** command in the preceding **set** command string (**cmd_str**) by invoking the Vegas shell (Vsh). The output for the **show** command is set as a return value in the interpreter and the script invoking the commands must capture the results and parse the data.
	For example, enter:
	`set cmd str “show rserver rs1”`
	`set buffer [vsh_show_cmd $cmd_str]`

The UDP command set allows Scotty-based TCL scripts to run on the ACE. Scotty is the name of a software package that allows you to implement site-specific network management software using high-level, string-based APIs. The TCL UDP command reference is located at this URL:

http://wwwhome.cs.utwente.nl/~schoenw/scotty/

Table A-5 lists the UDP commands used by the ACE.

Table A-5 UDP Commands

Command	Definition
udp_binary send *handle* [*host port*] *message*	Sends binary data containing a message to the destination specified by host and port. The *host* and *port* arguments may not be used if the UDP handle is already connected to a transport endpoint. If the UDP handle is not connected, you must use these optional arguments to specify the destination of the datagram.
udp bind *handle* **readable** [*script*] **udp bind** *handle* **writable** [*script*]	Allows binding scripts to a UDP handle. A script is evaluated once the UDP handle becomes either readable or writable, depending on the third argument of the **udp bind** command. The script currently bound to a UDP handle can be retrieved by calling the **udp bind** command without a *script* argument. Bindings are removed by binding an empty string.
udp close *handle*	Closes the UDP socket associated with handle.
udp connect *host port*	Opens a UDP datagram socket and connects it to a port on a remote host. A connected UDP socket only allows sending messages to a single destination. This usually allows shortening the code because there is no need to specify the destination address for each **udp send** command on a connected UDP socket. The command returns a UDP handle.
udp info [*handle*]	Without the *handle* argument, this command returns a list of all existing UDP handles. Information about the state of a UDP handle can be obtained by supplying a valid UDP handle. The result is a list containing the source IP address, the source port, the destination IP address and the destination port.
udp open [*port*]	Opens a UDP datagram socket and returns a UDP handle. The socket is bound to given port number or name. An unused port number is used if the *port* argument is missing.

Table A-5 UDP Commands (continued)

Command	Definition
udp receive *handle*	Receives a datagram from the UDP socket associated with the handle. This command blocks until a datagram is ready to be received.
udp send *handle* [*host port*] *message*	Sends ASCII data containing a message to the destination specified by host and port. The *host* and *port* arguments may not be used if the UDP handle is already connected to a transport endpoint. If the UDP handle is not connected, you must use these optional arguments to specify the destination of the datagram.

Environment Variables

Health probe scripts have access to many configured items through a predefined TCL array. The most common use of this array is to find the current real server IP addresses of the suspect during any particular launch of the script.

Whenever the ACE executes a script probe, a special array called scriptprobe_env is passed to the script. This array holds important parameters that may be used by the script.

Table A-6 lists the members of the scriptprobe_env array.

Table A-6 Member List for the scriptprobe_env Array

Member Name	Content
realIP	Suspect IP address.
realPort[1]	Suspect IP port.
intervalTimeout[1]	Configured probe interval in seconds.
openTimeout	Configured socket open timeout for this probe.
recvTimeout[1]	Configured socket receive timeout for this probe.
failedTimeout[1]	Configure failed timeout.
retries[1]	Configured retry count.
healthStatus	Current suspect health status.

Table A-6 Member List for the scriptprobe_env Array (continued)

Member Name	Content
contextID	The ID for the context running this script.
failedRetries[1]	Consecutive successful retries on a failed server before marking it as passed.
isRouted	Boolean to determine if this IP address is a routed address.
pid	Process identifier of the TCL process.
runID	Pointer to the event structure (em_event_t).

[1] Configurable parameter

For more information about the configurable parameters in the scriptprobe_env array, see the "Configuring General Probe Attributes" section in Chapter 4, Configuring Health Monitoring.

Exit Codes

The probe script uses exit codes to signify various internal conditions. The exit code information can help you troubleshoot your scripts if they do not operate correctly. A probe script indicates the relative health and availability of a real server using the exit code of the script. By calling exit 30001, a script indicates that the server successfully responded to the probe. Calling exit 30002 indicates that the server did not respond correctly to the health probe.

For example, if a probe script fails and exits with 30002, the corresponding server is marked as PROBE_FAILED and is temporarily disabled from the server farm. The ACE continues to probe the server. When the probe successfully reconnects and exits with 30001, the ACE marks the server status as OPERATIONAL and enables the server from the server farm again. The exit 30001 must occur the number of failedRetries times before the server is marked as OPERATIONAL. See the previous section on environmental variables for further information on the failedRetries member name.

These situations can cause a script to fail and mark the suspect PROBE_FAILED:

- TCL errors—Occurs when scripts contain errors that are caught by the TCL interpreter, for example, a syntax error.

  The syntax error message is stored in the special variable **erroInfo** and can be viewed using the **show script** command in Exec mode. Another example of TCL errors would cause the TCL interpreter to abort and call panic.
- A stopped script—Caused by an infinite loop and wait indefinitely for a response. Each script must complete its task within the configured time interval. If the script does not complete its task, the script controller terminates the script, and the suspect is failed implicitly.
- Error conditions—Occurs when a connection timeout or a peer-refused connection is also treated as an implicit failure.

Table A-7 shows all exit codes used in the ACE.

Table A-7 ACE Exit Codes

Exit Code	Description/Message
30001	Probe successful (no message).
30002	Probe error: Server did not respond as expected.
30003	Internal error: Fork failed for TCL script.
30004	Internal error: Script probe terminated due to timeout.
30005	Internal error: TCL interpreter PANIC (interpreter problem).
30006	Internal error: Script error.
30007	Internal error: Script-file lookup failed or empty buffer.
30008	Internal error: Failed to allocate memory for TCL_wt (worker thread) qnode.
30009	Internal error: Unknown script error.
30010	Internal error: Out of sockets for the TCL script.
30011	Internal error: Unable to read persistent variable table.
30012	Internal error: PData (probe data) pointer is null.

Example for Writing a Probe Script

This example shows how a script is written to probe an HTTP server using a health script:

```
# get the IP address of the real server from a predefined global array
# scriptprobe_env
set ip $scriptprobe_env(realIP)
set port 80
set url "GET /index.html HTTP/1.0\n\n"

# Open a socket to the server. This creates a TCP connection to the
# real server
set exit_msg "opening socket"
set sock [socket $ip $port]
fconfigure $sock -buffering none -eofchar {}

# Wait for the response from the server and read that in variable line
set exit_msg "receiving response"
set line [ read $sock ]

# Parse the response
if { [ regexp "HTTP/1.. (\[0-9\]+) " $line match status ] }

# Close the socket. Application MUST close the socket once the
# request/response is over.
# This allows other applications and tcl scripts to make
# a good use of socket resource. Health monitoring is allowed to open
# only 200 sockets simultaneously.
set exit_msg "closing socket"
close $sock

# decide the exit code to return to control module.
# If the status code is OK then script MUST do exit 30001
# to signal successful completion of a script probe.
# In this example any other status code means failure.
# User must do exit 30002 when a probe has failed.
if { $status == 200 } {
    set exit_msg "probe success"
    exit 30001
} else {
    set exit_msg "probe fail : can't find status code"
    exit 30002
```

Displaying Script Information

The following sections provide information for displaying scripted probe and script information:

- Displaying ACE Script and Scripted Probe Configuration
- Displaying Scripted Probe Information
- Displaying Global Scripted Probe Statistics
- Displaying the Statistics for an Active Script
- Displaying the Script Contents

Displaying ACE Script and Scripted Probe Configuration

You can display configuration information for scripts and scripted probes on the ACE by using the **show running-config** command in Exec mode. For example, enter:

```
switch/Admin# show running-config
Generating configuration....

logging enable
boot system image:

script file 200 ftp1.tcl

probe scripted bts_test_probe
  interval 10
  receive 8
  script test.tcl

rserver host bts_test_rserver
  ip address 10.86.209.1
  probe bts_test_probe
  inservice

snmp-server user www Network-Monitor
snmp-server user admin Network-Monitor

context Admin
...
```

Displaying Scripted Probe Information

You can display configuration and probe result information about scripted probes by using the **show probe** command in Exec mode. The syntax of this command is as follows:

show probe *scripted_probe_name* [**detail**]

The keyword, argument, and option are as follows:

- *scripted_probe_name*—Information for the specified scripted probe name.
- **detail**—(Optional) Displays detailed probe information including configuration information and statistics.

If you do not enter a probe name, this command shows a summary of information for all configured probes.

For example, to view the SCRIPT-1 scripted probe and detailed information, enter:

```
host1/Admin# show probe SCRIPT-1 detail
```

Table A-8 describes the fields in the **show probe** command output for a scripted probe.

Table A-8 Field Descriptions for the show probe Command for a Scripted Probe

Field	Description
probe	Name of the probe.
type	Probe type.
description	Configured string that describes the probe.
port	Port number that the probe uses. By default, the probe uses the port number based on its type.
address	Not used for scripted probes.
addr type	Not used for scripted probes.
interval	Time interval in seconds that the ACE sends probes to a server marked as passed.

Table A-8 Field Descriptions for the show probe Command for a Scripted Probe (continued)

Field	Description
pass intvl	Time period in seconds to send a probe to a failed server.
pass count	Number of successful probe replies before enabling failed server.
fail count	Consecutive number of failed probes before marking the server as failed.
recv timeout	Time period in seconds to receive a server response to the probe.
script filename	Script filename.
probe association	Serverfarm or real server association.
probe results	
probed-address	Destination or source address for the probe.
probes	Total number of probes.
failed	Total number of failed probes.
passed	Total number of passed probes.
health	Current health of probe: PASSED or FAILED.
Additional Detailed Output:	
Socket state	Socket state.
No. Passed states	Number of passed states.
No. Failed states	Number of failed states.
No. Probes skipped	Number of skipped probes.
Last status code	Last exit code (see Table A-7).
Last disconnect err	Message for the exit code (see Table A-7).
Last probe time	Timestamp for the last probe.
Last fail time	Timestamp for the last failed probe.
Last active time	Timestamp for the last active time.
Internal error	Counter for the number of internal errors encountered.

Displaying Global Scripted Probe Statistics

You can display the global statistics for all scripted probes by using the **show stats probe type scripted** command in Exec mode. For example, enter:

```
host1/Admin# show stats probe type scripted
```

Table A-9 describes the fields in the **show stats probe type scripted** command output.

Table A-9 Field Descriptions for the show stats probe type scripted command

Field	Description
Total probes sent	Total number of probes sent by all scripted probes.
Total send failures	Total number of send failures for all scripted probes. These failures are due to internal errors. For more information, see the last disconnect error field displayed by the **show probe** command.
Total probes passed	Total number of passed probes for all scripted probes.
Total probes failed	Total number of failed probes for all scripted probes.
Total connect errors	Total number of connection errors for all scripted probes.
Total conns refused	Total number of connections refused for all scripted probes.
Total RST received	Total number of resets received by all scripted probes.
Total open timeouts	Total number of open timeouts for all scripted probes.
Total receive timeouts	Total number of time outs received by all scripted probes.

Displaying the Statistics for an Active Script

You can display the statistics for a script file active on the ACE including exit codes and exit messages by using the **show script** command from Exec mode. The syntax for this command is as follows:

show script *script_name probe_name* [*rserver_name* [*port_name*]] [**serverfarm** *sfarm_name*]

The keywords, arguments, and options are as follows:

- *script_name*—Name of the script.
- *probe_name*—Name of the scripted probe that is associated with the script.
- *rserver_name* [*port_name*]—(Optional) Name of the real server and optional port number that uses the scripted probe.
- **serverfarm** *sfarm_name*—Specifies the name of the server farm that uses the scripted probe, enter.

For example, to display the statistics for the ECHO_PROBE_SCRIPT script for the SCRIPT1 probe, enter:

```
host1/Admin# show script ECHO_PROBE_SCRIPT SCRIPT1
```

Table A-10 describes the fields in the **show script** command output.

Table A-10 Field Descriptions for the show script command

Field	Description
Script	Name of the script.
Scripted probe	Probe associated with the script.
Probe-association(s): (count=*number*)	Total number of real servers and server farms associated with the scripted probe.
Rserver/Serverfarm	Name of the real server or server farm for the script statistics.
Exit code	Current exit code for the script. See Table A-7 for information on exit codes.
Child PID	Child process identifier for the script.
Exit message	Value of the TCL **gset** variable from the script.

Table A-10 Field Descriptions for the show script command (continued)

Field	Description
Panic string	Indication of an internal problem with the TCL interpreter.
Internal error	Error code message associated with the exit code.
Last RunStart count/Last RunStart time	Number of times that the script successfully started and the last time stamp.
Last RunEnd count/Last RunEnd time	Number of times that the script successfully ended and the last time stamp.
Last Probe OK count/ Last Probe OK time	Number of times that the scripted probe passed and the last time stamp.
Last ProbeFail count/ Last ProbeFail time	Number of times that the scripted probe failed and the last time stamp.
Last ForkFail count/ Last ForkFail time	Number of times that the fork failed and the last time stamp.
Last Kill count/ Last Kill time	Number of times that the script probe failed due to timeout and the last time stamp.
Last Panic count/ Last Panic time	Number of times that there was an internal problem with the TCL interpreter and the last time stamp.
Last Nodecreate Error count/Last Nodecreate Error time	Number of times that the ACE ran out of memory for the TCL worker thread node and the last time stamp.
Last Unassociated Script count/ Last Unassociated Script time	Number of times that the script in memory could not be associated and the last time stamp.
Last Fail Internal/ Last Fail Internal	Number of times that the script had an internal syntax error and the last time stamp.
Last Socket-Limit count/ Last Socket-Limit time	Number of times that the ACE ran out of sockets for the TCL script and the last time stamp.
Last PV-Read count/ Last PV-Read Time	Number of times that the persistent variable table was read and the last time stamp.

Table A-10 Field Descriptions for the show script command (continued)

Field	Description
Last PData-Null count/Last PData-Null time	Number of times that the probe data pointer is null and the last time stamp.
Last Unknown count/ Last Unknown time	Number of times that the script passed an error code that is not recognized and last time stamp.

Displaying the Script Contents

You can display the contents for a script file loaded on the ACE by using the **show script** command from Exec mode. The syntax for this command is as follows:

show script code *script_name*

The keyword and argument are as follows:

- **code**—Displays the code within the script file.
- *script_name*—Name of the script.

For example, to display the code within the ECHO_PROBE_SCRIPT script, enter:

```
host1/Admin# show script code ECHO_PROBE_SCRIPT
```

Debugging Probe Scripts

You can debug a script probe by doing the following:

- Use the EXIT_MSG variable in the script. Each probe suspect contains its own EXIT_MSG variable. This variable allows you to trace the status of a script and check the status of the probe.

  This example shows how to use the EXIT_MSG variable in a script:

  ```
  set EXIT_MSG "before opening socket"
  set sock [ socket $ip $port]
  set EXIT_MSG " before receive string"
  gets $s
  set EXIT_MSG "before close socket"
  close $s
  ```

  If a probe suspect fails when receiving the message, you should see EXIT_MSG = before you receive the string. You can view the EXIT_MSG variable with the **show script** command in Exec mode.

- Use the **show probe** command in Exec mode to view the current active probe suspects in the system. For more information on this command, see the "Displaying Script Information" section.

- Use the **show script** command in Exec mode to view the following:
  - The last exit status with the exit code number.
  - The internal error that is generated by the TCL compiler. When the script has a TCL runtime error, the TCL interpreter stops running the script and the ACE displays this error.
  - The EXIT_MSG variable value of the TCL **gset** command from the script.

INDEX