
# Configuring Health Monitoring

This chapter describes how to configure the health monitoring on the CSM and contains these sections:



## Configuring Probes for Health Monitoring

Configuring health probes to the real servers allows you to determine if the real servers are operating correctly. The health of a real server is categorized as follows:

- Active—The real server responds appropriately.
- Suspect—The real server is unreachable or returns an invalid response. The probes are retried.
- Failed—The real server fails to reply after a specified number of consecutive retries. You are notified and the CSM adjusts incoming connections accordingly. Probes continue to a failed server until the server becomes active again.

The CSM supports probes used to monitor real servers. Configuring a probe involves the following:

- Entering the probe submode
- Naming the probe
- Specifying the probe type

The CSM supports a variety of probe types that monitor real servers, including FTP, DNS, or HTTP.

**Note** By default, no probes are configured on the CSM.

When configuring the CSM for health probe monitoring, you can use a multiple-tiered approach that includes the following actions:

- Active probes—These probes run periodically. ICMP, TCP, HTTP, and other predefined health probes fall into this category. Scripted health probes are included here as well. Active probes do not impact the session setup or teardown system.
- Passive monitoring (in-band health monitoring)—Monitors sessions for catastrophic errors that can remove a server from services. Catastrophic errors may be reset (RST) from the server or no response from a server. These health checks operate at a full-session rate and recognize failing servers quickly.

- Passive HTTP error code checking (in-band response parsing)—The CSM parses HTTP return codes and watches for codes such as "service unavailable" so that it can take a server out of service. Passive HTTP error code checking has a small impact on session performance.

To set up a probe, you must configure it by naming the probe and specifying the probe type while in probe submode.

After configuring a probe, you must associate it with a server farm for the probe to take effect. All servers in the server farm receive probes of the probe types that are associated with that server farm. You can associate one or more probe types with a server farm.

If you assign a port number when configuring either the real server or the virtual server, you do not need to specify a port number when you configure a probe. The probe inherits the port number from the real or virtual server configuration.

You can override port information for a real server and virtual server by explicitly specifying a port to probe in the health probe configuration using the optional health probe port feature. This feature allows you to set a port for use by the health probes when no port is specified either in the real server or virtual server.

After you configure a probe, associate single or multiple probes with a server farm. All servers in the server farm receive probes of the probe types that are associated with that pool.

**Note** If you associate a probe of a particular type with a server farm containing real servers that are not running the corresponding service, the real servers send error messages when they receive a probe of that type. This action causes the CSM to place the real server in a failed state and disable the real server from the server farm.

To specify a probe type and name, perform this task:

| | Command | Purpose |
|---|---|---|
| **Step 1** | `Router(config-module-csm)# probe probe-name {`**`http`** \| **`icmp`** \| **`telnet`** \| **`tcp`** \| **`ftp`** \| **`smtp`** \| **`dns`** \| **`kal-ap-udp`**`}` | Specifies a probe type and a name[1] [2]:<br>• *probe-name* is the name of the probe being configured; it has a character string of up to 15 characters.<br>• **http** creates an HTTP probe with a default configuration.<br>• **icmp** creates an ICMP probe with a default configuration.<br>• **telnet** creates a Telnet probe with a default configuration.<br>• **tcp** creates a TCP probe with a default configuration.<br>• **ftp** creates an FTP probe with a default configuration.<br>• **smtp** creates an SMTP probe with a default configuration.<br>• **dns** creates a DNS probe with a default configuration.<br>• **kal-ap-udp** creates a GSLB target probe. |
| **Step 2** | `Router(config-slb-probe-tcp)# `**`port`** *`port-number: 1-MAXUSHORT`* | Configures an optional port for a probe[3]. |

| | Command | Purpose |
|---|---|---|
| **Step 3** | `Router# show module csm slot probe` | Displays all probes and their configuration. |
| **Step 4** | `Router# show module csm slot tech-support probe` | Displays probe statistics. |

1. The **no** form of this command removes the probe type from the configuration.
2. Inband health monitoring provides a more scalable solution if you are receiving performance alerts.
3. The **port** command does not exist for the ICMP or PING health probe.

**Note** When you specify a probe name and type, it is initially configured with the default values. Enter the probe configuration commands to change the default configuration.

This example shows how to configure a probe:

```
Router(config-module-csm)# probe probe1 tcp
Router(config-slb-probe-tcp)# interval 120
Router(config-slb-probe-tcp)# retries 3
Router(config-slb-probe-tcp)# failed 300
Router(config-slb-probe-tcp)# open 10
Router(config-slb-probe-tcp)# serverfarm sf4
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)# probe probe1
Router(config-slb-sfarm)# vserver vs4
Router(config-slb-vserver)# virtual 10.1.0.84 tcp 80
Router(config-slb-vserver)# serverfarm sf4
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# end
```

**Note** There are two different timeout values: open and receive. The open timeout specifies how many seconds to wait for the connection to open (that is, how many seconds to wait for SYN ACK after sending SYN). The receive timeout specifies how many seconds to wait for data to be received (that is, how many seconds to wait for an HTTP reply after sending a GET/HHEAD request). Because TCP probes close as soon as they open without sending any data, the receive timeout is not used.

# Probe Configuration Commands

These commands are common to all probe types:

| Command | Purpose |
|---|---|
| `Router(config-slb-probe)#` **`interval`** *seconds* | Sets the interval between probes in seconds (from the end of the previous probe to the beginning of the next probe) when the server is healthy[1] [2].<br>Range = 2–65535 seconds<br>Default = 120 seconds |
| `Router(config-slb-probe)#` **`retries`** *retry-count* | Sets the number of failed probes that are allowed before marking the server as failed[1].<br>Range = 0–65535<br>Default = 3 |
| `Router(config-slb-probe)#` **`failed`** *failed-interval* | Sets the time between health checks when the server has been marked as failed. The time is in seconds[1].<br>Range = 2–65535<br>Default = 300 seconds |
| `Router(config-slb-probe)#` **`open`** *open-timeout* | Sets the maximum time to wait for a TCP connection. This command is not used for any non-TCP health checks (ICMP or DNS[1]).<br>Range = 1–65535<br>Default = 10 seconds |

1. The **no** form of this command restores the defaults.
2. Inband health monitoring provides a more scalable solution if you are receiving performance alerts.

# Configuring an HTTP Probe

An HTTP probe establishes an HTTP connection to a real server and then sends an HTTP request and verifies the response. The **probe** *probe-name* **http** command places the user in HTTP probe configuration submode.

To configure an HTTP probe, perform this task:

| | Command | Purpose |
|---|---|---|
| **Step 1** | `Router(config-module-csm)#` **`probe`** *probe-name* ***`http`*** | Configures an HTTP probe and enters the HTTP probe submode[1]. |
| **Step 2** | `Router(config-slb-probe-http)#` **`credentials`** *username* [*password*] | Configures basic authentication values for the HTTP SLB probe[1]. |

| | Command | Purpose |
|---|---|---|
| **Step 3** | `Router(config-slb-probe-http)# expect status` *min-number* [*max-number*] | Configures a status code to expect from the HTTP probe. You can configure multiple status ranges by entering one **expect status** command at a time[1].<br><br>*min-number*—If you do not specify a *max-number*, this number is taken as a single status code. If you specify a maximum number, this number is taken as the minimum status code of a range.<br><br>*max-number*—The maximum status code in a range. The default range is 0–999. (Any response from the server is considered valid.)<br><br>**Note** If no maximum is specified, this command takes a single number (min-number). If you specify both a minimum number and a maximum number, it takes the range of numbers. |
| **Step 4** | `Router(config-slb-probe-http)# header` *field-name* [*field-value*] | Configures a header field for the HTTP probe. Multiple header fields may be specified[1]. |
| **Step 5** | `Router(config-slb-probe-http)# request` [**method** [**get**\|**head**]] [**url** *path*] | Configures the request method used by an HTTP probe[1]:<br>• **get**—The HTTP **get** request method directs the server to get this page.<br>• **head**—The HTTP **head** request method directs the server to get only the header for this page.<br>• **url**—A character string of up to 1275 characters specifies the URL path; the default path is "/".<br><br>**Note** The CSM supports only the **get** and **head** request methods; it does not support the **post** and other methods. The default method is **get**. |

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

# Configuring an ICMP Probe

An ICMP probe sends an ICMP echo (for example, ping) to the real server. The **probe icmp** command enters the ICMP probe configuration mode. All the common **probe** commands are supported except the **open** command, which is ignored.

To configure an ICMP probe, perform this task:

| | Command | Purpose |
|---|---|---|
| **Step 1** | `Router(config-module-csm)# probe` *probe-name* **icmp** | Configures an ICMP probe and enters the ICMP probe submode[1]. |
| **Step 2** | `Router(config-slb-probe-icmp)# interval` | Configures the intervals to wait between probes of a failed server and between probes. |

| | Command | Purpose |
|---|---|---|
| **Step 3** | `Router(config-slb-probe-icmp)# receive` | Specifies the time to make a TCP connection to receive a reply from the server. |
| **Step 4** | `Router(config-slb-probe-icmp)# retries` | Limits the number of retries before considering the server as failed. |

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

# Configuring a UDP Probe

The UDP probe requires ICMP because otherwise the UDP probe will be unable to detect when a server has gone down or has been disconnected. You must associate UDP to the supervisor engine and then configure ICMP.

Because the UDP probe is a raw UDP probe, the CSM is using a single byte in the payload for probe responses. The CSM does not expect any meaningful response from the UDP application. The CSM uses the ICMP Unreachable message to determine if the UDP application is not reachable. If there is no ICMP Unreachable reply in the receive timeout, the CSM assumes that the probe is operating correctly. If the IP interface of the real server is down or disconnected, the UDP probe by itself would not know that the UDP application is not reachable. You must configure the ICMP probe in addition to the UDP probe for any given server.

The CSM uses the DNS probe as the high-level UDP application. You can use a TCL script to configure this probe. See Chapter 10, “Using TCL Scripts with the CSM.”

To configure an ICMP probe, perform this task:

| | Command | Purpose |
|---|---|---|
| **Step 1** | `Router(config-module-csm)# probe` *probe-name* **udp** | Configures a UDP probe and enters the UDP probe submode[1]. |
| **Step 2** | `Router(config-slb-probe-icmp)# interval` | Configures the intervals to wait between probes of a failed server and between probes. |
| **Step 3** | `Router(config-slb-probe-icmp)# receive` | Specifies the time to make a TCP connection to receive a reply from the server. |
| **Step 4** | `Router(config-slb-probe-icmp)# retries` | Limits the number of retries before considering the server as failed. |

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

# Configuring a TCP Probe

A TCP probe establishes and removes connections. The **probe tcp** command enters the TCP probe configuration mode. All the common **probe** commands are supported.

To configure a TCP probe, perform this task:

| | Command | Purpose |
|---|---|---|
| **Step 1** | `Router(config-module-csm)# probe` *probe-name* **tcp** | Configures a TCP probe and enters the TCP probe submode[1]. |
| **Step 2** | `Router(config-slb-probe-icmp)# interval` | Configures the intervals to wait between probes of a failed server and between probes. |
| **Step 3** | `Router(config-slb-probe-icmp)# retries` | Limits the number of retries before considering the server as failed. |

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

# Configuring FTP, SMTP, and Telnet Probes

An FTP, SMTP, or Telnet probe establishes a connection to the real server and verifies that a greeting from the application was received. The **probe** (**ftp, smtp,** or **telnet**) command enters the corresponding probe configuration mode. All the **probe** common options are supported. Multiple status ranges are supported, one command at a time.

To configure a status code to expect from the FTP, SMTP, or Telnet probe, perform this task:

| | Command | Purpose |
|---|---|---|
| **Step 1** | `Router(config-module-csm)# probe` *probe-name* [**ftp** \| **smtp** \| **telnet**] | Configures an FTP, SMTP, or Telnet probe and enters the FTP, SMTP, or Telnet probe submode[1]. |
| **Step 2** | `Router(config-slb-probe-icmp)# interval` | Configures the intervals to wait between probes of a failed server and between probes. |
| **Step 3** | `Router(config-slb-probe-icmp)# receive` | Specifies the time to make a TCP connection to receive a reply from the server. |
| **Step 4** | `Router(config-slb-probe-icmp)# retries` | Limits the number of retries before considering the server as failed. |

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

# Specifying the DNS Resolve Request

A DNS probe sends a domain name resolve request to the real server and verifies the returned IP address. The **probe dns** command places the user in DNS probe configuration submode. All the probe common options are supported except **open,** which is ignored.

To specify the domain name resolve request, perform this task:

| | Command | Purpose |
|---|---|---|
| **Step 1** | `Router(config-module-csm)# probe probe-name dns` | Configures a DNS probe and enters the tcp probe submode[1]. |
| **Step 2** | `Router(config-slb-probe-dns)# [failed \| interval \| retries \| receive]` | Configures the times to wait between probes to make a DNS connection, to receive a reply from the server, and to limit the number of retries before considering the real server as failed. |

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

# Configuring Inband Health Monitoring

These sections describe inband health monitoring:



## Understanding Inband Health Monitoring

To efficiently balance connections, the CSM must continuously monitor the health of all real servers in its configuration. The inband health monitoring feature is configured for each server farm to monitor the health of the servers. The parameters configured per server farm are then applied to each real server in that server farm. You can configure the number of abnormal end sessions that occur before the system considers the real server unreachable, and you also can specify a time to wait before a real server is reintroduced into the server farm and a connection attempt is made.

This feature works with health probes. If health probes and inband health monitoring are both configured on a particular server, both sets of health checks are required to keep a real server in service within a server farm. If either health-checking feature finds a server out of service, the server will not be selected by the CSM for load balancing.

## Configuring Inband Health Monitoring

To configure inband health monitoring, follow these steps:

**Step 1** Verify that you have configured server farms. (See the "Configuring Server Farms" section on page 5-1.)

**Step 2** Enter the **serverfarm** submode command to enable inband health monitoring for each server farm:

```
Router(config-module-csm)# serverfarm serverfarm-name
Router(config-slb-sfarm)# health retries count failed seconds
```

**Note** Retries are the number of abnormal end sessions that the CSM will tolerate before removing a real server from service. The failed time is the number of seconds that the CSM waits before reattempting a connection to a real server that was removed from service by inband health checking.

This example shows how to enable inband health monitoring for a server farm named geo:

```
Router(config-module-csm)# serverfarm geo
Router(config-slb-sfarm)# health retries 43 failed 160
```

# Configuring HTTP Return Code Checking

These sections describe HTTP return code checking:



## Understanding HTTP Return Code Checking

The return error code checking (return code parsing) feature is used to indicate when a server is not returning web pages correctly. This feature extends the capability of CSM to inspect packets, parse the HTML return codes, and act upon the return codes returned by the server.

After receiving an HTTP request from the CSM, the server responds with an HTTP return code. The CSM can use the HTTP return error codes to determine the availability of the server. The CSM can be configured to take a server out of use in response to receiving specific return codes.

A list of predefined codes (100 through 599) are in RFC 2616. For return code checking, some codes are more usable than others. For example, a return code of 404 is defined as a URL not found, which may be the result of the user entering the URL incorrectly. Error code 404 also might mean that the web server has a hardware problem, such as a defective disk drive preventing the server from finding the data requested. In this case, the web server is still alive, but the server cannot send the requested data because of the defective disk drive. Because of the inability of the server to return the data, you do not want future requests for data sent to this server. To determine the error codes you want to use for return code checking, refer to RFC 2616.

When HTTP return code checking is configured, the CSM monitors HTTP responses from all balanced HTTP connections and logs the occurrence of the return code for each real server. The CSM stores return code counts. When a threshold for a return code is reached, the CSM may send syslog messages or remove the server from service.

You can apply a default action, return code counting, syslog messaging, or you can remove the real server from service. You can apply any of these actions or a set of these actions to a server farm. You also can bind a single virtual group to multiple server farms allowing you to reuse a single return code server farm policy on multiple server farms.

**Note** When you configure HTTP return code checking on a virtual server, the performance of that virtual server is impacted. Once return code parsing is enabled, all HTTP server responses must be parsed for return codes.

# Configuring HTTP Return Code Checking

When you configure return error code checking, you configure the attributes of a server farm and associate it with a return code map.

To configure the return code checking, follow these steps:

**Step 1** Verify that you have configured HTTP virtual servers. (See the "Configuring Redirect Virtual Servers" section on page 6-5.)

**Step 2** Enter the map return code command to enable return code mapping and enter the return code map submode:

```
Router(config-module-csm)# map name retcode
```

**Step 3** Configure the return code parsing:

```
Router(config-slb-map-retcode)# match protocol http retcode min max action [count | log |
remove] threshold [reset seconds]
```

You can set up as many matches as you want in the map.

**Step 4** Assign a return code map to a server farm:

```
Router(config-slb-sfarm)# retcode-map name
```

This example shows how to enable return error code checking:

```
Router(config-module-csm)# map httpcodes retcode
Route(config-slb-map-retcode)# match protocol http retcode 401 401 action log 5 reset 120
Route(config-slb-map-retcode)# match protocol http retcode 402 415 action count
Route(config-slb-map-retcode)# match protocol http retcode 500 500 action remove 3 reset 0
Route(config-slb-map-retcode)# match protocol http retcode 503 503 action remove 3 reset 0
Route(config-slb-map-retcode)# exit
Router(config-module-csm)# serverfarm farm1
Router(config-slb-sfarm)# retcode-map httpcodes
Router(config-slb-sfarm)# exit
Router(config-module-csm)# end
```