Configuring Sticky Cookies

This chapter provides information on CSS sticky using cookies.

For detailed information on services, sticky parameters and their uses, and Layer 3, Layer 4, and Layer 5 sticky, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Sticky Overview

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 customer become “stuck” to one server once the connection is established, or the application may not require this until the customer starts to build a shopping cart.

In either case, once the customer adds items to the shopping cart, it is important that all of the customer’s 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.

Stickiness is the association between a client and a server that the CSS maintains during a session. Stickiness enables transactions over the Web because the client must remain on the same server for the entire session. Depending on the content rule, the CSS "sticks" a client to an appropriate server after the CSS has determined which load-balancing method to use.

If the CSS determines that a client is already stuck to a particular service, then the CSS places the client request on that service, regardless of the load balancing criteria specified by the matched content rule. If the CSS determines that the client is not stuck to a particular service, it applies normal load balancing to the content request.

Client *cookies* uniquely identify clients to the services providing content. A cookie is a small data structure 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 state between the client and the server.

When the CSS examines a request for content and determines through content rule matching that the content is sticky, it examines any cookie or URL present in the content request. The CSS uses this information to place the content request on the appropriate server.

Advanced Load-Balancing Method Using Cookies

A content rule is "sticky" when additional sessions from the same user or client are sent to the same service as the first connection, overriding normal load balancing. By default, the advanced balancing method is disabled.

Use the **advanced-balance** command to specify an advanced load-balancing method for a content rule that includes stickiness. The **advanced-balance** command options (**cookies**, **cookieurl**, and **url**) use strings for sticking clients to servers. These options are beneficial when the sticky table limit is too small for your application requirements because the string methods do not use the sticky table.

The following sections provide configuration information for:

- Sticky Based on a Configured String in an HTTP Cookie Header, using the **advance-balanced cookies** command
- Sticky Based on a Cookie in a URL, using the **advance-balanced url** command
- Sticky Based on a Cookie in the HTTP Header or URL, using the **advance-balanced cookieurl** command

For information on additional advanced load-balancing methods including arrowpoint cookies, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Sticky Based on a Configured String in an HTTP Cookie Header

If the server returns a cookie that is static and uniquely identifies itself, use the **advanced-balance cookies** command. This command enables the content rule to stick a client to a server based on the configured string found in the HTTP cookie header. A content rule with a sticky configuration set to **advanced-balance cookies** requires all clients to enable cookies on their browser.

In the following configuration, the CSS looks for the cookie in the Cookie: field of the HTTP header:

1. The CSS looks for the configured string prefix, which is the cookie name. In this example, the string prefix in the content rule is MyCookie=.
2. If the CSS finds the prefix, then it looks for the value that matches one of the string values configured in one of the services. For example, the string value for service test 1 is server1. The CSS begins searching for the prefix and value at the beginning of the cookie field in the header and searches the entire field until the end of the field.

   If the HTTP header spans multiple packets, the CSS searches up to 5 packets by default; however, you can configure the CSS to search up to 20 packets (refer to the global **spanning-packets** command for more details).

3. If the CSS cannot find the string prefix or match the cookie value with one of the service string values, then the CSS load balances the request according to the configured balance method (roundrobin by default). For more details on what action the CSS takes when it cannot locate the cookie header or the specified cookie string, see the content rule mode **sticky-no-cookie-found-action** command.

```
configure

!************************** GLOBAL ***************************
  ip route 0.0.0.0 0.0.0.0 10.86.191.174 1

!************************ INTERFACE ************************
interface  3/2
  bridge vlan 2

!************************* CIRCUIT *************************
circuit VLAN1
  description "client vlan"

  ip address 10.86.191.161 255.255.255.240

circuit VLAN2
  description "server vlan"

  ip address 10.1.1.254 255.255.0.0

!The string value configured in the service must match the value of
the cookie for a particular server.
!************************* SERVICE *************************
service test1
  ip address 10.1.1.1
  string server1
  active

service test2
  ip address 10.1.1.2
  string server2
  active

service test3
  ip address 10.1.1.3
  string server3
  active
```

```
service test4
  ip address 10.1.1.4
  string server4
  active

!The string prefix must match the cookie name.  We recommend that you
include the '=' as part of the string prefix.
!************************** OWNER ***************************
owner test

  content stickyCookie
    advanced-balance cookies
    string prefix "MyCookie="
    add service test1
    add service test2
    add service test3
    add service test4
    port 80
    protocol tcp
active
```

Sticky Based on a Cookie in a URL

If the cookie is present in the URL instead of the cookie field of the HTTP header, use the **advanced-balance url** command. Some client applications do not accept cookies. When a site depends upon the information in the cookie, administrators sometimes modify the server application so that it appends the cookie data to the parameters section of the URL. The parameters typically follow a "?" at the end of the main data section of the URL.

In this configuration, the CSS functions in a similar manner as when using the **advanced-balance cookies** command; however, the CSS looks in the URL after the '?' for the cookie.

Using the full configuration of the "Sticky Based on a Configured String in an HTTP Cookie Header" section, the only difference is the **advanced-balance url** command in the content rule.

```
!************************** OWNER ***************************
owner test

  content stickyCookie
    advanced-balance url
```

```
    string prefix "MyCookie="
    add service test1
    add service test2
    add service test3
    add service test4
    port 80
    protocol tcp
active
```

Sticky Based on a Cookie in the HTTP Header or URL

If the cookie could be in either the cookie field of the HTTP header or the URL, use the **advanced-balance cookieurl** command.

In this configuration, the CSS searches for the cookie first in the cookie field of the HTTP header. If the cookie field does not exist, then the CSS looks for the cookie in the URL. This command is intended for applications where some clients cannot accept cookies but others can.

Using the full configuration of the "Sticky Based on a Configured String in an HTTP Cookie Header" section, the only difference is the **advanced-balance cookieurl** command in the content rule.

```
!************************** OWNER ***************************
owner test

  content stickyCookie
    advanced-balance cookieurl
    string prefix "MyCookie="
    add service test1
    add service test2
    add service test3
    add service test4
    port 80
    protocol tcp
active
```