Configuring Virtual Servers, Maps, and Policies

This chapter describes how to configure content switching and contains these sections:

Configuring Virtual Servers

This section describes how to configure virtual servers and contains these sections:

Note When a virtual server is configured with an IP address, it will start replying to ARP requests for that specific IP, even if it is still out of service. This is important especially when migrating operational virtual servers from existing devices over to the CSM. Make sure that you never have a virtual server on the CSM configured with the same IP of another device in the same network.

Virtual servers represent groups of real servers and are associated with real server farms through policies. Configuring virtual servers requires that you set the attributes of the virtual server specifying the default server farm (default policy) and that you associate other server farms through a list of policies. The default server farm (default policy) is used if a request does not match any SLB policy or if there are no policies associated with the virtual server.

Before you can associate a server farm with the virtual server, you must configure the server farm. For more information, see the "Configuring Server Farms" section on page 5-1. Policies are processed in the order in which they are entered in the virtual server configuration. For more information, see the "Configuring Policies" section on page 6-11.

You can configure each virtual server with a pending connection timeout to terminate connections quickly if the switch becomes flooded with traffic. This connection applies to a transaction between the client and server that has not completed the request and reply process.

In a service provider environment in which different customers are assigned different virtual servers, you may need to balance the connections to prevent an individual server from absorbing most or even all of the connection resources on the CSM. You can limit the number of connections going through the CSM

to a particular virtual server by using the VIP connection watermarks feature. With this feature, you may set limits on each virtual server, allowing a fair distribution of connection resources among all virtual servers.

Note You can configure a single virtual server to operate at either Level 4 or Level 7. To configure a virtual server to operate at Level 4, specify the server farm (default policy) as part of the virtual server configuration. (See Step 3 in the following task table.) To configure a virtual server to operate at Level 7, add SLB policies in the configuration of the virtual server. (See Step 7 in the following task table.)

The CSM can load-balance traffic from any IP protocol. When you configure a virtual server in virtual server submode, you must define the IP protocol that the virtual server will accept.

Note Although all IP protocols have a protocol number, the CSM allows you to specify TCP or UDP by name instead of requiring you to enter their numbers.

Configure the virtual server in the virtual server configuration submode.

To configure virtual servers, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# owner` *owner-name* `address` *street-address-information* `billing-info` *billing-address-information* `email-address` *email-information* `maxconns` *1:MAXULONG*	Restricts access to virtual servers to a specific owner object.
Step 2	`Router(config-module-csm)# vserver` *virtserver-name*	Identifies the virtual server and enters the virtual server configuration mode[1, 2].
Step 3	`Router(config-slb-vserver)# vs-owner` *owner-name* `maxconns` *1:MAXULONG*	Sets the owner object name for this virtual server.
Step 4	`Router(config-slb-vserver)# virtual` *ip-address [ip-mask] protocol port-number* [`service ftp`]	Sets the IP address for the virtual server optional port number or name and the connection coupling and type[2]. The *protocol* value is **tcp**, **udp**, **Any** (no port number is required), or a *number* value (no port number is required).
Step 5	`Router(config-slb-vserver)# serverfarm` *serverfarm-name*	Associates the default server farm with the virtual server[2 3]. Only one server farm is allowed. If the server farm is not specified, all the requests not matching any other policies will be discarded.
Step 6	`Router(config-slb-vserver)# sticky` *duration*	(Optional) Configures connections from the client to use the same real server[2 3]. The default is sticky off.
Step 7	`Router(config-slb-vserver)# sticky` *group-number* `reverse`	(Optional) Ensures that the CSM changes connections in the appropriate direction back to the same source.
Step 8	`Router(config-slb-vserver)# client` *ip-address network-mask* [`exclude`]	(Optional) Restricts which clients are allowed to use the virtual server[2 3].
Step 9	`Router(config-slb-vserver)# slb-policy` *policy-name*	(Optional) Associates one or more content switching policies with a virtual server[2].

	Command	Purpose
Step 10	`Router(config-slb-vserver)# inservice`	Enables the virtual server for use by the CSM[2].
Step 11	`Router# show module csm slot vserver [details]`	Displays information for virtual servers defined for content switching.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.
3. These parameters refer to the default policy.

This example shows how to configure a virtual server named barnett, associate it with the server farm named bosco, and configure a sticky connection with a duration of 50 minutes to sticky group 12:

```
Router(config)# mod csm 2
Router(config-module-csm)# sticky 1 cookie foo timeout 100
Router(config-module-csm)# exit
Router(config-module-csm)#
Router(config-module-csm)# serverfarm bosco
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)#
Router(config-slb-sfarm)# vserver barnett
Router(config-slb-vserver)# virtual 10.1.0.85 tcp 80
Router(config-slb-vserver)# serverfarm bosco
Router(config-slb-vserver)# sticky 50 group 12
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
Router(config-module-csm)# end
```

This example shows how to configure a virtual server, named vs1, with two policies and a default server farm when client traffic matches a specific policy. The virtual server will be load balanced to the server farm attached to that policy. When client traffic fails to match any policy, the virtual server will be load balanced to the default server farm named bosco.

```
Router(config)# mod csm 2
Router(config-module-csm)# map map3 url
Router(config-slb-map-url)# match protocol http url *finance*
Router(config-slb-map-url)#
Router(config-slb-map-url)# map map4 url
Router(config-slb-map-url)# match protocol http url *mail*
Router(config-slb-map-url)#
Router(config-slb-map-url)# serverfarm bar1
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)#
Router(config-slb-real)# serverfarm bar2
Router(config-slb-sfarm)# real 10.1.0.106
Router(config-slb-real)# inservice
Router(config-slb-real)#
Router(config-slb-real)# serverfarm bosco
Router(config-slb-sfarm)# real 10.1.0.107
Router(config-slb-real)# inservice
Router(config-slb-real)#
Router(config-slb-real)# policy pc1
Router(config-slb-policy)# serverfarm bar1
Router(config-slb-policy)# url-map map3
Router(config-slb-policy)# exit
Router(config-module-csm)#
Router(config-module-csm)# policy pc2
Router(config-slb-policy)# serverfarm bar2
```

```
Router(config-slb-policy)# url-map map4
Router(config-slb-policy)# exit
Router(config-module-csm)#
Router(config-module-csm)# vserver bar1
Router(config-slb-vserver)# virtual 10.1.0.86 tcp 80
Router(config-slb-vserver)# slb-policy pc1
Router(config-slb-vserver)# slb-policy pc2
Router(config-slb-vserver)# serverfarm bosco
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)#
```

Configuring TCP Parameters

Transmission Control Protocol (TCP) is a connection-oriented protocol that uses known protocol messages for activating and deactivating TCP sessions. In server load balancing, when adding or removing a connection from the connection database, the Finite State Machine correlates TCP signals such as SYN, SYN/ACK, FIN, and RST. When adding connections, these signals are used for detecting server failure and recovery and for determining the number of connections per server.

The CSM also supports User Datagram Protocol (UDP). Because UDP is not connection-oriented, protocol messages cannot be generically sniffed (without knowing details of the upper-layer protocol) to detect the beginning or end of a UDP message exchange. Detection of UDP connection termination is based on a configurable idle timer. Protocols requiring multiple simultaneous connections to the same real server are supported (such as FTP). Internet Control Management Protocol (ICMP) messages destined for the virtual IP address are also handled (such as ping).

To configure TCP parameters, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# vserver` *virtserver-name*	Identifies the virtual server and enters the virtual server configuration mode[1,2].
Step 2	`Router(config-slb-vserver)# idle` *duration*	Configures the amount of time (in seconds) that connection information is maintained in the absence of packet activity for a connection[2].

1. Enter the **exit** command to leave a mode or submode. To return to the Router (config)> top level of the menu, enter the **end** command.
2. The **no** form of this command restores the defaults.

This example shows how to configure TCP parameters for virtual servers:

```
Router(config-module-csm)# vserver barnett
Router(config-slb-vserver)# idle 10
```

The CSM provides support for fragmented TCP packets. The TCP fragment feature only works with VIPs that have Level 4 policies defined and will not work for SYN packets or for Layer 7 policies. To support fragmented TCP packets, the CSM matches the TCP fragments to existing data flows or by matching the bridging VLAN ID. The CSM will not reassemble fragments for Layer 7 parsing. Because the CSM has a finite number of buffers and fragment ID buckets, packet resending is required when there are hash collisions.

When enabling TCP splicing, you must designate a virtual server as a Layer 7 device even when it does not have a Layer 7 policy. This option is only valid for the TCP protocol.

To configure TCP splicing, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# vserver virtserver-name`	Identifies the virtual server and enters the virtual server configuration mode[1,2].
Step 2	`Router(config-slb-vserver)# vserver tcp-protect`	Designates the virtual server for TCP splicing[2].
Step 3	`Router(config-slb-vserver)# virtual 100.100.100.100 tcp any service tcp-termination`	Enables TCP splicing.

1. Enter the **exit** command to leave a mode or submode. To return to the Router (config)> top level of the menu, enter the **end** command.
2. The **no** form of this command restores the defaults.

Configuring Redirect Virtual Servers

The **redirect-vserver** command is a server farm submode command that allows you to configure virtual servers dedicated to real servers. This mapping provides connection persistence, which maintains connections from clients to real servers across TCP sessions.

Redirection configuration with the CSM this can be done by creating the initial virtual server which loadbalances to the redirect serverfarm as either a L4 or L7 (policy based) virtual server, depending on your preference.

The redirect server farm must have a redirect virtual server configured along with a redirirection string, as follows:

```
serverfarm REDIR-FARM
 nat server
 nat client CLIENTNAT
 redirect-vserver REDVS1
  webhost relocation 10.86.213.216
  inservice
```

The name given to the redirect virtual server only identifies it and plays no role unless you want the virtual server to stop issuing redirects if the real server is down. You will need to configure a virtual address under the redirect virtual server, add a real server, and configure the real server to the redirect virtual server. When this real server goes down the redirect virtual server goes down and it will stop sending redirects. For example:

```
!
serverfarm REDIR-FARM
 nat server
 nat client CLIENTNAT
 redirect-vserver REDVS1
  webhost relocation 10.86.213.216
  virtual 10.86.213.216 tcp www
  inservice
 real 10.86.213.193
  redirect-vserver REDVS1
  inservice
 probe TEST-TCP
!
vserver REDVS
 virtual 10.86.213.212 tcp www
 serverfarm REDIR-FARM
 persistent rebalance
```

```
  inservice

Router(config-slb-real)# do sho mod csm 6 serverfarm name redir-farm det
REDIR-FARM, type = SLB, predictor = RoundRobin
  nat = SERVER, CLIENT(CLIENTNAT)
  virtuals inservice = 1, reals = 1, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Redirect virtual servers:  1
    REDVS1, virtual 10.86.213.216:80, webhost 10.86.213.216 302, OUTOFSERVICE
  Probes:
    TEST-TCP, type = tcp
  Real servers:
    10.86.213.193, weight = 8, PROBE_FAILED, conns = 0
  Total connections = 0
```

The server farm always issuess the redirect unless configured in this manner. The virtual address under the redirect virtual server works as a virtual server and load balances to the real ser er configured in a 1-to-1 mapping. You cannot add more real servers to load balance under this virtual server, because you must create unique redirect virtual server for each real server.

```
serverfarm WEBFARM
  redirect-vserver SERV40_6000
   webhost relocation 172.1.2.40:6000
   virtual 172.1.2.40 tcp 6000
   inservice
  redirect-vserver SERV30_6000
   webhost relocation 172.1.2.30:6000
   virtual 172.1.2.30 tcp 6000
   inservice
  real 10.10.2.40
   redirect-vserver SERV40_6000
   inservice
  real 10.10.2.30
   redirect-vserver SERV30_6000
   inservice
  probe TEST-TCP
!
vserver WEBSITE
  virtual 172.1.2.150 tcp www
  serverfarm WEBFARM
  inservice
```

The **webhost backup** command allows a backup redirect server to be issued if the real server has failed. This command can only be used when you are using the virtual server under the redirect virtual server, under the server farm. This allows for clients that were given a redirect to this virtual server, but the server has gone down before the new request could come in. The backup string would be sent, which redirects the client to a different virtual server. This command backs up the real server associated with the redirect virtual server, not the redirect virtual server.

In the next example when the probe fails on real 10.86.213.188 8881, a redirect for test.url.com will be sent when a connection is made to the virtual 10.86.213.178 9991.

```
!
serverfarm SF1-REDIR
 nat server
 nat client CLIENT-NAT
 redirect-vserver VS1
  webhost relocation 10.86.213.178:9991
  webhost backup test.url.com
  virtual 10.86.213.178 tcp 9991
  inservice
 real 10.86.213.188 8881
```

```
  redirect-vserver VS1
  inservice
 probe TCP
!
vserver V2
 virtual 10.86.213.179 tcp 82
 serverfarm SF1-REDIR
 persistent rebalance
 inservice
!
```

Additional options for the redirect virtual server are available. You can adding **%p** to the end of the relocation string so that it appends the remainder of the URL with the redirection. Enter **CTRL+V ?** to embed a question mark into the URL. The default is to a type 302 redirect, but you can change the redirection to a 301 as follows:

```
serverfarm SF1-REDIR
  nat server
  nat client CLIENT-NAT
  redirect-vserver 22
   webhost relocation www.jw?.com%p 301
   inservice
```

You may also put **https://** or **ftp://** into the string, but this can also be done with the **ssl** *word* command Any number other than 21 or 80 prepends the **https://** and uses the port number given. Ports 21 and 80 prepend **ftp://** and **http://** respectively.

```
Route(config-slb-redirect-vs)# ssl ?
  <1-65535>  ssl port number
  ftp        File Transfer Protocol (21)
  https      Secure Hypertext Transfer Protocol (443)
  www        World Wide Web - Hypertext Transfer Protocol (80)
```

To configure redirect virtual servers, perform this task:

	Command	Purpose
Step 1	`Router(config-slb-sfarm)# redirect-vserver` *name*	Configures virtual servers dedicated to real servers and enters the redirect server submode[1, 2].
Step 2	`Router(config-slb-redirect-v)# webhost relocation` *relocation string*	Configures the destination URL host name when redirecting HTTP requests arrive at this server farm. Only the beginning of the URL can be specified in the relocation string. The remaining portion is taken from the original HTTP request[2].
Step 3	`Router(config-redirect-v)# webhost backup` *backup string*	Configures the relocation string sent in response to HTTP requests in the event that the redirect server is out of service. Only the beginning of the relocation string can be specified. The remaining portion is taken from the original HTTP request[2].
Step 4	`Router(config-redirect-v)# virtual` *v_ipaddress* **tcp** *port*	Configures the redirect virtual server IP address and port[2].
Step 5	`Router(config-redirect-v)# idle` *duration*	Sets the CSM connection idle timer for the redirect virtual server[2].
Step 6	`Router(config-redirect-v)# client` *ip-address network-mask* [**exclude**]	Configures the combination of the IP address and network mask used to restrict which clients are allowed to access the redirect virtual server[2].

	Command	Purpose
Step 7	`Router(config-redirect-v)# inservice`	Enables the redirect virtual server and begins advertisements[2].
Step 8	`Router(config-redirect-v)# ssl port`	(Optional) Enables SSL forwarding by the virtual server.
Step 9	`Router# show module csm vserver redirect [detail]`	Shows all redirect servers configured.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's-top level.
2. The **no** form of this command restores the defaults.

This example shows how to configure redirect virtual servers to specify virtual servers to real servers in a server farm:

```
Router (config)# serverfarm FARM1
Router (config-slb-sfarm)# redirect-vserver REDIR_1
Router (config-slb-redirect-)# webhost relocation 127.1.2.30 301
Router (config-slb-redirect-)# virtual 172.1.2.30 tcp www
Router (config-slb-redirect-)# inservice
Router (config-slb-redirect-)# exit
Router (config-slb-sfarm)# redirect-vserver REDIR_2
Router (config-slb-redirect-)# webhost relocation 127.1.2.31 301
Router (config-slb-redirect-)# virtual 172.1.2.31 tcp www
Router (config-slb-redirect-)# inservice
Router (config-slb-redirect-)# exit
Router (config-slb-sfarm)# real 10.8.0.8
Router (config-slb-real)# redirect-vserver REDIR_1
Router (config-slb-real)# inservice
Router (config-slb-sfarm)# real 10.8.0.9
Router (config-slb-real)# redirect-vserver REDIR_2
Router (config-slb-real)# inservice
Router (config-slb-real)# end
Router# show module csm serverfarm detail
```

Configuring Maps

You configure maps to define multiple URLs, cookies, HTTP headers, and return codes into groups that can be associated with a policy when you configure the policy. (See the "Configuring Policies" section on page 6-11.) Regular expressions for URLs (for example, *url1* and *url2*) are based on UNIX filename specifications. See Table 6-1 for more information.

To add a URL map, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# map url-map-name url`	Creates a group to hold multiple URL match criteria.[1, 2]
Step 2	`Router(config-slb-map-url)# match protocol http url url-path`	Specifies a string expression to match against the requested URL[2].

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

Table 6-1 Special Characters for Matching String Expressions

Convention	Description
*	Zero or more characters.
?	Exactly one character. **Note** You must precede the question mark with a Ctrl-V command to prevent the CLI Parser from interpretingit as a help request
\	Escaped character.
Bracketed range [0-9]	Matching any single character from the range.
A leading ^ in a range	Do not match any in the range. All other characters represent themselves.
.\a	Alert (ASCII 7).
.\b	Backspace (ASCII 8).
.\f	Form-feed (ASCII 12).
.\n	New line (ascii 10).
.\r	Carriage return (ASCII 13).
.\t	Tab (ASCII 9).
.\v	Vertical tab (ASCII 11).
.\0	Null (ASCII 0).
.\\	Backslash.
.\x##	Any ASCII character as specified in two-digit hex notation.

To add a cookie map, perform this task:

	Command	Purpose
Step 1	`Router(config)# map` *cookie-map-name* `cookie`	Configures multiple cookies into a cookie map[1].
Step 2	`Router(config-slb-map-cookie)# match protocol http cookie` *cookie-name* `cookie-value` *cookie-value-expression*	Configures multiple cookies[1].

1. The **no** form of this command restores the defaults.

This example shows how to configure maps and associate them with a policy:

```
Router(config-module-csm)# serverfarm p1_url_url_1
Router(config-slb-sfarm)# real 10.8.0.26
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit
Router(config-slb-policy)# serverfarm p1_url_url_1
Router(config-slb-policy)# url-map url_1
Router(config-slb-policy)# exit
Router(config-module-csm)# serverfarm p1_url_url_2
Router(config-slb-sfarm)# real 10.8.0.27
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
```

```
Router(config-slb-sfarm)# exit
Router(config-module-csm)# map url_1  url
Router(config-slb-map-url)# match protocol http url /url1
Router(config-slb-map-url)# exit
Router(config-module-csm)# map url_2 url
Router(config-slb-map-url)# match protocol http url /url/url/url
Router(config-slb-map-url)# match protocol http url /reg/*long.*
Router(config-slb-map-url)# exit
Router(config-module-csm)# policy policy_url_1
Router(config-module-csm)# policy policy_url_2
Router(config-slb-policy)# serverfarm p1_url_url_2
Router(config-slb-policy)# url-map url_2
Router(config-slb-policy)# exit
Router(config-module-csm)# vserver vs_url_url
Router(config-slb-vserver)# virtual 10.8.0.145 tcp 80
Router(config-slb-vserver)# slb-policy policy_url_1
Router(config-slb-vserver)# slb-policy policy_url_2
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
```

Using the **map** command, you create a map group with the type HTTP header. When you enter the **map** command, you are placed in a submode where you can specify the header fields and values for CSM to search for in the request.

To create a map for the HTTP header, perform this task:

Command	Purpose
`Router(config-module-csm)# map` *name* **header**	Creates and names an HTTP header map group.

For more information about header maps, see the "Configuring Generic Header Parsing" section on page 6-12.

To create a map for return code checking, perform this task:

Command	Purpose
`Router(config-module-csm)# map` *name* **retcode**	Creates and names a return code map group.

To configure HTTP return error code checking, perform this task:

Command	Purpose
`Router(config-slb-sfarm)# retcode-map` *name_of_map*	Configures HTTP return error code checking.

For more information about return code maps, see the "Configuring HTTP Return Code Checking" section on page 9-8.

Configuring Policies

Policies are access rules that traffic must match when balancing to a server farm. Policies allow the CSM to balance Layer 7 traffic. Multiple policies can be assigned to one virtual server, creating multiple access rules for that virtual server. When configuring policies, you first configure the access rules (maps, client-groups, and sticky groups) and then you combine these access rules under a particular policy.

Note You must associate a server farm with a policy. A policy that does not have an associated server farm cannot forward traffic. The server farm associated with a policy receives all the requests that match that policy.

When the CSM is able to match policies, it selects the policy that appears first in the policy list. Policies are located in the policy list in the sequence in which they were bound to the virtual server.

A policy can be matched even if all the servers in the associated server farm are down. The default behavior of the policy in that case is to not accept those connections and send back a reset (RST) to the clients. To change this behavior, add a backup server farm for that policy.

You can reorder the policies in the list by removing policies and reentering them in the correct order. To remove and enter policies, enter the **no slb-policy** *policy name* command and the **slb-policy** *policy name* command in the virtual server submode.

To configure load-balancing policies, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# policy policy-name`	Creates the policy and enters the policy submode to configure the policy attributes[1].
Step 2	`Router(config-slb-policy)# url-map url-map-name`	Associates a URL map to a policy[2]. You must have previously created and configured the URL maps and cookie maps with the **map** command. See the "Configuring Generic Header Parsing" section on page 6-12.
Step 3	`Router(config-slb-policy)# cookie-map cookie-map-name`	Associates a cookie map to a policy[2].
Step 4	`Router(config-slb-policy)# header-map name`	Associates an HTTP header map to a policy.
Step 5	`Router(config-slb-policy)# sticky-group group-id`	Associates this policy to a specific sticky group[2].
Step 6	`Router(config-slb-policy)# client-group value | std-access-list-name`	Configures a client filter associated with a policy. Only standard IP access lists are used to define a client filter.
Step 7	`Router(config-slb-policy)# serverfarm serverfarm-name`	Configures the server farm serving a particular load-balancing policy. Only one server farm can be configured per policy[2].
Step 8	`Router(config-slb-policy)# set ip dscp dscp-value`	Marks traffic with a DSCP value if packets matched with the load-balancing policy[2].

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

This example assumes that the URL map, map1, has already been configured and shows how to configure server load-balancing policies and associate them to virtual servers:

```
Router(config-slb-policy)# serverfarm pl_sticky
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-sfarm)# inservice
Router(config-slb-policy)# exit
Router(config-module-csm)# policy policy_sticky_ck
Router(config-slb-policy)# serverfarm pl_sticky
Router(config-slb-policy)# url-map map1
Router(config-slb-policy)# exit
Router(config-module-csm)# vserver vs_sticky_ck
Router(config-slb-vserver)# virtual 10.1.0.80 tcp 80
Router(config-slb-vserver)# slb-policy policy_sticky_ck
Router(config-slb-sfarm)# inservice
Router(config-slb-policy)# exit
```

Configuring Generic Header Parsing

In software release 2.1(1), the CSM supports generic HTTP request header parsing. The HTTP request header contains fields that describe how content should be formatted to meet the user's requirements.

Understanding Generic Header Parsing

The CSM uses the information it learns by parsing and matching fields in the HTTP header along with policy information to make load-balancing decisions. For example, by parsing the browser-type field in the HTTP header, the CSM can determine if a user is accessing the content with a mobile browser and can select a server that contains content formatted for a mobile browser.

An example of a HTTP Get request header record is as follows:

```
GET /?u HTTP/1.1<0D><0A>
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg<0D><0A>
Referer: http://www.yahoo.com/<0D><0A>
Accept-Language: en-us<0D><0A>
Accept-Encoding: gzip, deflate<0D><0A>
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)<0D><0A>
Host: finance.yahoo.com<0D><0A>
Connection: Keep-Alive<0D><0A>
Cookie: B=51g3cjstaq3vm; Y=1<0D><0A>
<0D><0A>
```

Generic Header Parsing Configuration

You configure generic header parsing by entering commands that instruct the CSM to perform policy matching on fields in the HTTP header. These sections describe how to configure generic header parsing on the CSM:

- Creating a Map for the HTTP Header, page 6-13
- Specifying Header Fields and Match Values, page 6-14
- Assigning an HTTP Header Map to a Policy, page 6-14
- Assigning the Policy to a Virtual Server, page 6-15

- Generic Header Parsing Example, page 6-15

Creating a Map for the HTTP Header

Using the **map** command, you create a map group with the type HTTP header. When you enter the **map** command, you are placed in a submode where you can specify the header fields and values for CSM to search for in the request.

To create a map for the HTTP header, perform this task:

Command	Purpose
`Router(config-module-csm)# map` *name* `header`	Creates and names a HTTP header map group.

Note Other map types include a URL and a cookie.

Specifying Header Fields and Match Values

You can specify the name of the field and the corresponding value for the CSM to match when receiving an HTTP request by using the **match** command.

To specify head fields and match values, perform this task:

Command	Purpose
`Router(config-slb-map-header)# match protocol http header` *field* `header-value` *expression*	Specifies the name of the field and value. The field can be any HTTP header except cookie. You can configure cookie map if you want to configure cookie header.

Note The CSM allows you to specify one or more fields in the HTTP header to be the criteria for policy matching. When multiple fields are configured in a single HTTP header group, all of the expressions in this group must match in order to satisfy this criteria.

Assigning an HTTP Header Map to a Policy

In policy submode, you specify the header map to include in that policy. The header map contains the HTTP header criteria to be included in a policy.

To assign an HTTP header map to a policy, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# policy` policy-*name*	Creates a policy.
Step 2	`Router(config-slb-policy)# header-map` *name*	Assigns an HTTP header map to a policy.

Note By default, a policy rule can be satisfied with any HTTP header information. The HTTP URL and HTTP cookie are specific types of header information and are handled separately by the CSM.

Assigning the Policy to a Virtual Server

In virtual server submode, specify the name of the policy that has the header map assigned, using the **vserver** *virtserver-name* command.

To specify a policy with a header map assigned, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)#` **vserver** `virtserver-`*name*	Configures a virtual server.
Step 2	`Router(config-slb-policy)#` **header-map** *name*	Assigns an HTTP header map to a policy.

Generic Header Parsing Example

This example shows how to configure generic header parsing:

```
Router(config)# mod csm 2
Router(config-module-csm)# !!!configure generic header map
Router(config-module-csm)# map map2 header
Router(config-slb-map-heaer)# $col http header Host header-value *.yahoo.com

Router(config-slb-map-header)# !!! configure serverfarm
Router(config-slb-map-header)# serverfarm farm2
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit

Router(config-module-csm)# !!! configurate policy
Router(config-module-csm)# policy pc2
Router(config-slb-policy)# serverfarm farm2
Router(config-slb-policy)# header-map map2
Router(config-slb-policy)# exit

Router(config-module-csm)# !!! config vserver
Router(config-module-csm)# vserver vs2
Router(config-slb-vserver)# virtual 10.1.0.82 tcp 80
Router(config-slb-vserver)# slb-policy pc2
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# end
Router(config)# show module csm 2 map det
```