CHAPTER 5

Configuring Stickiness

Note The information in this chapter applies to both the ACE module and the ACE appliance unless otherwise noted. The ACE supports all the features described in this chapter with IPv6 except where noted.

This chapter describes how to configure stickiness (sometimes referred to as session persistence) on an ACE. 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 4095 sticky groups in an ACE. 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 IPV6 prefix-length or IPv4 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.

Note The ACE does not support IP address stickiness where the source address is IPv4 and the destination address is IPv6 or the reverse. IP address stickiness where the source and destination are both IPv6 is supported.

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 for IPv4only. 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 for IPv4only. 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 for IPv4 only. 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 the following:

- (ACE module only) A maximum of 4,000,000 (4,000,000 simultaneous users). This limit applies equally to IPv6 and IPv4.
- (ACE appliance only) A maximum of 800,000 entries (800,000 simultaneous users). This limit applies equally to IPv6 and IPv4.

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 4095 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 or ND 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:

- Starting with software version A4(1.0), it is no longer necessary to configure a resource class in the Admin context to allocate resources for stickiness. You can still explicitly allocate sticky resources if you wish, but skipping this step will not affect sticky functionality. So, you can create a sticky group and create sticky database entries even if you have not explicitly allocated resources for stickiness. This is possible because the ACE uses a global pool of resources, including sticky resources. When a context needs additional resources, it takes them from the global pool if there are resources available. When resources are released from a context, the ACE returns them to the

global pool. If there are no resources available in the global pool when a context needs them, the ACE places the context in starvation mode until more resources are released.

If you explicitly allocate resources for stickiness, the ACE considers both the minimum and the maximum values, including max set to unlimited. If the minimum value is reached, the maximum value is set to unlimited, and there are resources available in the global pool, the LB module can take resources from the pool to create a new sticky database entry.

For static sticky entries, if the ACE accepts the CLI command, it inserts the entry into the sticky database. If there are no resources immediately available, the ACE evaluates other contexts and takes resources from one or more contexts if possible.

For details about configuring resource groups and allocating resources, see the *Virtualization Guide, Cisco ACE Application Control Engine*.

- 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.
- For two VIPs pointing to the same Layer 7 policy, if you have enabled the case-sensitive option for one VIP and disabled it for the other VIP under the Layer 3 policy-map, you can have up to 8K static sticky entries.
- 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 *Administration Guide, Cisco ACE Application Control Engine*.

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

3. for IPv6, create a sticky-IP group and enter sticky-IP configuration mode.
`host1/Admin(config)# sticky v6-prefix 64 address both GROUP1`
`host1/Admin(config-sticky-ip)#`

4. For IPv4, 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`

Table 5-1 IP Address Stickiness Configuration Quick Start (continued)

Task and Command Example
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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-ip)# 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-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 v6-prefix** or the **sticky ip-netmask** command in configuration mode. You can create a maximum of 4095 sticky groups on an ACE. The syntax of this command is as follows:

sticky {**v6-prefix** *prefix_length* | **ip-netmask** *netmask*} **address** {**source** | **destination** | **both**} *name*

The keywords and arguments are as follows:

- **v6-prefix** *prefix_length*—For IPv6, specifies how many of the most significant bits (MSBs) of the IPv6 address are used for the network identifier. Enter an integer from 1 to 128.
- **ip-netmask** *netmask*—For IPv4, 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 (ACE module only) If you configure a network mask other than 255.255.255.255 (/32), the ACE module may populate the sticky entries only on one of its four network processors which may reduce the number of available sticky entries by 25 percent. This reduction in resources may cause problems when heavy sticky use occurs on the ACE module.

- **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.

IPv6 Example

To create a sticky group that uses IPv6 address stickiness based on both the source IPv6 address and the destination IPv6 address, enter:

```
host1/Admin(config)# sticky v6-prefix 64 address both GROUP1
host1/Admin(config-sticky-ip)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky v6-prefix 64 address both GROUP1
```

IPv4 Example

To create a sticky group that uses IPv4 address stickiness based on both the source IP address and the destination IPv4 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 it 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 theIPv6 or IPv4 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 4095 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.
- **destination** *ip_address*—Specifies that the static entry be based on the destination IP address.
- **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.

IPv6 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 2001:DB8:12::15
destination 2001:DB8:27::3 rserver SERVER1 2000
```

To remove the static entry from the sticky table, enter:

```
host1/Admin(config-sticky-ip)# no static client source 2001:DB8:12::15
destination 2001:DB8:27::3 rserver SERVER1 2000
```

IPv4 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.

IPv6 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 2001:DB8:252::240
  inservice
rserver host SERVER2
  ip address 2001:DB8:252::241
  inservice
rserver host SERVER3
  ip address 2001:DB8:252::242
  inservice

serverfarm host SFARM1
  probe ICMP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice
```

```
sticky v6-prefix 64 address both SGROUP1
  timeout 20
  replicate sticky
  serverfarm SFARM1

class-map match-all L4STICKY-IP_115:ANY_CLASS
  2 match virtual-address 2001:DB8:120::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 2001:DB8:120::1/64
  fragment chain 20
  fragment min-mtu 68
  access-group input ACL1
  nat-pool 1 2001:DB8:120::70 2001:DB8:120::70/64 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 2001:DC8:1::1/64 2001:DB8:120::254
```

IPv4 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 *Administration Guide, Cisco ACE Application Control Engine*.

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

3. 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)#`

4. Configure a timeout for Layer 4 payload stickiness. `host1/Admin(config-sticky-l4payloa)# timeout 720`
5. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-l4payloa)# timeout activeconns`

Table 5-2 Layer 4 Payload Stickiness Configuration Quick Start

Task and Command Example
6. 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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-l4payloa)# replicate sticky`
7. Enable sticky learning for responses from the server. `host1/Admin(config-sticky-l4payloa)# response 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-l4payloa)# serverfarm SFARM1 backup BKUP_SFARM2 sticky`
9. (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`
10. (Optional) Configure one or more static sticky payload entries. `host1/Admin(config-sticky-l4payloa)# static layer4-payload 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 5-2 Layer 4 Payload Stickiness Configuration Quick Start

Task and Command Example
15. 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`
16. (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. You can create a maximum of 4095 sticky groups. 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 it 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 4095 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 *Administration Guide, Cisco ACE Application Control Engine*.

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

Table 5-3 HTTP Content Stickiness Configuration Quick Start (continued)

Task and Command Example

3. 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)#`

4. Configure a timeout for HTTP content stickiness. `host1/Admin(config-sticky-content)# timeout 720`
5. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-content)# timeout activeconns`
6. 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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-content)# replicate sticky`
7. 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`
8. (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`
9. (Optional) Configure one or more static sticky content entries. `host1/Admin(config-sticky-content)# static content stingray rserver RS1 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.

Table 5-3 HTTP Content Stickiness Configuration Quick Start (continued)

Task and Command Example
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 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`
15. 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 4095 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 it 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 4095 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 *Administration Guide, Cisco ACE Application Control Engine*.

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

3. 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)#`

4. Configure a timeout for HTTP cookie stickiness. `host1/Admin(config-sticky-cookie)# timeout 720`
5. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-cookie)# timeout activeconns`

Table 5-4 HTTP Cookie Stickiness Configuration Quick Start (continued)

Task and Command Example
6. 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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-cookie)# replicate sticky`
7. 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`
8. (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`
9. (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`
10. (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`
11. (Optional) Configure one or more static sticky cookie entries. `host1/Admin(config-sticky-cookie)# static cookie-value corvette rserver SERVER1 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.

Table 5-4 HTTP Cookie Stickiness Configuration Quick Start (continued)

Task and Command Example
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 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`
17. (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 4095 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 it 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).

Note When you configure sticky timeout for an HTTP cookie, the timeout translates into the expiration date for the cookie. This expiration date can be longer than the actual timeout specified in the **timeout** command, with sometimes as much as 20 to 25 minutes added to the expiration date.

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.

You can also specify a custom cookie name for cookie insertion by using the **cookie-string** *value* command in server farm real server configuration mode. For more information, see the "Configuring a Real Server Cookie Value for Cookie Insertion" section in Chapter 2, Configuring Real Servers and Server Farms.

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. Replication suggests that the ACE builds sticky entry based on the primary cookie only if both cookies are present in the client request. When only the secondary cookie is present and it is configured under stickiness, it will create a new sticky table. 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 4095 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.

Note Port number can only be configured if the real server is configured under a server farm with a port number. If no port is configured for the server farm, then you cannot specify a port for the static cookie.

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 examples show an IPv6 and an IPv4 running configuration that defines HTTP cookie stickiness. The HTTP cookie stickiness configuration appears in bold in the example.

IPv6 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 anyv6 anyv6

probe icmp ICMP
  interval 2
  faildetect 2
  passdetect interval 2

rserver host SERVER1
  ip address 2001:DB8:252::240
  inservice
rserver host SERVER2
  ip address 2001:DB8:252::241
  inservice
rserver host SERVER3
  ip address 2001:DB8:252::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 2001:DB8:120::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 2001:DB8:120::1/64
  fragment chain 20
  fragment min-mtu 68
  access-group input ACL1
  nat-pool 1 2001:DB8:120::70 2001:DB8:120::70/64 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 2001:DB8::1/64 2001:DB8:120::254
```

IPv4 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 *Administration Guide, Cisco ACE Application Control Engine*.

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

3. 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)#`

4. Configure a timeout for header stickiness. `host1/Admin(config-sticky-header)# timeout 720`
5. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-header)# timeout activeconns`

Table 5-5 HTTP Header Stickiness Configuration Quick Start (continued)

Task and Command Example
6. 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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-header)# replicate sticky`
7. 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`
8. (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`
9. (Optional) Configure one or more static header sticky entries. `host1/Admin(config-sticky-header)# static header Host 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 5-5 HTTP Header 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 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 4095 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 it 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 4095 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 examples show an IPv6 and an IPv4 running configuration that defines HTTP header stickiness. The HTTP header stickiness configuration appears in bold in the example.

IPv6 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 anyv6 anyv6

probe http HTTP
  interval 5
  passdetect interval 10
  receive 5
  expect status 200 200
  open 3

rserver host SERVER1
  ip address 2001:DB8:252::240
  inservice
rserver host SERVER2
  ip address 2001:DB8:252::241
  inservice
rserver host SERVER3
  ip address 2001:DB8:252::242
  inservice
rserver host SERVER4
  ip address 2001:DB8:252::243
  inservice
rserver host SERVER5
  ip address 2001:DB8:252::244
  inservice
rserver host SERVER6
  ip address 2001:DB8:252::245
  inservice
rserver host SERVER7
  ip address 2001:DB8:252::246
  inservice
rserver host SERVER8
  ip address 2001:DB8:252::247
  inservice
rserver host SERVER9
  ip address 2001:DB8:252::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 2001:DB8:120::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 2001:DB8:120::1 255.255.255.0
  fragment chain 20
  fragment min-mtu 68
  access-group input ACL1
  nat-pool 1 2001:DB8:120::70 2001:DB8:120::70/64 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 2001:DC8::/64 2001:DB8:120::254
```

IPv4 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. The ACE does not support this feature with IPv6.

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 *Administration Guide, Cisco ACE Application Control Engine*.

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

3. 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)#`

4. Configure a timeout for RADIUS-attribute stickiness. `host1/Admin(config-sticky-radius)# timeout 720`
5. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-radius)# timeout activeconns`
6. 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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-radius)# replicate sticky`

Table 5-7 RADIUS-Attribute Stickiness Configuration Quick Start

Task and Command Example
7. 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`
8. Configure a Layer 3 and Layer 4 SLB traffic policy. See Chapter 3, Configuring Traffic Policies for Server Load Balancing.
9. 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.
10. 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.
11. 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.
12. 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`
13. (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 4095 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 it 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 *Administration Guide, Cisco ACE Application Control Engine*.

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

3. 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)#`

4. Configure a timeout for RTSP header stickiness. `host1/Admin(config-sticky-header)# timeout 720`
5. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-header)# timeout activeconns`
6. 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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-header)# replicate sticky`

Table 5-8 RTSP Session Stickiness Configuration Quick Start (continued)

Task and Command Example
7. 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`
8. (Optional) Configure one or more static RTSP header sticky entries. `host1/Admin(config-sticky-header)# static header Session rserver SERVER1 4000`
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.
13. 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`
14. (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 4095 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 it 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 4095 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 *Administration Guide, Cisco ACE Application Control Engine*.

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

3. 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)#`

4. Configure a timeout for SIP header stickiness. `host1/Admin(config-sticky-header)# timeout 720`
5. (Optional) Enable the timeout to override active connections. `host1/Admin(config-sticky-header)# timeout activeconns`
6. 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 *Administration Guide, Cisco ACE Application Control Engine*. `host1/Admin(config-sticky-header)# replicate sticky`
7. 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`
8. (Optional) Configure one or more static header sticky entries. `host1/Admin(config-sticky-header)# static header Call-ID rserver SERVER1 4000`

Table 5-9 SIP Call-ID Stickiness Configuration Quick Start (continued)

Task and Command Example
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.
13. 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`
14. (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 4095 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 it 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 4095 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 *Administration Guide, Cisco ACE Application Control Engine*.

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 76`
`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 it 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"**. 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"
```

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 examples show 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.

IPv6 Example

```
resource-class RC1
  limit-resource sticky minimum 10.00 maximum equal-to-min

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

```
parameter-map type generic SSLID_PARAMMAP
  set max-parse-length 76

rserver SSL_SERVER1
  ip address 2001:DB8:12::2
  inservice
rserver SSL_SERVER2
  ip address 2001:DB8:12::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 2001:DB8:16::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 -------->ACE module only
    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 2001:DB8:16::1/64
  access-group input ACL1
  service-policy input L4_POLICY

interface vlan 300
  ip address 2001:DB8:12::1/64

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.

IPv4 Example

```
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 76

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 -------->ACE module only
    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:

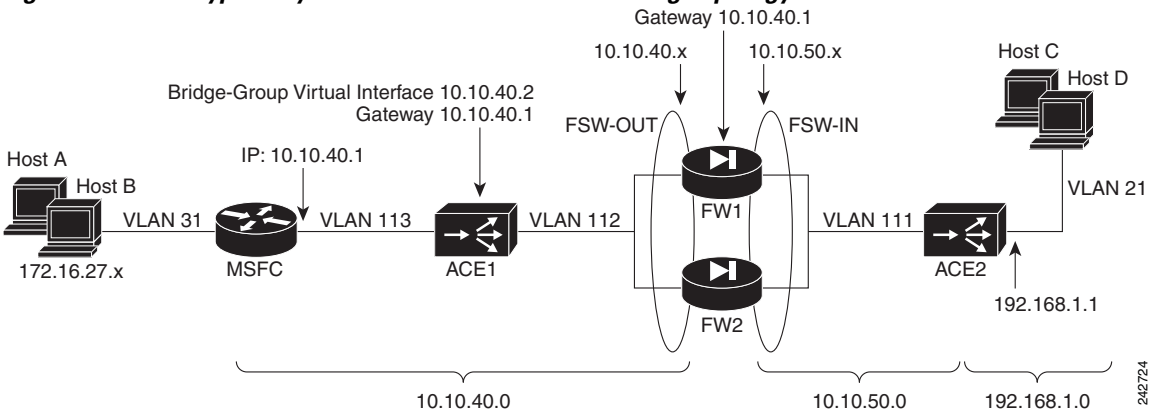
- Symmetric Topology
- Asymmetric Topology

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 does the following:

- (ACE module only) Forwards the traffic to the MSFC and from there to the client.
- (ACE appliance only) Forwards the traffic 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 does the following:

- (ACE module only) Forwards the traffic to the MSFC and from there to the client (Host A).
- (ACE appliance only) Forwards the traffic to the client (Host A).

Follow these guidelines and observations when you configure reverse IP stickiness:

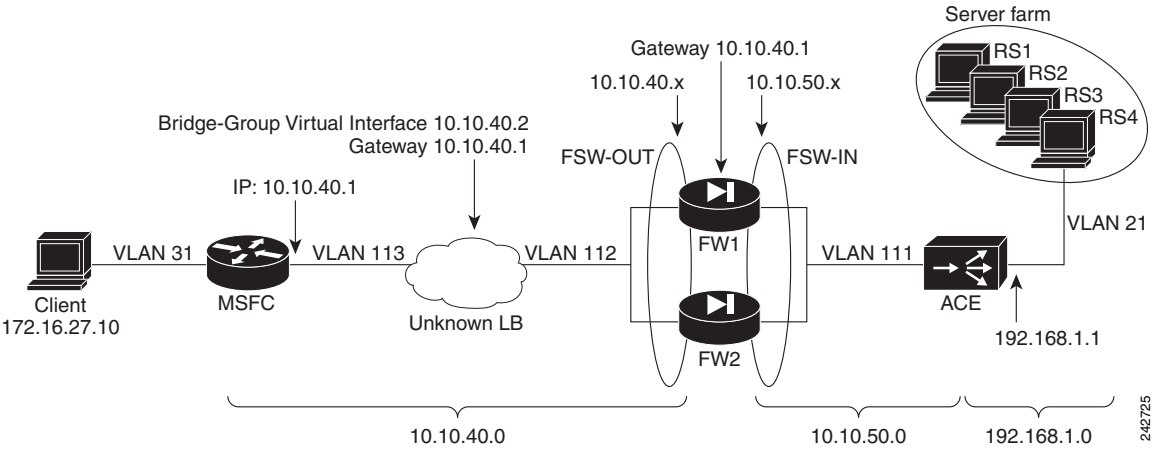
- When reverse IP sticky is enabled, the sticky entry is populated in one direction (for incoming traffic) and looked up in the opposite direction (for outgoing traffic), allowing traffic to flow through the same firewall in both directions.
- The example that is described in the steps above is symmetric 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 and another sticky group based on the destination IP associated with the VLAN 21 interface of the ACE 2 (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 which in turn does the following:

- (ACE module only) Forwards the traffic to the MSFC and then to the client.
- (ACE appliance only) Forwards the traffic 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 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 *Server Load-Balancing Guide, Cisco ACE Application Control Engine*.

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.
- You cannot configure regular stickiness and reverse stickiness under the same Layer 7 policy map. If you need both types of stickiness, configure them in separate Layer 7 policy maps and apply them to different interfaces using different service policies.
- 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.

IPv6 Example

To configure reverse IP stickiness for a sticky group called DEST_IP_STICKY with IPv6, enter the following sequence of commands:

```
host1/Admin(config)# sticky v6-prefix 64 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
```

IPv4 Example

To configure reverse IP stickiness for a sticky group called DEST_IP_STICKY with IPv4, 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:

IPv6 Example

ACE 1 Configuration

```
access-list acl1 line 8 extended permit ip anyv6 anyv6

rserver host FW1
  ip address 2001:DB8:40::10
  inservice
rserver host FW2
  ip address 2001:DB8:40::20
  inservice

serverfarm host FWS-OUT
  transparent
  rserver FW1
    inservice
  rserver FW2
    inservice

sticky v6-prefix 64 address source SOURCE_IP_STICKY
  serverfarm FWS-OUT

class-map match-all CATCH-ALL-VIP
  2 match virtual-address ::/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 2001:DB8:40::1
```

ACE 2 Configuration

```
access-list acl1 line 8 extended permit ip anyv6 anyv6

rserver host FW1
  ip address 2001:DB8:50::10
  inservice
rserver host FW2
  ip address 2001:DB8:50::20
  inservice
```

```
serverfarm host FWS-IN
  transparent
  rserver FW1
    inservice
  rserver FW2
    inservice

sticky v6-prefix 64 address destination DEST_IP_STICKY
  serverfarm FWS-IN

class-map match-all CATCH_ALL_VIP
  2 match virtual-address ::/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 2001:DB8:1::1/64
  access-group input acl1
  service-policy input L4_TO_FWS
  no shutdown
interface vlan 111
  description inside FW vlan
  ip address 2001:DB8:50::1/64
  mac-sticky enable
  access-group input acl1
  service-policy input L4_TO_REALS
  no shutdown
```

IPv4 Example

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 2001:DB8:1::10 tcp eq
80
or
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:

- Active connections
- Client IP address
- Sticky group ID

- HTTP content value
- HTTP cookie value
- HTTP header value
- Layer 4 payload
- RADIUS information
- Real-server name
- RTSP header
- SIP header
- Static sticky database entries
- Sticky group type
- Time to expire

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**] [**active-conn-count min** *value1* **max** *value2* | **client** *ip_address1* | **group** *name1* | **http-content** *value3* | **http-cookie** *value2*4 | **http-header** *value5* | **ip-netmask** {**both** {**source** *ip_address2* **destination** *ip_address3*} | **destination** *ip_address4* | **source** *ip_address5*} | **layer4-payload** *value6* | **rserver** *name2* [*port*] **serverfarm** *name3* | **rtsp-header** *value7* | **sip-header** *value8* | **time-to-expire min** *value9* **max** *value10* | **type** {**http-content** | **http-cookie** | **http-header** | **ip-netmask** {**both** | **destination** | **source**} | **layer4-payload** *value* | **radius** {**calling-id** | **framed-ip** | **username**} | **rtsp-heade**r value | **sip-header value**} [**count** | **detail**]]

The keywords, arguments, and options are as follows:

- **static**—(Optional) Displays a static sticky database entry for one of the static sticky entry types that follow.

Note Only the **show sticky database count** and **show sticky database static count** commands display the total number of entries in the system as Total Active Sticky Entries.

Note If you enable cookie insertion using the cookie insert command in sticky cookie configuration mode, the show sticky database static http-cookie command does not display the hash key.

- **active-conn-count min** *value1* **max** *value2*—(Optional) Displays sticky database entries within the specified connection count range.
- **client** *ip_address*—(Optional) Displays sticky database entries for the source IPv6 or IPv4 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.
- **ip-netmask** {**both** {**source** *ip_address2* **destination** *ip_address3*} | **destination** *ip_address4* | **source** *ip_address5*}—(Optional) Displays sticky database entries for both the source and destination addresses, the destination address only, or the source address only.
- **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, IPv4 only) Displays sticky database entries for the RTSP header value that you specify.
- **sip-header** *value6*—(Optional, IPv4 only) Displays sticky database entries for the SIP header value that you specify.
- **time-to-expire min** *value9* **max** *value10*—(Optional) Displays the sticky database entries within the specified time to expire range.
- **type**—(Optional) Displays sticky database entries for one of the following sticky group types:
  - **http-content**
  - **http-cookie**

  - **http-header**
  - **ip-netmask**
  - **layer4-payload**
  - **radius** (IPv4 only)
  - **rtsp-header** (IPv4 only)
  - **sip-header** (IPv4 only)

- **count**—(Optional) Displays the count for the sticky databae entries.
- **detail**—(Optional) Displays detailed statistics for the specified sticky database component. The **detail** option includes the sticky-hit-count field to display the total number of times that a sticky entry is hit..

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      -

Total Active Sticky Entries: 3000
```

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).

Table 5-12 Field Descriptions for the show sticky database Command Output (continued)

Field	Description
Sticky-Entry	Hashed value of the sticky entry in the database. For IP stickiness, displays the source or the destination address in dotted decimal notation.
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.
Total Active Sticky Entries	Total number of sticky entries that are inserted in the system at the time of executing the command. **Note** Regardless of the type of filters applied for the **show sticky database** command, the counter always displays the total number of entries in the system. For instance, if sticky group http-cookie has 0 entries and sticky group content has 3000 entries, the output for the **show sticky database type http-cookie** command is Total Active Sticky Entries: 3000

Generating and Displaying a Sticky Hash Entry

To correlate a known cookie or URL value with its corresponding sticky database entry (hash), use the **show sticky hash** command in Exec mode. This command allows you to generate the hash value from a known cookie or URL value using the same algorithm that is used by the URL and cookie hashing function. The syntax of this command is as follows:

show sticky hash *text*

The *text* argument indicates the cookie or URL text for which you want to calculate the hash value. Enter the cookie or URL value as an unquoted text string with no spaces and with a maximum of 1024 alphanumeric characters. If you want to include spaces in the text string, enclose the text string in quotation marks (“ ”).

For example, to generate the hash value for the cookie value 1.1.1.10, enter the following command:

```
host1/Admin# show sticky hash 1.1.1.10
Hash: 0x8a0937592c500bfb - 9946542108159511547
```

Now you can display the sticky database for a particular sticky group and match the generated hash with the sticky entry (hash) in the sticky database.

For example, to display the sticky database for the group STICKY_GROUP1, enter the following command:

```
host1//Admin# show sticky database group STICKY_GROUP1
sticky group : STICKY_GROUP1
type         : HTTP-COOKIE
timeout      : 1440          timeout-activeconns : FALSE
  sticky-entry         rserver-instance time-to-expire   flags
  --------------------+---------------+----------------+-------+
  9946542108159511547  SERVER1:80       86390            -
```

Displaying the Connections Related to a Sticky Entry

Each entry in the sticky database is associated with an internal sticky ID. You can retrieve this information with the **show sticky database** command. After you have retrieved the internal sticky ID, use the **show conn sticky** *internal_id* command in Exec mode to display all the connections that are linked to that sticky entry. This command is useful in identifying why a sticky entry does not timeout, for example. The syntax of this command is as follows:

show conn sticky *internal_id*

The *internal_id* argument indicates the internal identifier of a sticky entry in the sticky database.

The following example shows how to use the two above-mentioned commands to display all the connections associated with a particular sticky entry.

```
switch/Admin# show sticky database static detail | i internal
internal entry-id:           0x200006
internal entry-id:           0x200007
```

After you have obtained the internal sticky id, use the **show conn sticky** command to display all the connections linked to that sticky entry as follows:

```
switch/Admin# show conn sticky 0x200006

conn-id np dir proto vlan source              destination    state
-------+--+---+-----+----+------------------+---------------+------+
242     1  in  TCP   20  192.168.20.45:44425192.168.20.15:80 ESTAB
243     1  out TCP   40  192.168.40.28:80    192.168.20.45:44425 ESTAB
switch/Admin# show conn sticky 0x200007

conn-id np dir proto vlan source           destination       state

-------+--+---+-----+----+----------------+----------------+------+
switch/Admin#
```

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 reverse sticky entries	Total number of entries in the sticky database that currently have reverse sticky flows mapped to them.
Total active sticky conns	Total number of sticky connections that are currently active.

Table 5-13 Field Descriptions for the show stats sticky Command Output (continued)

Field	Description
Total static sticky entries	Total number of configured static entries that are in the sticky database.
Total sticky entries from global pool	Total number of entries in the sticky database from the global pool.
Total insertion failures due to lack of resources	Total number of sticky cookie insertion failures that resulted from insufficient resources.

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 ACE only leaves the standby ACE'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 {**active-conn-count min** *value1* **max** *value2* | **all** | **group** *group_name* | **time-to-expire min** *value9* **max** *value10* | **type** {**hash-key** *value* | **http-cookie** *value* | **ip-netmask** {**both** {**source** *ip_address2* **destination** *ip_address3*} | **destination** *ip_address4* | **source** *ip_address5*}}

The keywords and arguments are as follows:

- **active-conn-count min** *value1* **max** *value2*—Clears sticky database entries within the specified connection count range.
- **all**—Clears all dynamic sticky database entries in the context.
- **group** *group_name*—Clears all dynamic sticky database entries for the specified sticky group.
- **time-to-expire min** *value9* **max** *value10*— Clears the sticky database entries within the specified time to expire range.
- **type**—Clears sticky database entries for one of the following sticky group types:
  - **hash-key** *value*
  - **http-cookie** *value*
  - **ip-netmask** {**both** {**source** *ip_address2* **destination** *ip_address3*} | **destination** *ip_address4* | **source** *ip_address5*}

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 7, Configuring Firewall Load Balancing.