



Configuring Stickiness

This chapter describes how to configure stickiness (sometimes referred to as session persistence) on an Cisco Application Control Engine (ACE) module. It contains the following major sections:

- [Overview of Stickiness](#)
- [Configuration Requirements and Considerations](#)
- [Configuring IP-Address Stickiness](#)
- [Configuring HTTP-Cookie Stickiness](#)
- [Configuring HTTP-Header Stickiness](#)
- [Configuring an SLB Traffic Policy for Stickiness](#)
- [Displaying Sticky Configurations and Statistics](#)
- [Clearing Sticky Statistics](#)
- [Example of a Sticky Configuration](#)

Overview of Stickiness

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, as used here, 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 subsections:

- [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 once the connection is established, or the application may not require this until 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. Once you configure a sticky group and its attributes, you associate the sticky group with a **match** statement or a Layer 7 policy-map action in a Layer 7 SLB policy map. You can create a maximum of 4096 sticky groups in each context. Each sticky group that you configure on the ACE contains a series of parameters that determine:

- Sticky method
- Timeout
- Replication
- Cookie offset and other cookie-related attributes
- HTTP header offset and other header-related 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
- Cookie
- Hypertext Transfer Protocol (HTTP) header

The e-commerce application itself often dictates which of these methods is appropriate for a particular e-commerce vendor.

This section contains the following subsections:

- [IP Address Stickiness](#)
- [Cookie Stickiness](#)
- [HTTP Header Stickiness](#)

IP Address Stickiness

You can use the source IP address, the destination IP address, or both to uniquely identify individual clients and their requests for stickiness purposes based on their IP netmask. However, if an enterprise or a service provider uses a megaproxy to establish client connections to the Internet, the source IP address no longer is a reliable indicator of the true source of the request. In this case, you can use cookies or one of the other sticky methods to ensure session persistence.

Cookie Stickiness

Client *cookies* uniquely identify clients to the ACE and the servers providing 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.

Sticky Table

To keep track of sticky connections, the ACE uses a sticky table. Table entries include the following items:

- Sticky groups
- Sticky methods
- Sticky connections
- Real servers

The sticky table can hold a maximum of four million entries (four million simultaneous users). When the table reaches the maximum number of entries, additional sticky connections cause the table to wrap and the first users become unstuck from their respective servers.

The ACE uses a configurable timeout mechanism to age out sticky table entries. When an entry times out, it becomes eligible for reuse. High connection rates may cause the premature aging out of sticky entries. In this case, the ACE reuses the entries that are closest to expiration first.

Sticky entries can be either dynamic (generated by the ACE on-the-fly) or static (user-configured). When you create a static sticky entry, the ACE places the entry in the sticky table immediately. Static entries remain in the sticky database until you remove them from the configuration. You can create a maximum of 4096 static sticky entries in each context.

If the ACE takes a real server out of service for whatever reason (probe failure, no inservice command, or ARP timeout), the ACE removes from the database any sticky entries that are related to that server.

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 optional **backup** keyword of the same command, the backup server farm automatically becomes sticky, too, regardless of whether you configure the **sticky** option or not.

If all the servers in the primary server farm go down, the ACE sends all connections to the backup server farm. When the primary server farm comes back up (that is, at least one server becomes active), existing connections to the backup server farm continue to be serviced by the backup server farm. The ACE also sends new requests for existing sticky connections to the backup server farm.

The ACE sends all new connection requests to the primary server farm. Such requests include those for which there is no entry in the sticky table, the sticky entry aged out, or the ACE overwrote the sticky entry because the sticky table was full.

If you want to configure a sorry server farm and you want existing connections to revert to the primary server farm after it comes back up, do not use stickiness. For details about configuring backup server farms and sorry servers, see [Chapter 1, Configuring Traffic Policies for Server Load Balancing](#).

Configuration Requirements and Considerations

For best results, we recommend that you observe the following requirements and considerations when you configure stickiness on your ACE module.

- For each context in which you configure stickiness, you must:
 - Configure a resource class in the Admin context that you can associate with one or more contexts where you want to configure stickiness
 - Allocate a minimum non-zero percentage of resources to stickiness in the resource class using the **limit-resource** command
 - Associate one or more user contexts with the resource class

For details about configuring resource groups and allocating resources, refer to the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

- You can configure the same sticky group in multiple policies or virtual servers. In that case, the sticky behavior applies to all connections to any of those policies or class maps. These connections are referred to as *buddy connections* because, if you configure both policy or class map 1 and 2 with the same sticky group, a client stuck to server A through policy or class map 1 also will be stuck to the same server A through policy or class map 2.



Note

You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a Layer 4 policy map. This limit applies to all Layer 7 policy-map types. You configure regexes in:

- 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
-

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 subsections:

- [Configuration Requirements and Considerations](#)
- [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](#)

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 for illustration purposes, unless otherwise specified. For details on creating contexts, refer to the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the [“Configuration Requirements and Considerations”](#) section. For details about configuring a resources class and allocating resources, refer to the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create a sticky-IP group and enter sticky IP configuration mode.

```
host1/Admin(config)# sticky ip-netmask 255.255.255.0 address both  
GROUP1  
host1/Admin(config-sticky-ip)#
```

5. Configure a timeout for IP address stickiness.

```
host1/Admin(config-sticky-ip)# timeout 720
```

6. (Optional) Enable the timeout to override active connections.

```
host1/Admin(config-sticky-ip)# timeout activeconns
```

7. Enable the replication of sticky table information to the standby context in case of a switchover in a redundancy configuration. For details about configuring redundancy on an ACE, refer to the *Cisco Application Control Engine Module Administration Guide*.

```
host1/Admin(config-sticky-ip)# replicate sticky
```

Table 5-1 IP-Address Stickiness Configuration Quick Start (continued)**Task and Command Example**

8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of the virtual server.

```
host1/Admin(config-sticky-ip)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky aggregate-state
```

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 1, 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 1, 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 1, 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 1, Configuring Traffic Policies for Server Load Balancing](#).

14. (Recommended) Use the following command to display your IP address 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. (Optional) If necessary, 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](#)” section.

To create a sticky group for IP-address stickiness, use the **sticky ip netmask** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The full syntax of this command is:

```
sticky ip-netmask netmask address {source | destination | both} name
```

The keywords, arguments, and options are:

- **ip netmask** *netmask*—Specifies the network mask that the ACE applies to the IP address. Enter a network mask in dotted-decimal notation (for example, 255.255.255.0).
- **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*—Specifies a unique identifier of the sticky group. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a sticky group that uses IP address stickiness based on both the source IP address and the destination IP address, enter:

```
host1/Admin(config)# sticky ip netmask 255.255.255.0 address both  
GROUP1  
host1/Admin(config-sticky-ip)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky ip netmask 255.255.255.0 address both  
GROUP1
```

Configuring a Timeout for IP-Address Stickiness

The sticky timeout specifies the period of time that the ACE keeps (if possible) the IP-address sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection or receives a new HTTP GET on an existing connection matching 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:

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.

The syntax of this command is:

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 that, 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:

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 default behavior of the ACE to not replicating IP address sticky table entries, enter:

```
host1/Admin(config-sticky-ip)# no replicate sticky
```

Configuring Static IP-Address Sticky Table Entries

You can configure static sticky table entries based on the source IP address, destination IP address, or real server name and port. Static sticky-IP values remain constant over time and you can configure multiple static entries.



Note

When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

To configure static sticky-IP table entries, use the **static client** command in sticky-IP configuration mode. The syntax of this command varies according to the **address** option 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:

```
static client source ip_address rserver name [number]
```

If you configured the sticky group with the **destination** option, the syntax of this command is:

```
static client destination ip_address rserver name [number]
```

If you configured the sticky group with the **both** option, the syntax of this command is:

```
static client source ip_address [destination ip_address]{rserver name  
[number]}
```

The keywords, arguments, and options are:

- **source** *ip_address*—Specifies that the static entry be based on the source IP address. Enter an IP address in dotted decimal notation (for example, 192.168.12.15).
- **destination** *ip_address*—Specifies that the static entry be based on the destination IP address. Enter an IP address in dotted-decimal notation (for example, 172.16.27.3).

- **rserver** *name*—Specifies that the static entry be based on the real server name. Enter the name of an existing real server as an unquoted text string with no spaces and a maximum of 64 characters.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, to configure a static sticky entry based on the source IP address, the destination IP address, and the server name and port number, enter:

```
host1/Admin(config-sticky-ip)# static client source 192.168.12.15  
destination 172.16.27.3 rserver SERVER1 2000
```

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

```
host1/Admin(config-sticky-ip)# no static client source 192.168.12.15  
destination 172.16.27.3 rserver SERVER1 2000
```

Associating a Server Farm with an IP-Address Sticky Group

To complete a sticky group configuration, you must configure a server-farm entry for the group. To configure a server-farm entry for a sticky group, use the **serverfarm** command in sticky-IP configuration mode. The syntax of this command is:

```
serverfarm name1 [backup name2 [sticky] [aggregate-state]]
```

The keywords, arguments, and options are:

- *name1*—Identifier of an existing serverfarm that you want to associate with the sticky group. You can associate one serverfarm 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. If all real servers in the primary server farm go down, the ACE sends all connections to the backup server farm. If at least one real server returns to service in the primary server farm, the backup server farm continues to service existing connections and new requests over existing sticky connections. The ACE sends all new connection requests to the primary server farm.
- **aggregate-state**—(Optional, but recommended) Specifies that the state of the primary server farm is tied to the state of all the real servers in that server farm and in the backup server farm, if configured. The ACE declares the primary server farm down if all real servers in the primary server farm and all real servers in the backup server farm are down.

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 aggregate-state
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-ip) # no serverfarm
```

Configuring HTTP-Cookie Stickiness

This section describes how to configure stickiness based on HTTP cookies. The ACE learns cookie values from the:

- 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, it is common for clients to 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:

```
Set-Cookie: session_cookie = 123
```

The URL may be:

```
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 specifies 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 2048. 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 1, 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 1, 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 Layer 4 policy map. This limit applies to all Layer 7 policy-map types. You configure regexes in:

- 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
-

This section contains the following subsections:

- [HTTP-Cookie Stickiness Configuration Quick Start](#)
- [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](#)

HTTP-Cookie Stickiness Configuration Quick Start

Table 5-2 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 5-2.

Table 5-2 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 for illustration purposes, unless otherwise specified. For details on creating contexts, refer to the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the “[Configuration Requirements and Considerations](#)” section. For details about configuring a resources class and allocating resources, refer to the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create an HTTP cookie sticky group and enter sticky-cookie configuration mode.

```
host1/Admin(config)# sticky http-cookie cisco.com GROUP2  
host1/Admin(config-sticky-cookie)#
```

5. Configure a timeout for HTTP-cookie stickiness.

```
host1/Admin(config-sticky-cookie)# timeout 720
```

Table 5-2 HTTP Cookie Stickiness Configuration Quick Start (continued)**Task and Command Example**

-
6. (Optional) Enable the timeout to override active connections.
- ```
host1/Admin(config-sticky-cookie)# timeout activeconns
```
- 
7. Enable the replication of sticky table information to the standby context in case of switchover in a redundancy configuration. For details about configuring redundancy on an ACE, refer to the *Cisco Application Control Engine Module Administration Guide*.
- ```
host1/Admin(config-sticky-cookie)# replicate sticky
```
-
8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of the virtual server.
- ```
host1/Admin(config-sticky-cookie)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky aggregate-state
```
- 
9. (Optional) Enable cookie insertion to allow the ACE to insert a cookie in the Set-Cookie header of the response from the server to the client. Use this command when the server is not setting the appropriate cookie.
- ```
host1/Admin(config-sticky-cookie)# cookie insert browser-expire
```
-
10. (Optional) Configure the cookie sticky offset and length to instruct the ACE to use only a portion of the cookie (that part of the cookie that remains constant) for stickiness.
- ```
host1/Admin(config-sticky-cookie)# cookie offset 3000 length 1000
```
- 
11. (Optional) Configure the ACE to use an alternative cookie that appears in the URL string of the HTTP request from the client.
- ```
host1/Admin(config-sticky-cookie)# cookie secondary  
arrowpoint.com
```
-
12. (Optional) Configure one or more static sticky cookie entries.
- ```
host1/Admin(config-sticky-cookie)# static cookie-value corvette
rserver SERVER1 4000
```
- 
13. Configure a Layer 3 and Layer 4 SLB traffic policy. See [Chapter 1, Configuring Traffic Policies for Server Load Balancing](#).
- 
14. Configure a Layer 7 SLB traffic policy and associate the sticky group with the Layer 7 policy map. See [Chapter 1, Configuring Traffic Policies for Server Load Balancing](#).
-

**Table 5-2 HTTP Cookie Stickiness Configuration Quick Start (continued)**

| Task and Command Example |                                                                                                                                                                                                                                                                           |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15.                      | Associate the Layer 7 SLB traffic policy with the Layer 3 and Layer 4 SLB traffic policy. See <a href="#">Chapter 1, Configuring Traffic Policies for Server Load Balancing</a> .                                                                                         |
| 16.                      | Apply the Layer 3 and Layer 4 traffic policy to an interface using a service policy. See <a href="#">Chapter 1, Configuring Traffic Policies for Server Load Balancing</a> .                                                                                              |
| 17.                      | (Recommended) Use the following command to display your HTTP cookie sticky configuration. Make any modifications to your configuration as necessary, then reenter the command to verify your configuration changes.<br><br>host1/Admin# <b>show running-config sticky</b> |
| 18.                      | (Optional) If necessary, save your configuration changes to Flash memory.<br><br>host1/Admin# <b>copy running-config startup-config</b>                                                                                                                                   |

## 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](#)” section.

You can configure the ACE to learn a cookie from either the HTTP header of a client request or the Set-Cookie message sent by the server to a client. The ACE then uses the learned cookie to provide stickiness between a client and a server for the duration of a transaction. To configure the ACE to use HTTP cookies for stickiness, use the **sticky http-cookie** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is:

```
sticky http-cookie name1 name2
```

The keywords and arguments are:

- **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*—Specifies 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 cookie stickiness, enter:

```
host1/Admin(config)# sticky http-cookie cisco.com GROUP3
host1/Admin(config-sticky-cookie)#
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config)# no sticky http-cookie cisco.com GROUP3
```

## Configuring a Cookie Sticky Timeout

The sticky timeout specifies the period of time that the ACE keeps the HTTP-cookie sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.



### Note

The total sticky timeout is equal to the current time plus the timeout interval that you configure plus an internal update interval. The update interval can vary from 10 to 30 minutes depending on when you configured the sticky timeout and when the timeout update process reaches the particular sticky entry in the table.

To configure a sticky timeout, use the **timeout** *minutes* command in sticky-cookie configuration mode. The syntax of this command is:

```
timeout minutes
```

For the *minutes* argument, enter an integer from 1 to 65535. The default is 1440 minutes (24 hours).

For example, enter:

```
host1/Admin(config-sticky-cookie)# timeout 720
```

To reset the timeout to the default value of 1440 minutes, enter:

```
host1/Admin(config-sticky-cookie)# no timeout 720
```

## Enabling a Sticky Cookie Timeout to Override Active Connections

To specify that the ACE time out HTTP-cookie sticky table entries even if active connections exist after the sticky timer expires, use the **timeout activeconns** command in sticky-cookie configuration mode. The syntax of this command is:

```
timeout activeconns
```

For example, enter:

```
host1/Admin(config-sticky-cookie)# timeout activeconns
```

To restore the behavior of the ACE to the 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 that, 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:

```
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 default behavior of the ACE to 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 1, Configuring Traffic Policies for Server Load Balancing](#). For information about persistence rebalance, see “Enabling HTTP Persistence Rebalance” in [Chapter 1, 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:

```
cookie insert [browser-expire]
```

The **browser-expire** option allows the client’s browser to expire a cookie when the session ends.

For example, to enable cookie insertion and allow a browser to expire the cookie, enter:

```
host1/Admin(config-sticky-cookie)# cookie insert browser-expire
```

To disable cookie insertion, enter:

```
host1/Admin(config-sticky-cookie)# no cookie insert browser-expire
```

## Configuring the Offset and Length of an HTTP Cookie

An HTTP-cookie value may change over time with only a portion remaining constant throughout a transaction between the client and a server. You can configure the ACE to use the constant portion of a cookie to make persistent connections to a specific server. To define the portion of the cookie that you want the ACE to use, you specify cookie offset and length values. The ACE stores these values in the sticky table.

To configure the cookie offset and length, use the **cookie offset** command in sticky-cookie configuration mode. The syntax of this command is:

```
cookie offset number1 length number2
```

The keywords and arguments are:

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

For example, enter:

```
host1/Admin(config-sticky-cookie)# cookie offset 500 length 1000
```

To remove the cookie offset and length from the configuration, enter:

```
host1/Admin(config-sticky-cookie)# no cookie offset
```

## Configuring a Secondary Cookie

You can configure an alternative cookie name that appears in the URL string of the Web page on the server. The ACE uses this cookie to maintain a sticky connection between a client and a server and adds a secondary entry in the sticky table. To configure a secondary cookie, use the **cookie secondary** command in sticky-cookie configuration mode. The syntax of this command is:

```
cookie secondary name
```

Enter a cookie name as an unquoted text string with no spaces and a maximum of 64 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 there can exist only one unique real-server name for a given static cookie value.



### Note

---

When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

---

To configure a static cookie, use the **static cookie-value** command in sticky-cookie configuration mode. The syntax of this command is:

```
static cookie-value value rserver name [number]
```



### Note

---

You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a Layer 4 policy map. This limit applies to all Layer 7 policy-map types. You configure regexes in:

- 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
- 

The keywords, arguments, and options of this command are:

- *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*—Host name of an existing real server.
- *number*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.

For example, enter:

```
host1/Admin(config-sticky-cookie)# static cookie-value CORVETTE
rserver SERVER1 4000
```

To remove a static cookie from 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 serverfarm entry for a sticky group, use the **serverfarm** command in sticky-IP configuration mode. The syntax of this command is:

```
serverfarm name1 [backup name2 [sticky] [aggregate-state]]
```

The keywords, arguments, and options are:

- **name1**—Identifier of an existing serverfarm that you want to associate with the sticky group. You can associate one serverfarm 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**—(Optional, but recommended) Specifies that the state of the primary server farm is tied to the state of all the real servers in that server farm and in the backup server farm, if configured. The ACE declares the primary server farm down if all real servers in the primary server farm and all real servers in the backup server farm are down.

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 aggregate-state
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-cookie)# no serverfarm
```

## Configuring HTTP-Header Stickiness

When a client requests a URL from a server, the client sends an HTTP request to the ACE. The HTTP request contains a 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. You can specify any header name or select from one of the standard HTTP 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 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 “[Configuring an HTTP Parameter Map](#)” section in [Chapter 1, 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 2048. 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 1, 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 1, 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 Layer 4 policy map. This limit applies to all Layer 7 policy-map types. You configure regexes in:

- 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

This section contains the following subsections:

- [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](#)

## HTTP-Header Stickiness Configuration Quick Start

[Table 5-3](#) 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 5-3](#).

**Table 5-3 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 for illustration purposes, unless otherwise specified. For details on creating contexts, refer to the *Cisco Application Control Engine Module Administration Guide*.

2. Enter configuration mode.

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

3. Ensure that you have configured a resource class, allocated a minimum percentage of resources to stickiness, and associated one or more contexts where you want to configure stickiness with the resource class. See the [“Configuration Requirements and Considerations”](#) section. For details about configuring a resources class and allocating resources, refer to the *Cisco Application Control Engine Module Virtualization Configuration Guide*.

4. Create an HTTP-header sticky group and enter sticky-header configuration mode.

```
host1/Admin(config)# sticky http-header Host GROUP4
host1/Admin(config-sticky-header)#
```

5. Configure a timeout for HTTP-header stickiness.

```
host1/Admin(config-sticky-header)# timeout 720
```

6. (Optional) Enable the timeout to override active connections.

```
host1/Admin(config-sticky-header)# timeout activeconns
```

7. Enable the replication of HTTP-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, refer to the *Cisco Application Control Engine Module Administration Guide*.

```
host1/Admin(config-sticky-header)# replicate sticky
```

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

---

**Task and Command Example**

---

8. Associate a server farm with the sticky group for sticky connections and, optionally, tie the state of the backup server farm with the state of the virtual server.

```
host1/Admin(config-sticky-ssl)# serverfarm SFARM1 backup
BKUP_SFARM2 sticky aggregate-state
```

9. (Optional) Configure the HTTP-header sticky offset and length to instruct the ACE to use only a portion of the header (that part of the header that remains constant) for stickiness.

```
host1/Admin(config-sticky-header)# header offset 3000 length 1000
```

10. (Optional) Configure one or more static HTTP-header sticky entries.

```
host1/Admin(config-sticky-header)# static header Host rserver
SERVER1 4000
```

11. Configure a Layer 3 and Layer 4 SLB traffic policy. See [Chapter 1, 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 1, 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 1, 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 1, Configuring Traffic Policies for Server Load Balancing](#).

15. (Recommended) Use the following command to display your HTTP-header sticky configuration. Make any modifications to your configuration as necessary, then reenter the command to verify your configuration changes.

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

16. (Optional) If necessary, 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 an HTTP-header sticky group, be sure that you have allocated resources to stickiness as described in the “[Configuration Requirements and Considerations](#)” section.

To create an HTTP-header sticky group to enable the ACE to stick client connections to the same real server based on HTTP headers, use the **sticky http-header** command in configuration mode. You can create a maximum of 4096 sticky groups on an ACE. The syntax of this command is:

```
sticky http-header name1 name2
```

The keywords and arguments are:

- **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 below. For a list and description of the standard HTTP header, see [Table 5-4](#).

**Table 5-4 Standard HTTP Header Fields**

| Field Name             | Description                                                                                                                                                                                                                                             |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Accept</b>          | A semicolon-separated list of representation schemes (content type meta information values) that will be accepted in the response to the request.                                                                                                       |
| <b>Accept-Charset</b>  | The character sets 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. |
| <b>Accept-Encoding</b> | Restricts the content encoding that a user will accept from the server.                                                                                                                                                                                 |
| <b>Accept-Language</b> | The 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.                                                                      |

**Table 5-4 Standard HTTP Header Fields (continued)**

| <b>Field Name</b>    | <b>Description</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Authorization</b> | Specifies that the user agent wants to authenticate itself with a server, usually after receiving a 401 response.                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Cache-Control</b> | 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.                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Connection</b>    | Allows the sender to specify connection options.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Content-MD5</b>   | An 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.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Expect</b>        | Used by a client to inform the server about what behaviors the client requires.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>From</b>          | Contains the e-mail address of the person that controls the requesting user agent.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Host</b>          | The 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 <b>MUST</b> represent the naming authority of the origin server or gateway given by the original URL.                                                                                                                                                                                                                                                                                                              |
| <b>If-Match</b>      | 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. The purpose of this feature is to allow efficient updates of cached information with a minimum amount of transaction overhead. It is also used, on updating requests, to prevent inadvertent modification of the wrong version of a resource. As a special case, the value “*” matches any current entity of the resource. |

**Table 5-4 Standard HTTP Header Fields (continued)**

| Field Name               | Description                                                                                                                                                                                                                                                                                          |
|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Pragma</b>            | 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 <b>accept</b> field, a comma-separated list of entries, for which the optional parameters are separated by semicolons.              |
| <b>Referer</b>           | The address (URI) of the resource from which the URI in the request was obtained.                                                                                                                                                                                                                    |
| <b>Transfer-Encoding</b> | Indicates 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.                                                                                                                                             |
| <b>User-Agent</b>        | Information about the user agent, for example a software program originating the request. This information is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. |
| <b>Via</b>               | 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*—Specifies 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 group for HTTP-header stickiness, enter:

```
host1/Admin(config-sticky-header) # sticky http-header Host GROUP4
```

To remove the sticky group from the configuration, enter:

```
host1/Admin(config-sticky-header) # no sticky http-header Host GROUP4
```

## Configuring a Timeout for HTTP-Header Stickiness

The sticky timeout specifies the period of time that the ACE keeps the HTTP-header sticky information for a client connection in the sticky table after the latest client connection terminates. The ACE resets the sticky timer for a specific sticky-table entry each time that the module opens a new connection matching that entry.

To configure a sticky timeout, use the **timeout** command in sticky-header configuration mode. The syntax of this command is:

**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:

**timeout activeconns**



### 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)# timeout activeconns
```

To restore the behavior of the ACE to the default of not timing out HTTP-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 that, if a switchover occurs, the new active ACE can maintain existing sticky connections. To instruct the ACE to replicate HTTP-header sticky table entries on the standby ACE, use the **replicate sticky** command in sticky-header configuration mode. The syntax of this command is:

```
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 behavior of the ACE to the 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 HTTP header that you want the ACE to use, you specify HTTP-header offset and length values. The ACE stores these values in the sticky table.

To configure the HTTP-header offset and length, use the **header offset** command in sticky-header configuration mode. The syntax of this command is:

```
header offset number1 length number2
```

The keywords and arguments are:

- **offset** *number1*—Specifies the portion of the HTTP header 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 HTTP 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 HTTP 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 HTTP-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 HTTP-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 HTTP-header static entries, but there can exist only one unique real-server name for a given static HTTP-header sticky value.



### Note

---

When you configure a static entry, the ACE enters it into the sticky table immediately. You can create a maximum of 4096 static entries.

---

To configure a static HTTP header, use the **static header-value** command in sticky-header configuration mode. The syntax of this command is:

```
static header-value value rserver name [number]
```



#### Note

You can associate a maximum of 1024 instances of the same type of regular expression (regex) with a Layer 4 policy map. This limit applies to all Layer 7 policy-map types. You configure regexes in:

- 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

The keywords, arguments, and options of this command are:

- *value*—HTTP-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*—Host name of an existing real server.
- *number*—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 HTTP-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 a sticky group configuration, you must configure a server farm entry for the group. To configure a serverfarm entry for a sticky group, use the **serverfarm** command in sticky-IP configuration mode. The syntax of this command is:

```
serverfarm name1 [backup name2 [sticky] [aggregate-state]]
```

The keywords, arguments, and options are:

- *name1*—Identifier of an existing serverfarm that you want to associate with the sticky group. You can associate one serverfarm 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**—(Optional, but recommended) Specifies that the state of the primary server farm is tied to the state of all the real servers in that server farm and in the backup server farm, if configured. The ACE declares the primary server farm down if all real servers in the primary server farm and all real servers in the backup server farm are down.

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 aggregate-state
```

To disassociate a server farm from a sticky group, enter:

```
host1/Admin(config-sticky-header)# no serverfarm
```

# 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, enter:

1. Configure a Layer 7 class map and Layer 7 policy map, 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 cookie cookie1
cookie-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)#
```

2. Associate the sticky group as an action in the Layer 7 policy map.

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

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
```

4. Configure a Layer 3 and Layer 4 class map and policy map, then associate the class map with the policy map

```
host1/Admin(config)# class-map L4VIPCLASS
host1/Admin(config-cmap)# match virtual-address 192.168.1.10 tcp
port 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)#
```

5. Associate the Layer 7 policy map with the Layer 3 and Layer 4 policy map.

```
host1/Admin(config-pmap-c)# loadbalance policy L7SLBPOLICY
```

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
```

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 1, 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 subsections:

- [Displaying a Sticky Configuration](#)
- [Displaying Sticky Database Entries](#)
- [Displaying Sticky Statistics](#)

## Displaying a Sticky Configuration

To display the current sticky configuration, use the **show running-config** command in EXEC mode. The syntax of this command is:

```
show running-config sticky
```

The output of this command displays configured sticky groups and their attributes.

## Displaying Sticky Database Entries

The ACE stores sticky entries in a sticky database based on the following categories of information:

- Client IP address
- Sticky group ID
- HTTP-cookie value
- HTTP-header value
- Real-server name
- Static sticky database entries
- Sticky group type

To display sticky database entry information, use the **show sticky database** command in EXEC mode. The syntax of this command is:

```
show sticky database [static] [client ip_address | group name1 |
http-cookie value1 | http-header value2 | rserver name2 | type
{http-cookie | http-header | ip-netmask}]
```

The keywords, arguments, and options are:

- **static**—Displays static sticky database entries for one of the static sticky entry types that follow.
- **client** *ip\_address*—Displays sticky database entries for the source IP address of a client that you specify
- **group** *name1*—Displays sticky database entries for the sticky group name that you specify
- **http-cookie** *value1*—Displays sticky database entries for the HTTP-cookie value that you specify
- **http-header** *value2*—Displays sticky database entries for the HTTP-header value that you specify
- **rserver** *name2*—Displays sticky database entries for the real-server name that you specify
- **type**—Displays sticky database entries for one of the following sticky group types:
  - **http-cookie**
  - **http-header**
  - **ip-netmask**

For example, enter:

```
host1/Admin# show sticky database
sticky group : src-ip
type : IP
timeout : 1440 timeout-activeconns : FALSE
sticky-entry rserver-instance time-to-expire flags
-----+-----+-----+-----+
3232236541 rs1:0 86399 -
```

Table 5-5 describes the fields in the **show sticky database** command output.

**Table 5-5 Field Descriptions for the show sticky database Command Output**

| Field               | Description                                                                                                                                                                                                                           |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Sticky Group        | Name of the sticky group.                                                                                                                                                                                                             |
| Type                | Kind of sticky group (for example, HTTP-HEADER).                                                                                                                                                                                      |
| Timeout             | Timeout (in minutes) for the entry in the sticky table.                                                                                                                                                                               |
| Timeout-Activeconns | Indicates whether the <b>timeout activeconns</b> command is enabled or disabled. When enabled, this command times out sticky connections even when active connections exist. Possible values are: TRUE (enabled) or FALSE (disabled). |
| Sticky-Entry        | Hashed value of the sticky entry in the database.                                                                                                                                                                                     |
| Rserver-instance    | Name and, optionally, port of a real server associated with the sticky group (for example, rs1:81). If no port is configured for the real server in the serverfarm, 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    | Indicates whether the ACE replicates sticky entries to the peer ACE in a redundancy configuration.                                                                                                                                    |

## Displaying Sticky Statistics

You can display global sticky statistics for the current context by using the **show stats sticky** command in Exec mode. The syntax of this command is as follows:

```
show stats sticky
```

Table 5-6 describes the fields in the **show stats sticky** command output.

**Table 5-6** *Field Descriptions for the show stats sticky Command Output*

| Field                                       | Description                                                                                                              |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| Total Sticky Entries Reused Prior to Expiry | Total number of entries in the sticky database that the ACE was able to use more than once before the entries timed out. |
| Total Active Sticky Entries                 | Total number of entries in the sticky database that currently have flows mapped to them.                                 |
| Total Active Sticky Conns                   | Total number of sticky connections that are currently active.                                                            |
| Total Static Sticky Entries                 | Total number of configured static entries that are in the sticky database.                                               |

## Clearing Sticky Statistics

To clear all sticky statistics for the current context, use the **clear stats sticky** command in Exec mode. The syntax of this command is:

```
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, then you need to explicitly clear sticky statistics on both the active and the standby ACEs. Clearing statistics on the active module alone will leave the standby module's statistics at the old values.

# Clearing Dynamic Sticky Database Entries

To clear dynamic sticky database entries, use the **clear sticky database** command in Exec mode. The syntax of this command is:

```
clear sticky database { all | group group_name }
```

The keywords and arguments are:

- **all**—Clears all dynamic sticky database entries in the context
- *group\_name*—Clears all dynamic sticky database entries for the specified sticky group



## Note

---

This command does not clear static sticky database entries. To clear static sticky database entries, use the **no** form of the **static** command.

---

For example, to clear all dynamic sticky database entries for the sticky group named GROUP1, enter:

```
host1/Admin# clear sticky database GROUP1
```

## Example of a Sticky Configuration

This section provides a sample sticky configuration.

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

resource-class RC1
 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 type loadbalance first-match L7POLICY
 class L7CLASS
 sticky-serverfarm GROUP4
 insert-http Host header-value *.cisco.com

policy-map multi-match L4POLICY
 sequence interval 10
 class L4CLASS
 policy L7POLICY

interface vlan 193
 ip address 192.168.3.13 255.255.255.0
 service-policy input L4POLICY
 no shutdown
```

## Where to Go Next

If you want to configure firewall load balancing (FWLB), see [Chapter 6, Configuring Firewall Load Balancing](#).

