CHAPTER 8

Configuring Server Persistence Using Stickiness

This chapter describes how to configure server persistence using stickiness on the Cisco Application Control Engine (ACE) module.

This chapter contains the following sections:

- Information About Configuring Stickiness
- Configuring HTTP Cookie Stickiness
- Configuration Example for HTTP Cookie Stickiness
- Where to Go Next

Information About Configuring Stickiness

After reading this chapter, you should have a basic understanding of how the ACE provides server persistence using stickiness, and how to configure HTTP cookie stickiness.

When customers visit an e-commerce site, they usually start by browsing the site. Depending on the application, the site may require that the client remain connected (stuck) to one server as soon as the initial connection is established, or the application may require this action only when the client starts to create a transaction, such as building a shopping cart.

For example, after the client adds items to a shopping cart, it is important that all subsequent client requests are directed to the same real 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 server rather than duplicated across multiple servers.

E-commerce applications are not the only types of applications that require a sequence of client requests to be directed to the same real server. Any web applications that maintain client information may require this behavior, such as banking and online trading applications, or FTP and HTTP file transfers.

You can configure the ACE so that the same client can maintain multiple, simultaneous, or subsequent TCP or IP connections with the same real server for the duration of a session. This session persistence capability of the ACE is called stickiness. 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).

Depending on the configured server load-balancing policy, the ACE sticks a client to an appropriate server after the ACE determines 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. If the ACE determines that the client is not stuck to a particular server, it applies the normal load-balancing rules to the request.

To determine how a particular client is stuck to a specific web server and how an application distinguishes each client or a group of clients, the ACE supports the following sticky methods:

- Source and/or destination IP address—For stickiness, you can use the source IP address, the destination IP address, or both to uniquely identify individual clients and their requests based on their IP net masks. However, if an enterprise or service provider uses a mega-proxy (a free, anonymous web proxy service that can represent hundreds or thousands of different clients with a single source IP address) to establish client connections to the Internet, the source IP address is not a reliable indicator of the true source of the request. In this case, you can use another sticky method to ensure session persistence.
- Cookie—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 a server uses to deliver data to a web client, with the request that the client store the information. The cookie may be inserted into a response packet from the server or the ACE can insert the cookie. This information may include items that users have added to their shopping carts or travel dates that they have chosen. When the ACE examines a request for content and determines 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.
- Hypertext Transfer Protocol (HTTP) header—You can specify a header offset to provide stickiness based on a unique portion of the HTTP header.
- 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.
- RADIUS Attribute Stickiness—The ACE supports stickiness based on RADIUS attributes. The following attributes are supported for RADIUS sticky groups:
  - Framed IP
  - Framed IP and calling station ID
  - Framed IP and username
- RTSP Session Header Stickiness—The ACE supports stickiness based on the RTSP session header field. With RTSP header stickiness, you can specify a header offset to provide stickiness based on a unique portion of the RTSP header.
- SIP Call-ID Header Stickiness—The ACE supports stickiness based on the SIP Call-ID header field. SIP header stickiness requires the entire SIP header, so you cannot specify an offset.
- SSL Session-ID Stickiness—This feature allows the ACE to stick the same client to the same SSL server based on the SSL Session ID. 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 the handshake time translates directly into lower computational requirements for the server and reduced CPU utilization, and, therefore, increased SSL transactions per second (TPS).

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

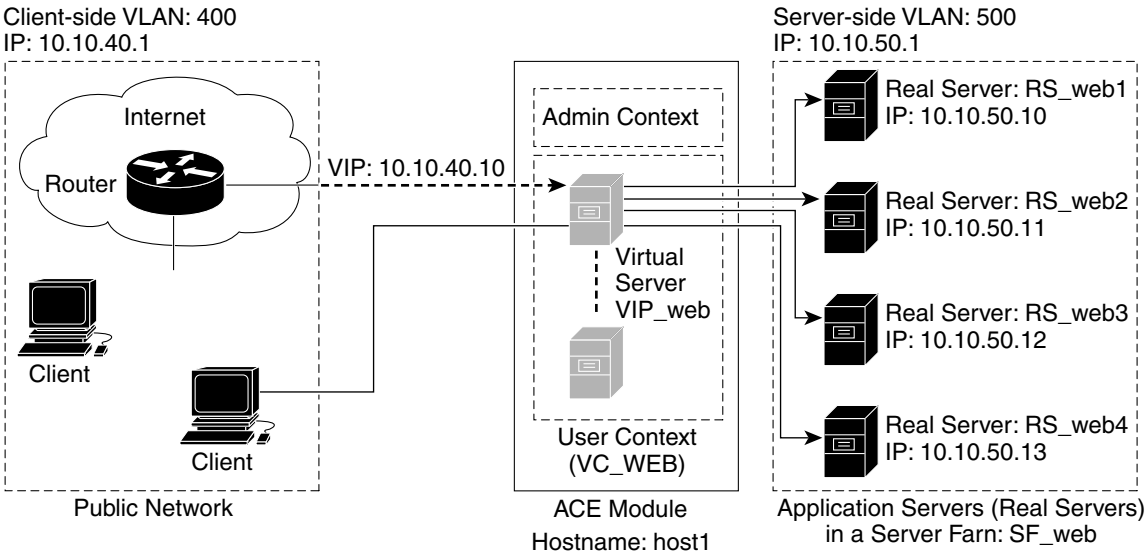
The ACE uses sticky groups for stickiness attributes. These attributes include the sticky method, timeout, replication, and attributes related to a particular sticky method.

To track sticky connections, the ACE uses a sticky table with information about sticky groups, sticky methods, sticky connections, and real 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.

Entries in the sticky table can be either dynamic (generated by the ACE as needed) or static (configured). When you create a static sticky entry, the ACE places the entry in the sticky table immediately, and it remains in the sticky database until you remove it from the configuration.

Figure 8-1 illustrates that, in a server load-balancing environment, requests from a client are stuck to real server RS_WEB4 in a session.

Figure 8-1 Client Requests Stuck to a Server



This chapter describes how to configure stickiness using the HTTP cookie sticky method. For information on how to configure stickiness using the IP address and HTTP header methods, see the *Server Load-Balancing Guide, Cisco ACE Application Control Engine.*

Configuring HTTP Cookie Stickiness

Procedure

	Command	Purpose
Step 1	**changeto** *context* **Example:** host1/Admin# changeto VC_WEB host1/VC_WEB#	Changes to the correct context if necessary. Check the CLI prompt to verify that you are operating in the desired context.
Step 2	**config** **Example:** host1/VC_WEB# config host1/VC_WEB(config)#	Enters configuration mode.
Step 3	**sticky http-cookie** *cookie_name group_name* **Example:** host1/VC_WEB(config)# sticky http-cookie COOKIE1 STICKYGROUP1 host1/VC_WEB(config-sticky-cookie)#	Creates an HTTP-cookie-type sticky group and enters cookie configuration mode.
Step 4	**cookie insert** [**browser-expire**] **Example:** host1/VC_WEB(config-sticky-cookie)# cookie insert browser-expire	Instructs the ACE to insert a cookie into the response packet from the server. The browser-expire option allows the browser to expire the cookie.
Step 5	**timeout** *number* **Example:** host1/VC_WEB(config-sticky-cookie)# timeout 1440	Configures a timeout for HTTP cookie stickiness.
Step 6	**serverfarm** *name* **Example:** host1/VC_WEB(config-sticky-cookie)# serverfarm SF_WEB	Associates a server farm with the sticky group and exits configuration mode.
Step 7	**exit** **Example:** host1/VC_WEB(config-sticky-cookie)# exit host1/VC_WEB(config)#	Exits sticky cookie configuration mode.
Step 8	**policy-map type loadbalance first-match** *policy_name* **class class-default** **Example:** host1/VC_WEB(config)# policy-map type loadbalance first-match PM_LB host1/VC_WEB(config-pmap-lb)# class class-default host1/VC_WEB(config-pmap-lb-c)#	Enters policy map load-balancing class configuration mode for the Layer 7 load-balancing policy.

	Command	Purpose
Step 9	**sticky-serverfarm** *group_name* **Example:** host1/VC_WEB(config-pmap-lb-c)# sticky-serverfarm STICKYGROUP1	Associates the sticky group with the Layer 7 load-balancing policy.
Step 10	**Ctrl-Z** **Example:** host1/VC_WEB(config-pmap-lb-c)# Ctrl-Z host1/VC_WEB#	Returns to Exec mode from any configuration mode.
Step 11	**show running-config sticky** **Example:** host1/VC_WEB# show running-config sticky	Displays the HTTP cookie configuration.
Step 12	**copy running-config startup-config** **Example:** host1/Admin# copy running-config startup-config	(Optional) Copies the running configuration to the startup configuration.

Configuration Example for HTTP Cookie Stickiness

The following example shows how to configure HTTP cookie stickiness. The commands that you have configured in this chapter appear in bold text.

```
switch/VC_WEB(config)# do show running config
Generating configuration....

access-list INBOUND line 8 extended permit ip any any

rserver host RS_WEB1
  description content server web-one
  ip address 10.10.50.10
  inservice
rserver host RS_WEB2
  description content server web-two
  ip address 10.10.50.11
  inservice
rserver host RS_WEB3
  description content server web-three
  ip address 10.10.50.12
  inservice
rserver host RS_WEB4
  description content server web-four
  ip address 10.10.50.13
  inservice

serverfarm host SF_WEB
  predictor hash header Accept
  rserver RS_WEB1 80
    inservice
  rserver RS_WEB2 80
    inservice
  rserver RS_WEB3 80
    inservice
  rserver RS_WEB4 80
    inservice
```

```
sticky http-cookie Cookie1 StickyGroup1
  cookie insert browser-expire
  timeout 3600
  serverfarm SF_WEB

class-map type management match-any REMOTE_ACCESS
  description Remote access traffic match
  2 match protocol ssh any
  3 match protocol telnet any
  4 match protocol icmp any
class-map match-all VS_WEB
  2 match virtual-address 10.10.40.10 tcp eq www

policy-map type management first-match REMOTE_MGMT_ALLOW_POLICY
  class REMOTE_ACCESS
    permit
policy-map type loadbalance first-match PM_LB
  class class-default
    sticky-serverfarm STICKYGROUP1
policy-map multi-match PM_MULTI_MATCH
  class VS_WEB
    loadbalance vip inservice
    loadbalance policy PM_LB

service-policy input REMOTE_MGMT_ALLOW_POLICY

interface vlan 400
  description Client connectivity on VLAN 400
  ip address 10.10.40.1 255.255.255.0
  access-group input INBOUND
  service-policy input PM_MULTI_MATCH
  no shutdown
interface vlan 500
  description Server connectivity on VLAN 500
  ip address 10.10.50.1 255.255.255.0
  no shutdown

domain DOMAIN1
add-object all

ip route 0.0.0.0 0.0.0.0 172.25.91.1
username USER1 password 5 $1$vAN9gQDI$MmbmjQgJPj45lxbtzXPpB1  role SLB-Admin domain
DOMAIN1
```

Where to Go Next

In this chapter, you have configured server persistence with a sticky group that uses the HTTP-cookie method. In the next chapter, you will configure secure sockets layer (SSL) security.