CHAPTER 12

Configuring HTTP Header Load Balancing

This chapter describes how to configure HTTP header load balancing by creating an HTTP header field group and configuring HTTP header fields. Information in this chapter applies to all CSS models except where noted.

This chapter contains the following major sections:

- HTTP Header Load-Balancing Overview
- HTTP Header Load Balancing Configuration Quick Start
- Creating a Header Field Group
- Describing the Header Field Group
- Configuring a Header-Field Entry
- Associating a Header Field Group with a Content Rule
- Showing a Content Rule Header Field Group Configuration
- Showing Header Field Groups
- Header Field Group Configuration Examples

Note You must enable service remapping in order for HTTP header load balancing to work properly. For information on the service remapping feature, see Chapter 10, Configuring Content Rules.

HTTP Header Load-Balancing Overview

Configuring HTTP header load balancing enables the CSS to inspect incoming content requests for HTTP header fields. HTTP header load balancing allows the CSS to make load-balancing decisions based on the HTTP header field information and then direct content requests to the servers designed to handle the type of content being requested.

The CSS can direct content requests to specific servers based on different types of browsers or different representations of the same content that has been modified for end users. For example, a client running a hand-held personal organizer may want the same content as a client using a PC, but with fewer graphics. Users may want to see content in only a particular language.

Using HTTP header load balancing eliminates the need to duplicate various forms of the same content across all of the servers, thus freeing up valuable server space. In addition to dividing the server farm for different types of clients, you can also use HTTP header load balancing to bypass noncacheable traffic and prioritize client browser traffic from search engine services.

Using HTTP Header Load Balancing in a Content Rule

Using an HTTP header field group in a Layer 5 content rule enables a rule to be more specific than if the rule defined just a URL. The HTTP header field group makes the content match more specific. Because content rules are hierarchical, if a request for content matches more than one rule, the characteristics of the most specific rule apply to the flow. This hierarchy for Layer 5 rules is defined below. The CSS uses this order of precedence to process requests for the content, with 1 being the highest match and 4 being the lowest match.

1. Domain name, IP address, protocol, port, URL, HTTP header field group
2. IP address, protocol, port, URL, HTTP header field group
3. Domain name, protocol, port, URL, HTTP header field group
4. Protocol, port, URL, HTTP header field group

HTTP Header Load Balancing Configuration Quick Start

Table 12-1 provides a quick overview of the steps required to create and configure HTTP header load balancing. Each step includes the CLI command required to complete the task. For a complete description of each feature and all the HTTP header load-balancing configuration options, see the sections following Table 12-1.

Ensure that you have already created and configured a service and owner for the content rules. The command examples in Table 12-1 create HTTP load balancing for owner arrowpoint and content rule rule1.

Table 12-1 HTTP Load Balancing Configuration Quick Start

Task and Command Example
1. Enter config mode by typing **config**. `(config)#`

2. Create a header field group. This example creates the group ppilot.
`(config)# header-field-group ppilot`
`(config-header-field-group[ppilot])#`

3. Describe the header field group (optional). `(config-header-field-group[ppilot])# description "ppilot content"`
4. Configure header field entries by defining a header, field, name, field type, and operator. `(config-header-field-group[ppilot])# header-field palm1 user-agent contain "MSIE" 20`
5. Associate the header field group with a content rule. `(config-owner-content[arrowpoint-rule1])# header-field-rule ppilot`
6. (Recommended) Display the header field group to verify your configuration. `(config)# show header-field-group`

The following running-configuration example shows the results of entering the commands in Table 12-1.

```
!******************** HEADER FIELD GROUP ********************
header-field-group ppilot
  description "ppilot content"
  header-field palm1 user-agent contain "MSIE" 20

!************************** OWNER ***************************
owner arrowpoint
  address "200 Beaver Brook Road, Boxborough, MA 01719"

  content rule1
    vip address 192.1.1.100
    protocol tcp
    port 80
    add service server1
    header-field-rule ppilot
```

Creating a Header Field Group

Header field group configuration mode allows you to create a header field group. A header field group contains a list of user-defined header field entries used by the CSS content rule lookup process. A group can contain several header-field entries.

Note The CSS supports a maximum number of 1024 header field groups, with a maximum of 4096 header field entries.

Note When there is more than one header field entry in a group, each header field entry must be successfully matched before the CSS uses the associated content rule.

To create a header field group or to access header field group configuration mode, use the **header-field-group** command from any configuration mode except boot and RMON modes.

The prompt changes to (config-header-field-group [group_name]). You can also use this command in header-field-group mode to access another group.

The syntax for this mode-transition command is:

header-field-group *group_name*

Enter the *group_name* of the header-field group you want to create. You must define a unique name for each header field group so different content rules can use the groups. Enter a text string with a maximum of 32 characters. To see an existing list of header-field groups, use the **header-field-group ?** command.

For example, enter:

```
(config)# header-field-group ppilot
(config-header-field-group[ppilot])#
```

To remove a header-field group, use the **no header-field-group** command. For example, enter:

```
(config)# no header-field-group ppilot
```

Describing the Header Field Group

To provide a description for a header field group, use the **description** command. The syntax for this command is:

description "*text***"**

Enter the text as a quoted text string with a maximum length of 64 characters.

For example,

```
(config-header-field-group[ppilot])# description "ppilot content"
```

To remove a description for a header-field group, enter:

```
(config-header-field-group[ppilot])# no description
```

Configuring a Header-Field Entry

Configure a header-field entry in a header-field group to specify a field in an HTTP header and an operator to perform a function on that field. When the CSS receives an HTTP content request, it inspects the HTTP header field specified in the header-field entry and performs the function on that field specified in the *operator* variable. The CSS uses the results of the header-field operation to load balance all subsequent packets in the flow.

A header field entry contains a header field name, field type to be used, an operation to be performed, the header-string to be searched for, and an optional search length.

If a header field group contains multiple header field entries, a content request must match each entry for the rule to be used.

Note The CSS supports a maximum number of 1024 header field groups, with a maximum of 4096 header field entries.

Use the **header-field** command to define a header field entry in a header field group. The syntax for this command is:

header-field *name field_type* {*custom_string*} *operator* {*header_string* {*search_length*}}

The variables and options are:

- *name* - The name uniquely identifies the header field entry. Enter the name as a string from 1 to 31 characters. You must define a header field entry name because the CSS can use the same field type multiple times in a header field group.
- *field_type* - The field type includes one of the following:
  - **user-agent** - Information about the user agent, for example a software program originating the request. This information is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations.

- **language -** 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.
- **host -** 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 MUST represent the naming authority of the origin server or gateway given by the original URL.
- **cache-control -** Directives that must be obeyed by all caching mechanisms along the request/response chain. The directives specify behavior intended to prevent caches from adversely interfering with the request or response.
- **pragma -** Pragma directives understood by servers to whom the directives are relevant. The syntax is the same as for other multiple-value fields in HTTP, for example, the **accept** field, a comma-separated list of entries, for which the optional parameters are separated by semicolons.
- **encoding -** The encoding mechanism used.
- **charset -** 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 representing documents in those character sets.
- **connection -** Options for the connection.
- **referer -** The address (URI) of the resource from which the URI in the request was obtained.
- **accept -** A semicolon-separated list of representation schemes (content type metainformation values) that will be accepted in the response to this request.
- **request-line** - When you attempt to access an Internet resource using your browser (for example, http://www.cisco.com), the browser issues a request for the resource in an HTTP request message. The request line contains the HTTP method (GET, HEAD, or PUSH), the request URI, and the HTTP version. A uniform resource identifier (URI) consists of a string of alphanumeric and sometimes special characters that identify a resource on the Internet. The request line is a required HTTP request message field.

For example, suppose an HTTP request contains the following URI:

/cgi-bin/some-app.pl?session=12345678912345678&user=CiscoUser&action=LoadBalanceMe&foo=bar

By creating a header field group and header field rules, you can configure a CSS to make a content rule selection based on a string in the URI. For example, you can configure a CSS to make a content rule selection based on the string LoadBalanceMe in the above URI using the following configuration:

```
header-field-group url
    header-field urlString request-line contain "LoadBalanceMe"
owner arrowpoint
    content rule UrlString
        vip address 192.168.128.151
        protocol tcp
        port 80
        url "/*"
        add service server1
        add service server2
        header-field-rule url
        active
    content rule2
        vip address 192.168.128.151
        protocol tcp
        port 80
        url "/*"
        add service server21
        add service server22
        active
```

- **cookies -** The configured string found in the HTTP header that the CSS uses to stick the client to the server.
- **msisdn** - The header field type for Wireless Application Protocol (WAP). HTTP requests from certain wireless gateways contain the MSISDN field in the HTTP header. By configuring the **msisdn** header field type in a header field group, you can load balance wireless requests. See the "Example 3. Wireless configuration that load balances HTTP requests based on the MSISDN header field" section later in this chapter.

  You can use this option alone or with the **advanced-balance wap-msisdn** sticky command. See the "Specifying an Advanced Load-Balancing Method for Sticky Content" section in Chapter 11, Configuring Sticky Parameters for Content Rules.

  - **custom** - Field type keyword that indicates a user-defined header field. Use the **custom** header field with the *custom_string* variable to perform HTTP header matching on the Name: value in the HTTP header field.

- *custom_string* - A case-insensitive, alphanumeric string used with the **custom** field type. Enter a quoted alphanumeric string from 1 to 31 chartacters. You cannot use the following ASCII characters: control characters (decimal 0 through 31), DEL (decimal 127), and special characters (,), <, >, @, “,”, ;, \, “, /, [,], ?, =, {, }, SP, and HT. You can define a maximum of 16 unique custom header fields for each CSS.

  Note You cannot configure a custom header field that is identical to one of the currently predefined header field tags.

  The **custom** header field uses the current header-field matching rules; that is, it does not add any new matching rules. See the “Example 4. Configuration that load-balances HTTP requests based on user-defined header fields” section.

  For example:

```
header-field customtag2 custom "Peak" contain "CD"
```

- *operator* - Enter one of the following operators:
  - **exist|not-exist** - Use the **exist** and **not-exist** operators to check whether a specified header field exists in a content request header.
  - **equal|not-equal** {**“***header_string***”**} - Use the **equal** and **not-equal** operators to match a defined *header_string* to the contents of the specified header field, and to determine whether it is equal to the header string. Enter the *header_string* as a case-insensitive, quoted text string with a maximum of 31 characters including spaces.
  - **contain|not-contain** {**“***header_string***”** {*search_length*}} - Use the **contain** and **not-contain** operators to match the configured *header_string* to a substring in the contents of the specified field type, and to determine whether its contents contain the *header_string*. Enter the *header_string* as a case-insensitive, quoted text string with a maximum of 31 characters including spaces.

    You may include an optional *search_length* to define the header field portion to be used for the operation. If you do not define a search length, the CSS uses the entire header field (delimited by a CR and LF) for the operation. To define the search length, enter a number from 0 to 1024.

For example, enter:

```
(config-header-field-group[ppilot])# header-field palm1 user-agent
contain "MSIE" 20

(config-header-field-group[ppilot])# header-field palm2 user-agent
contain "palm"
```

To remove a header field entry, use the **no header-field** command. For example, enter:

```
(config-header-field-group[ppilot])# no header-field palm1
```

Note To completely delete a custom header field from a CSS and to make it available for reuse, you must remove all instances of that custom header field from all header-field groups on the CSS using the **no header-field** command.

Associating a Header Field Group with a Content Rule

To associate a header field group with a content rule, and optionally assign a weight value to the header field group, use the **header-field-rule** command. Use weights to allow the CSS to prefer one content rule over a similar content rule. For example, you want to load balance French clients to a specific server, and you also want to differentiate the clients using Microsoft Internet Explorer from those using Netscape Navigator. If it is more important to direct the French clients to a specific server than to direct them to a server based on whether they are using Internet Explorer or Netscape Navigator, then you need to weight the "French" content rule higher than the "Internet Explorer/Netscape" content rule.

Note The CSS supports only one header field group for each content rule.

The syntax for this content mode command is:

header-field-rule *name* {*weight number*}

The variables are:

- *name* - The name of the header field group used with the content rule. To see a list of groups, use the **header-field-rule ?** command.
- **weight** *number* - The weight you want to assign to the header field group. Enter a number from 0 to 1024. The default weight is 0.

For example, enter:

```
(config-owner-content[arrowpoint-rule1])# header-field-rule french
weight 3
```

To remove the header field group from the content rule, enter:

```
(config-owner-content[arrowpoint-rule1])# no header-field-rule
```

Showing a Content Rule Header Field Group Configuration

Use the **show rule header-field** command to display information about the header field group associated with a content rule. For example, to display information about the header-field rule and group associated with a specific content rule, enter:

```
(config-owner-content[arrowpoint-rule1])# show rule header-field
```

Showing Header Field Groups

Use the **show header-field-group** command to display the configuration for all header field groups or a specific group. This command is available in all modes.

The syntax and options for this command are:

- **show header-field-group** - Displays a summary of all configured header field groups
- **show header-field-group all** - Displays detailed information about all configured header field groups
- **show header-field-group** *name* - Displays detailed information about a specific header field group

For example, to show a summary of all configured header field groups, enter:

```
(config)# show header-field-group
```

Table 12-2 describes the fields in the **show header-field-group** command output.

Table 12-2 Field Descriptions for the show header-field-group Command Output

Field	Description
Header field group	The name of the header-field group
Description	The configured description for the header-field group

Header Field Group Configuration Examples

When configuring header field groups, it is good practice to configure rules to be specific in rule matching (as shown in configuration Example 2). If the rules are not specific enough, the CSS may match a client request to the first rule it finds, and the first-matched rule may change on subsequent requests.

This section contains the following configuration examples:

- Example 1. Configuration that is ambiguous in rule-matching capabilities
- Example 2. Configuration that broadens the rule-matching capabilities
- Example 3. Wireless configuration that load balances HTTP requests based on the MSISDN header field
- Example 4. Configuration that load-balances HTTP requests based on user-defined header fields

Example 1. Configuration that is ambiguous in rule-matching capabilities

Example 1 shows a configuration that is ambiguous. If a client request specifies the language as French and the user-agent as Netscape, this request may match equally to ruleA2 or ruleA3. In this example, the rule matching may not be consistent. One way to solve the ambiguity between ruleA2 and ruleA3 is to use different weight values (not shown in the configuration example). If you assign a weight value of 10 to header field group B when you associate it with ruleA2, the CSS will always use ruleA2 as a match to the client request. Another method is to configure more specific rules as shown in configuration Example 2.

```
! ***************** HEADER FIELD GROUP ********************

header-field-group A
    header-field ua1 language equal "en"

header-field-group B
    header-field ua2 language equal "fr"

header-field-group C
    header-field-group ua3 user-agent contain "Netscape"
! ********************** OWNER ***************************

owner arrowpoint
    content ruleA
        protocol tcp
        vip address 192.168.128.151
        port 80
        url "/*"
        add service server1
        add service server2

    content ruleA1
        protocol tcp
        vip address 192.168.128.151
        port 80
        url "/*"
        header-field-rule A
        add service server11
        add service server12

    content ruleA2
        protocol tcp
        vip address 192.168.128.151
        port 80
        url "/*"
        header-field-rule B
        add service server21
        add service server22

    content ruleA3
        protocol tcp
        vip address 192.168.128.151
        port 80
        url "/*"
        header-field-rule C
        add service server31
        add service server32
```

Example 2. Configuration that broadens the rule-matching capabilities

Example 2 shows the same configuration as Example 1, only modified to broaden the rule-matching capabilities. Each content rule is specific. The client request specifying the language as French and the user-agent as Netscape will match only content rule ruleA2.

```
! ***************** HEADER FIELD GROUP ********************

header-field-group A
    header-field ua1 language equal "en"
    header-field ua2 user-agent contain "Netscape"

header-field-group B
    header-field ua3 language equal "fr"
    header-field ua4 user-agent contain "Netscape"

header-field-group C
    header-field ua5 language equal "en"
    header-field ua6 user-agent not-contain "Netscape"

header-field-group D
    header-field ua7 language equal "fr"
    header-field ua8 user-agent not-contain "Netscape"

! ********************** OWNER **************************

owner arrowpoint
    content ruleA
        protocol tcp
        vip address 192.168.128.151
        port 80
        url "/*"
        add service server1
        add service server2

    content ruleA1
        protocol tcp
        vip address 192.168.128.151
        port 80
        url "/*"
        header-field-rule A
        add service server11
        add service server12
```

```
content ruleA2
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/*"
    header-field-rule B
    add service server21
    add service server22

content ruleA3
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/*"
    header-field-rule C
    add service server31
    add service server32

content ruleA4
    protocol tcp
    vip address 192.168.128.151
    port 80
    url "/*"
    header-field-rule D
    add service server41
    add service server42
```

Example 3. Wireless configuration that load balances HTTP requests based on the MSISDN header field

Example 3 shows a configuration that makes load-balancing decisions based on whether a client is a wireless client. Wireless devices use the Wireless Application Protocol (WAP). When a wireless client sends a request for content, the WAP protocol gateway (a device that translates requests from the WAP protocol stack to the WWW protocol stack) generates the MSISDN field and adds it to the HTTP header. You can test for the presence of the MSISDN header field using the **exist** and **not-exist** operators in the header field entry of a header field group. Then, you can make load-balancing decisions based on the presence or absence of the MSISDN header field. For details on configuring the MSISDN header field type, see the "Configuring a Header-Field Entry" section earlier in this chapter.

In the following example, any TCP port 80 traffic destined for VIP 192.168.128.151 that has the MSISDN field in the HTTP header will match on the content rule ruleWap. Any TCP port 80 traffic destined for 192.168.128.151 that does not have the MSISDN field in the HTTP header will match on the content rule ruleNoWap.

```
header-field-group wap
   header-field 1 msisdn exist

owner arrowpoint
   content ruleWap
     vip address 192.168.128.151
     protocol tcp
     port 80
     url "/*"
     add service server1
     add service server2
     header-field-rule wap
     active

   content ruleNoWap
     vip address 192.168.128.151
     protocol tcp
     port 80
     url "/*"
     add service server21
     add service server22
     active
```

Note You can use the MSISDN header field with the **advanced-balance wap-msisdn** command to configure wireless users for e-commerce applications. For details on configuring a wireless user, see the "Configuring Wireless Users for E-Commerce Applications" section in Chapter 11, Configuring Sticky Parameters for Content Rules.

Example 4. Configuration that load-balances HTTP requests based on user-defined header fields

Example 4 shows a configuration that enables a CSS to make load-balancing decisions based on custom header fields. You can define a maximum of 16 unique custom header fields on one CSS. However, you can define more than one custom header field in each header field group. If you configure an identical custom header field in more than one header field group, the custom header field counts as only one of the 16 maximum custom header fields that you can configure on that CSS.

In the following example, two unique custom header fields are configured. Any TCP port 80 traffic destined for VIP 192.168.128.15 with the tag "Acme" *and* with the tag "Peak" that contains the string "CD" will match on the content rule HTTPrule1. Any TCP port 80 traffic destined for VIP 192.168.128.15 with the tag "Peak" that contains the string "CD" will match on the content rule HTTPrule2.

The CSS finds the best match based on all the commands configured in the content rules. For more information about configuring content rules, see Chapter 10, Configuring Content Rules.

```
header-field-group group1
   header-field customtag1 custom "Acme" exist
   header-field customtag2 custom "Peak" contain "CD"

header-field-group group2
   header-field customtag1 custom "Peak" contain "CD"

owner arrowpoint

   content HTTPrule1
     vip address 192.168.128.15
     protocol tcp
     port 80
     url "/*"
     add service server1
     add service server2
     header-field-rule group1
     active

content HTTPrule2
     vip address 192.168.128.15
     protocol tcp
     port 80
     url "/*"
     add service server3
     add service server4
     header-field-rule group2
     active
```