

Configuring Health Monitoring

This chapter describes how to configure health monitoring on the ACE to track the state of a server by sending out probes. Also referred to as out-of-band health monitoring, the ACE verifies the server response or checks for any network problems that can prevent a client to reach a server. Based on the server response, the ACE can place the server in or out of service, and can make reliable load-balancing decisions.

You can also use health monitoring to detect failures for a gateway or a host in high-availability (redundancy) configurations. For more information, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

The ACE identifies the health of a server in the following categories:

- **Passed**—The server returns a valid response.
- **Failed**—The server fails to provide a valid response to the ACE and is unable to reach a server for a specified number of retries.

By configuring the ACE for health monitoring, the ACE sends active probes periodically to determine the server state. The ACE supports 1000 unique probe configurations, which includes ICMP, TCP, HTTP, and other predefined health probes. The ACE can execute only up to 200 concurrent script probes at a time. The ACE also allows the opening of 2048 sockets simultaneously.

You can associate the same probe with multiple real servers or server farms. Each time that you use the same probe again, the ACE counts it as another probe instance. You can allocate a maximum of 4000 probe instances.

This chapter contains the following major sections:

- [Configuring Active Health Probes](#)
- [Configuring KAL-AP](#)
- [Displaying Probe Information](#)

- [Clearing Probe Statistics](#)
- [Where to Go Next](#)

Configuring Active Health Probes

By default, no active health probes are configured on the ACE. You can configure health probes on the ACE to actively make connections and explicitly send traffic to servers. The probes determine whether the health status of a server passes or fails by its response.

Configuring active probes is a three-step process:

1. Configure the health probe with a name, type, and attributes.
2. Associate the probe with one of the following:
 - A real server.
 - A real server and then associate the real server with a server farm. You can associate a single probe or multiple probes to real servers within a server farm.
 - A server farm. All servers in the server farm receive probes of the associated probe types.
3. Activate the real server or server farm.

For information on associating a probe with a real server or a server farm, and putting it into service, see [Chapter 2, Configuring Real Servers and Server Farms](#).

You can also configure one or more probes to track a gateway or host. For more information, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

This section contains the following topics:

- [Defining an Active Probe and Accessing Probe Configuration Mode](#)
- [Configuring General Probe Attributes](#)
- [Configuring an ICMP Probe](#)
- [Configuring a TCP Probe](#)
- [Configuring a UDP Probe](#)
- [Configuring an Echo Probe](#)

- [Configuring a Finger Probe](#)
- [Configuring an HTTP Probe](#)
- [Configuring an HTTPS Probe](#)
- [Configuring an FTP Probe](#)
- [Configuring a Telnet Probe](#)
- [Configuring a DNS Probe](#)
- [Configuring an SMTP Probe](#)
- [Configuring an IMAP Probe](#)
- [Configuring a POP3 Probe](#)
- [Configuring a SIP Probe](#)
- [Configuring an RTSP Probe](#)
- [Configuring a RADIUS Probe](#)
- [Configuring an SNMP-Based Server Load Probe](#)
- [Configuring a Scripted Probe](#)
- [Example of a UDP Probe Load-Balancing Configuration](#)

Defining an Active Probe and Accessing Probe Configuration Mode

When you initially configure a health probe, you define its type and name. The CLI then enters the probe configuration mode, which allows you to configure the attributes for the probe type.

To define a probe and access its configuration mode, use the **probe** command in configuration mode. The syntax of this command is as follows:

```
probe probe_type probe_name
```

The arguments are as follows:

- *probe_type*—Probe type that determines what the probe sends to the server. Enter one of the following keywords:
 - **icmp**—Specifies an Internet Control Message Protocol (ICMP) probe type and accesses its configuration mode. For configuration details, see the “[Configuring an ICMP Probe](#)” section.
 - **tcp**—Specifies a TCP probe type and accesses its configuration mode. For configuration details, see the “[Configuring a TCP Probe](#)” section.
 - **udp**—Specifies a UDP probe type and accesses its configuration mode. For configuration details, see the “[Configuring a UDP Probe](#)” section.
 - **echo {tcp | udp}**—Specifies an ECHO TCP or UDP probe type and accesses its configuration mode. For configuration details, see the “[Configuring an Echo Probe](#)” section.
 - **finger**—Specifies a Finger probe type and accesses its configuration mode. For configuration details, see the “[Configuring a Finger Probe](#)” section.
 - **http**—Specifies an HTTP probe type and accesses its configuration mode. For configuration details, see the “[Configuring an HTTP Probe](#)” section.
 - **https**—Specifies an HTTPS probe type for SSL and accesses its configuration mode. For configuration details, see the “[Configuring an HTTPS Probe](#)” section.
 - **ftp**—Specifies an FTP probe type and accesses its configuration mode. For configuration details, see the “[Configuring an FTP Probe](#)” section.
 - **telnet**—Specifies a Telnet probe type and accesses its configuration mode. For configuration details, see the “[Configuring a Telnet Probe](#)” section.
 - **dns**—Specifies a DNS probe type and accesses its configuration mode. For configuration details, see the “[Configuring a DNS Probe](#)” section.
 - **smtp**—Specifies a Simple Mail Transfer Protocol (SMTP) probe type and accesses its configuration mode. For configuration details, see the “[Configuring an SMTP Probe](#)” section.
 - **imap**—Specifies an Internet Message Access Protocol (IMAP) probe type and accesses its configuration mode. For configuration details, see the “[Configuring an IMAP Probe](#)” section.

- **pop**—Specifies a POP probe type and accesses its configuration mode. For configuration details, see the “[Configuring a POP3 Probe](#)” section.
- **sip {tcp | udp}**—Specifies the SIP TCP or UDP probe and accesses its configuration mode. For configuration details, see the “[Configuring a SIP Probe](#)” section.
- **rtsp**—Specifies the RTSP probe and accesses its configuration mode. For configuration details, see the “[Configuring an RTSP Probe](#)” section.
- **radius**—Specifies a RADIUS probe type and accesses its configuration mode. For configuration details, see the “[Configuring a RADIUS Probe](#)” section.
- **snmp**—Specifies an SNMP-based server load probe type and accesses its configuration mode. For configuration details, see the “[Configuring an SNMP-Based Server Load Probe](#)” section.
- **scripted**—Specifies a scripted probe type and accesses its configuration mode. For configuration details, see the “[Configuring a Scripted Probe](#)” section. For information on scripts, see [Appendix A, “Using TCL Scripts with the ACE”](#).
- *probe_name*—Name that you want to assign to the probe. Use the probe name to associate the probe to the real server or server farm. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a TCP probe named PROBE1 and access the TCP probe configuration mode, enter:

```
host1/Admin(config)# probe tcp PROBE1  
host1/Admin(config-probe-tcp)#
```

To delete a TCP probe named PROBE1, enter:

```
host1/Admin(config)# no probe tcp PROBE1
```

Some probe attributes and associated commands apply to all probe types. For information on configuring these attributes, see the “[Configuring General Probe Attributes](#)” section.

Configuring General Probe Attributes

When you access probe configuration mode to configure the attributes for the probe, the ACE provides a set of commands that you can configure for all probe types, except as indicated. The following topics describe how to configure the general attributes for a probe:

- [Configuring a Probe Description](#)
- [Configuring the Destination IP Address](#)
- [Configuring the Port Number](#)
- [Configuring the Time Interval Between Probes](#)
- [Configuring the Retry Count for Failed Probes](#)
- [Configuring the Wait Period and Threshold for Successful Probes](#)
- [Configuring the Wait Interval for the Opening of the Connection](#)
- [Configuring the Timeout Period for a Probe Response](#)

Configuring a Probe Description

You can provide a description for a probe by using the **description** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

```
description text
```

The *text* argument is a description of the probe. Enter a text string with a maximum of 240 alphanumeric characters.

For example, to configure a description THIS PROBE IS FOR TCP SERVERS, enter:

```
host1/Admin(config-probe-type) # description THIS PROBE IS FOR TCP  
SERVERS
```

To remove the description for the probe, use the **no description** command. For example, enter:

```
host1/Admin(config-probe-type) # no description
```

Configuring the Destination IP Address

By default, the probe uses the IP address from the real server or server farm configuration for the destination IP address. You can configure the destination address that the probe uses by using the **ip address** command. This command is available for all probe-type configuration modes except scripted. The syntax of this command is as follows:

```
ip address ip_address [routed]
```

The argument and option are as follows:

- *ip_address*—Destination IP address. Enter a unique IPv4 address in dotted-decimal notation (for example, 192.8.12.15).
- **routed**—(Optional) Specifies that the ACE will route the address according to the ACE internal routing table. Hardware-initiated SSL probes do not support this option.



Note For HTTPS probes, the non-routed mode (without the **routed** keyword) behaves the same as the routed mode.

For example, to configure an IP address of 192.8.12.15, enter:

```
host1/Admin(config-probe-type) # ip address 192.8.12.15
```

To reset the default behavior of the probe using the IP address from the real server or server farm configuration, use the **no ip address** command. For example, enter:

```
host1/Admin(config-probe-type) # no ip address
```

Configuring the Port Number

By default, the probe uses the port number based on its type. [Table 4-1](#) lists the default port numbers for each probe type.

Table 4-1 Default Port Numbers for Probe Types

Probe Type	Default Port Number
DNS	53
Echo	7

Table 4-1 *Default Port Numbers for Probe Types (continued)*

Probe Type	Default Port Number
Finger	79
FTP	21
HTTP	80
HTTPS	443
ICMP	Not applicable
IMAP	143
POP3	110
RADIUS	1812
RTSP	554
SIP (both TCP and UDP)	5060
SNMP	161
SMTP	25
TCP	80
Telnet	23
UDP	53

To configure the port number that the probe uses, use the **port** command. This command is available for all probe-type configuration modes except ICMP. The syntax of this command is as follows:

port *number*

The *number* argument is the number of the port. Enter a number from 1 to 65535.

For example, to configure a port number of 88 for an HTTP probe, enter:

```
host1/Admin(config-probe-http) # port 88
```

To reset the port number to its default value, use the **no port** command. For example, to remove an HTTP probe port number of 88 and reset an HTTP probe port number to its default setting of 80, enter:

```
host1/Admin(config-probe-http) # no port
```

Port Number Inheritance for Probes

If you choose not to specify a port number for a probe, the ACE can dynamically inherit the port number specified:

- From the real server specified in a server farm (see the “[Associating Multiple Health Probes with a Server Farm](#)” section).
- From the VIP specified in a Layer 3 and Layer 4 class map (see the “[Configuring a Layer 3 and Layer 4 Class Map for SLB](#)” section).

This flexibility provides you with an ease of configuration. In this case, all you need is a single probe configuration, which will be sufficient to probe a real server on multiple ports or on all VIP ports. The same probe inherits all of the real server’s ports or all of the VIP ports and creates probe instances for each port.

When you explicitly configure a default port through the **probe** command, the probes will always be sent to the default port. In this case, the probe will not dynamically inherit the port number from the real server specified in a server farm or from the VIP specified in a Layer 3 and Layer 4 class map.



Note

Probe port inheritance is not applicable for the server farm predictor method (see the “[Configuring the Server Farm Predictor Method](#)” section), a probe assigned to a standalone real server (see the “[Configuring Real Server Health Monitoring](#)” section), or a probe configured on the active FT group member in a redundant configuration (see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*).

For a Layer 3 and Layer 4 class map, a VIP port will be inherited only if a **match** command consists of a single port. If you specify a wildcard value for the IP protocol value (the **any** keyword) or a port range for the port, port inheritance does not apply for those match statements.

In the following configuration, only match statements 2,3, and 4 will be taken into consideration for port inheritance.

```
class-map match-any l3class
  2 match virtual-address 11.0.0.10 tcp eq 201
  3 match virtual-address 11.0.0.10 tcp eq 202
  4 match virtual-address 11.0.0.10 tcp eq 203
  5 match virtual-address 11.0.0.10 204
  6 match virtual-address 1.1.1.1 10
  7 match virtual-address 1.1.1.1 tcp range 12 34
  9 match virtual-address 1.1.1.1 tcp eq 0
```


The order of precedence for inheriting the probe's port number is as follows:

1. Probe's configured port
2. Server farm real server's configured port
3. VIP's configured port
4. Probe's default port

For example, if the configured probe does not contain a specified port number, the ACE will look for the configured port associated with the real server specified in a server farm. If a port number is not configured, the ACE looks for the configured port associated with the VIP specified in a Layer 3 and Layer 4 class map. If a port number is also not configured, the ACE then uses the probe's default port to perform health monitoring on the back-end real server.

Based on configuration changes, probe instances will be automatically created or deleted accordingly by the ACE. For example, if you did not specify a port number for the probe or for the real server in a server farm, the ACE creates probe instances using the VIP's port information. If you then assign a port number to the probe, all of the previous probe instances that correspond to the VIP's port are no longer valid. The ACE automatically deletes those probe instances and creates new probe instances based on the port number assigned to the probe, and new probe. In the case of the VIP having a port range instead of a single port, the ACE creates a probe instance to the back-end real server with the default probe port.

Deployment Scenario #1—Inheriting a Real Server Port

In the following example without port inheritance, different port numbers (8001 and 8002) are assigned for the two HTTP probes. This is the only difference in the probe configuration.

```
probe http HTTP_PROBE_1
  port 8001
  request method get url /isalive.html

probe http HTTP_PROBE_2
  port 8002
  request method get url /isalive.html

rserver host RS1
  ip address 192.168.210.1
  inservice
```

```

serverfarm host SF1
  rserver RS1 8001
    probe HTTP_PROBE_1
    inservice
  rserver RS1 8002
    probe HTTP_PROBE_2
    inservice

```

In the following example with port inheritance, the single HTTP probe inherits the ports specified for real server RS1 and creates probe instances for each port.

```

probe http HTTP_PROBE
  request method get url /isalive.html

rserver host RS1
  ip address 192.168.210.1
  inservice

serverfarm host SF1
  probe HTTP_PROBE
    rserver RS1 8001
    inservice
  rserver RS1 8002
  inservice

```

Deployment Scenario #2—Inheriting VIP Ports from a Layer 3 and Layer 4 Class Map

In the following example without port inheritance, different port numbers (8001 and 8002) are assigned for the two HTTP probes. This is the only difference in the probe configuration.

```

class-map match-any HTTP_VIP
  match virtual-address 10.0.0.1 eq 8001
  match virtual-address 10.0.0.1 eq 8002

probe http HTTP_PROBE_1
  port 8001
  request method get url /isalive.html

probe http HTTP_PROBE_2
  port 8002
  request method get url /isalive.html

rserver host RS1
  ip address 192.168.210.1
  inservice

```

```
serverfarm host SF1
  probe HTTP_PROBE_1
  probe HTTP_PROBE_2
  rserver RS1
  inservice
```

In the following example with port inheritance, the single HTTP probe inherits the VIP ports from the HTTP_VIP class map and creates probe instances for each port.

```
class-map match-any HTTP_VIP
  match virtual-address 10.0.0.1 eq 8001
  match virtual-address 10.0.0.1 eq 8002

probe http HTTP_PROBE
  request method get url /isalive.html

rserver host RS1
  ip address 192.168.210.1
  inservice

serverfarm host SF1
  probe HTTP_PROBE
  rserver RS1
  inservice
```

Configuring the Time Interval Between Probes

The time interval between probes is the frequency that the ACE sends probes to the server marked as passed. You can change the time interval between probes by using the **interval** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

interval *seconds*

The *seconds* argument is the time interval in seconds. Enter a number from 2 to 65535. By default, the time interval is 15.

The open timeout value for TCP- or UDP- based probes and the receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to these timeout values and the server takes a long time to respond or it fails to reply within the timeout values, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For UDP probes or UDP-based probes, we recommend a time interval value of 30 seconds. The reason for this recommendation is that the ACE data plane has a management connection limit of 100,000. Management connections are used by all probes as well as Telnet, SSH, SNMP, and other management applications. In addition, the ACE has a default timeout for UDP connections of 15 seconds. This means that the ACE does not remove the UDP connections even though the UDP probe has been closed for two minutes. Using a time interval less than 30 seconds may limit the number of UDP probes that can be configured to run without exceeding the management connection limit, which may result in skipped probes.

For example, to configure a time interval of 50 seconds, enter:

```
host1/Admin(config-probe-type) # interval 50
```

To reset the time interval to the default setting of 15, use the **no interval** command. For example, enter:

```
host1/Admin(config-probe-type) # no interval
```

Configuring the Retry Count for Failed Probes

Before the ACE marks a server as failed, it must detect that probes have failed a consecutive number of times. By default, when three consecutive probes have failed, the ACE marks the server as failed. You can configure this number of failed probes by using the **faildetect** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

```
faildetect retry_count
```

The *retry_count* argument is the consecutive number of failed probes before marking the server as failed. Enter a number from 1 to 65535. By default, the count is 3.

For example, to configure the number of failed probes at 5 before declaring the server as failed, enter:

```
host1/Admin(config-probe-type) # faildetect 5
```

To reset the number of probe failures to the default setting of 3, use the **no faildetect** command. For example, enter:

```
host1/Admin(config-probe-type) # no faildetect
```

Configuring the Wait Period and Threshold for Successful Probes

After the ACE marks a server as failed, it waits a period of time and then sends a probe to the failed server. When the ACE receives a number of consecutive successful probes, it marks the server as passed. By default, the ACE waits 60 seconds before sending out a probe to a failed server and marks a server as passed if it receives 3 consecutive successful responses.

To configure the time interval after which the ACE sends a probe to a failed server and the number of consecutive successful probes required to mark the server as passed, use the **passdetect** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

```
passdetect {interval seconds | count number}
```

The keyword and argument are as follows:

- **interval** *seconds*—Specifies the wait time interval in seconds. Enter a number from 2 to 65535. The default is 60.
- **count** *number*—Specifies the number of successful probe responses from the server. Enter a number from 1 to 65535. The default is 3.



Note

The receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to this timeout value and the server takes a long time to respond or it fails to reply within the timeout value, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For example, to configure wait interval at 60 seconds, enter:

```
host1/Admin(config-probe-type) # passdetect interval 60
```

For example, to configure five success probe responses from the server before declaring it as passed, enter:

```
host1/Admin(config-probe-type) # passdetect count 5
```

To reset the wait interval to its default setting, use the **no passdetect interval** command. For example, enter:

```
host1/Admin(config-probe-type) # no passdetect interval
```

To reset the successful probe responses to its default setting, use the **no passdetect count** command. For example, enter:

```
host1/Admin(config-probe-type) # no passdetect count
```

Configuring the Wait Interval for the Opening of the Connection

When the ACE sends a probe, it waits for the SYN-ACK after sending a SYN to open and then send an ACK to establish the connection with the server. You can configure the time interval for a connection to be established by using the **open** command. This command is available in Echo TCP, Finger, FTP, HTTP, HTTPS, IMAP, POP, scripted, SIP, SMTP, TCP, and Telnet probe configuration mode (all TCP-based probes). The syntax of this command is as follows:

open *timeout*

The *timeout* argument is the time to wait in seconds to open a connection with a server. Enter an integer from 1 to 65535. The default wait interval is 1 second.



Note

The open timeout value for TCP-based probes and the receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to these timeout values and the server takes a long time to respond or it fails to reply within the timeout values, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For example, to configure the wait time interval to 25 seconds, enter:

```
host1/Admin(config-probe-type) # open 25
```

To reset the time interval to its default setting of 1 second, use the **no open** command. For example, enter:

```
host1/Admin(config-probe-type) # no open
```

Configuring the Timeout Period for a Probe Response

By default, when the ACE sends a probe, it expects a response within a time period of 10 seconds. For example, for an HTTP probe, the timeout period is the number of seconds to receive an HTTP reply for a GET or HEAD request. If the server fails to respond to the probe, the ACE marks the server as failed.

You can configure this time period to receive a server response to the probe by using the **receive** command. This command is available for all probe-type configuration modes. The syntax of this command is as follows:

receive *timeout*

The *timeout* argument is the timeout period in seconds. Enter a number from 1 to 65535. By default, the timeout period is 10.

**Note**

The open timeout value for TCP-based probes and the receive timeout value can impact the execution time for a probe. When the probe interval is less than or equal to these timeout values and the server takes a long time to respond or it fails to reply within the timeout values, the probe is skipped. When the probe is skipped, the No. Probes skipped counter through the **show probe detail** command increments.

For example, to configure the timeout period for a response at 5 seconds, enter:

```
host1/Admin(config-probe-type) # receive 5
```

To reset the time period to receive a response from the server to its default setting of 10 seconds, use the **no receive** command.

For example, enter:

```
host1/Admin(config-probe-type) # no receive
```

Configuring an ICMP Probe

An ICMP probe sends an ICMP echo request and listens for a response. If a server returns a response, the ACE marks the server as passed. If the server does not send a response causing the probe to time out, or if the server sends an unexpected ICMP echo response type, the ACE marks the probe as failed.

You can create an ICMP probe and access its configuration mode by using the **probe icmp** *name* command in configuration mode.

For example, to define an ICMP probe named PROBE3 and access its mode, enter:

```
host1/Admin(config)# probe icmp PROBE3  
host1/Admin(config-probe-icmp) #
```

After you create an ICMP probe, you can configure the attributes in the [“Configuring General Probe Attributes”](#) section.

Configuring a TCP Probe

A TCP probe initiates a TCP 3-way handshake and expects the server to send a response. By default, a successful response causes the probe to mark the server as passed. The probe then sends a FIN to end the session. If the response is not valid or if there is no response, the probe marks the server as failed.

Optionally, you can configure the probe to send an RST or specific data, and to expect a specific response in order to mark the server as passed. You can also configure the probe to send specific data and receive a specific response from the server. If the response is not valid or there is no response, the probe marks the server as failed.

You can create a TCP probe and access its configuration mode by using the **probe tcp *probe name*** command in configuration mode.

For example, to define a TCP probe named PROBE1 and access its mode, enter:

```
host1/Admin(config)# probe tcp PROBE1  
host1/Admin(config-probe-tcp)#
```

You can configure attributes for a TCP probe, as described in the following topics:

- [Configuring the Termination of the TCP Connection](#)
- [Configuring an Expected Response String from the Server](#)
- [Configuring Data that the Probe Sends to the Server Upon Connection](#)

You can also configure the attributes described in the [“Configuring General Probe Attributes”](#) section.

Configuring the Termination of the TCP Connection

A TCP probe makes a connection, and if the connection through a 3-way handshake (SYN, SYN-ACK, and ACK) is successful, when the ACE receives FIN-ACK from the server, the server is marked as passed. By default, the ACE terminates a TCP connection gracefully by sending a FIN to the server.

**Note**

If a probe is configured for the default graceful connection termination (FIN) and the target server does not send the expected data, the probe terminates the TCP connection to the server with a reset (RST). The probe will continue to send a RST to terminate the server connection as long as the returned data is not the expected data. When the server responds with the correct data again, the probe reverts to terminating the connection with a FIN.

To configure the ACE to terminate a TCP connection by sending an RST, use the **connection term** command. This command is available for TCP-based connection-oriented probes (ECHO TCP, Finger, FTP, HTTP, HTTPS, IMAP, POP, RTSP, SIP TCP, SMTP, TCP, and Telnet probe configuration modes). The syntax of this command is as follows:

connection term forced

For example, enter:

```
host1/Admin(config-probe-tcp)# connection term forced
```

To reset the method to terminate a connection gracefully, use the **no connection term** command. For example, enter:

```
host1/Admin(config-probe-tcp)# no connection term forced
```

Configuring an Expected Response String from the Server

When you configure a probe to expect a regular expression (regex) response string from a server, it searches the response for it. If the ACE finds it, the server is marked as passed. If you do not configure an expected string, the ACE ignores the server response.

**Note**

For HTTP or HTTPS probes, the server response must include the Content-Length header for the **expect regex** command to function. Otherwise, the probe does not attempt to parse the regex.

You can configure what the ACE expects as a response string from the probe destination server by using the **expect regex** command. This command is available in Finger, HTTP, HTTPS, SIP, TCP, and UDP probe configuration modes.

The syntax of this command is as follows:

```
expect regex string [offset number]
```

The argument and option are as follows:

- *string*—Regular expression expected response string from the probe destination. Enter an unquoted text string with no spaces. If the string includes spaces, enclose the string in quotes. The string can be a maximum of 255 alphanumeric characters.
- **offset number**—(Optional) Sets the number of characters into the received message or buffer where the ACE starts searching for the defined expression. Enter a number from 1 to 4000.

For example, to configure the ACE to expect a response string of ack, enter:

```
host1/Admin(config-probe-tcp) # expect regex ack
```

To remove the expectation of a response string, use the **no expect regex** command. For example, enter:

```
host1/Admin(config-probe-http) # no expect regex
```

Configuring Data that the Probe Sends to the Server Upon Connection

You can configure the ASCII data that the probe sends when the ACE connects to the server by using the **send-data** command. This command is available in Echo, Finger, TCP, and UDP probe configuration modes. The syntax of this command is as follows:

```
send-data expression
```

The *expression* argument is the data that the probe sends. Enter an unquoted text string with a maximum of 255 alphanumeric characters including spaces.



Note

If you do not configure the **send-data** command for a UDP probe, the probe sends one byte, 0x00.

For example, to configure the probe to send TEST as the data, enter:

```
host1/Admin(config-probe-tcp) # send-data test
```

To remove the data, use the **no send-data** command. For example, enter:

```
host1/Admin(config-probe-tcp)# no send-data
```

Configuring a UDP Probe



Note

When configuring a UDP probe, you must configure a management-based policy. For more information about policies, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

By default, the UDP probe sends a UDP packet to a server and marks the server as failed if the server returns an ICMP Host or Port Unreachable message. If the ACE does not receive any ICMP errors for the UDP request that was sent, the server is marked as passed. Optionally, you can configure this probe to send specific data and expect a specific response to mark the server as passed.

If the real server is not directly connected to the ACE (for example, it is connected via a gateway) and the IP interface of the server is down or disconnected, the UDP probe by itself would not know that the UDP application is not reachable. If the real server is directly connected to the ACE and the IP interface of the server is down, then the UDP probe fails.

You can create a UDP probe and access its configuration mode by using the **probe udp name** command.

For example, to define a UDP probe named PROBE2 and access its mode, enter:

```
host1/Admin(config)# probe udp PROBE2
host1/Admin(config-probe-udp)#
```

You can configure the following attributes for a UDP probe:

- To configure what the ACE expects as a response from the probe destination server, use the **expect regex** command. For more details about this command, see the “[Configuring an Expected Response String from the Server](#)” section.
- To configure the data sent on the connection for a UDP probe, use the **send-data expression** command. For more details about this command, see the “[Configuring Data that the Probe Sends to the Server Upon Connection](#)” section.

You can also configure the attributes described in the “[Configuring General Probe Attributes](#)” section.

Configuring an Echo Probe

The Echo probe sends a specified string to the server and compares the response with the original string. You must configure the string that needs to be echoed. If the response string matches the original string, the server is marked as passed. If you do not configure a string, the probe behaves like a TCP or UDP probe (see the “[Configuring a TCP Probe](#)” section or the “[Configuring a UDP Probe](#)” section).

You can create an Echo probe and access its configuration mode by using the **probe echo** command. The syntax of this command is as follows:

```
probe echo {tcp | udp} name
```

The keywords and argument are as follows:

- *name*—Identifier of the probe. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- **tcp**—Configures the probe for a TCP connection.
- **udp**—Configures the probe for a UDP connection.

For example, to define a TCP Echo probe named PROBE and access its mode, enter:

```
host1/Admin(config)# probe echo tcp PROBE  
host1/Admin(config-probe-echo-tcp)#
```

For example, to define a UDP Echo probe named PROBE17 and access its mode, enter:

```
host1/Admin(config)# probe echo udp PROBE17  
host1/Admin(config-probe-echo-udp)#
```

For Echo TCP and Echo UDP probes, you can configure the attributes described in the “[Configuring General Probe Attributes](#)” section.

For an Echo TCP probe (configured using the **tcp** keyword), you can also configure the attributes described in the “[Configuring a TCP Probe](#)” section.

For an Echo UDP probe (configured using the **udp** keyword), you can also configure the attributes described in the “[Configuring a UDP Probe](#)” section.

Configuring a Finger Probe

The Finger probe uses a Finger query to a server for an expected response string. The ACE searches the response for the configured string. If the ACE finds the expected response string, the server is marked as passed. If you do not configure an expected response string, the ACE ignores the server response.

You can create a Finger probe and access its configuration mode by using the **probe finger** command. The syntax of this command is as follows:

```
probe finger name
```

The *name* argument is the identifier of the probe. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a Finger probe named PROBE8 and access its mode, enter:

```
host1/Admin(config)# probe finger PROBE8  
host1/Admin(config-probe-finger)#
```

To configure the attributes for a Finger probe, see the “[Configuring General Probe Attributes](#)” and “[Configuring a TCP Probe](#)” sections.

Configuring an HTTP Probe

An HTTP probe establishes a TCP connection and issues an HTTP request to the server for an expected string and status code. The ACE can compare the received response with configured codes, looking for a configured string in the received HTTP page, or verifying hash for the HTTP page. If any of these checks fail, the server is marked as failed.

For example, if you configure an expected string and status code and the ACE finds them both in the server response, the server is marked as passed. However, if the ACE does not receive either the server response string or the expected status code, it marks the server as failed.

**Note**

If you do not configure an expected status code, any response from the server is marked as failed.

The server response must include the Content-Length header for the **expect regex** or **hash** command to function. Otherwise, the probe does not attempt to parse the regex or hash value.

To create an HTTP probe and access its configuration mode, use the **probe http name** command. For example, to define an HTTP probe named PROBE4 and access its mode, enter:

```
host1/Admin(config)# probe http PROBE4  
host1/Admin(config-probe-http)#
```

To configure attributes for an HTTP probe, see the following topics:

- [Configuring the Credentials for a Probe](#)
- [Configuring the Header Field for the HTTP Probe](#)
- [Configuring the HTTP Method for the Probe](#)
- [Configuring the Status Code from the Destination Server](#)
- [Configuring an MD5 Hash Value](#)

After you create an HTTP probe, you can configure the general probe attributes described in the “[Configuring General Probe Attributes](#)” section. You can also configure the TCP probe attributes, including an expected response string, described in the “[Configuring a TCP Probe](#)” section.

Configuring the Credentials for a Probe

The credentials for a probe are the username and password used for authentication on the server. You can configure the credentials for the probe by using the **credentials** command. The syntax of this command is as follows:

```
credentials username [password]
```

The arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—(Optional) The password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-http) # credentials ENG1 TEST
```

To delete the credentials for the probe, use the **no credentials** command. For example, enter:

```
host1/Admin(config-probe-http) # no credentials
```

Configuring the Header Field for the HTTP Probe

You can configure an HTTP header or multiple header fields for the HTTP probe by using the **header** command. The syntax of this command is as follows:

```
header field_name header-value value
```

The keywords and arguments are as follows:

- *field_name*—Identifier for a standard header field. Enter a text string with no spaces and a maximum of 64 alphanumeric characters. If the header field includes spaces, enclose its string with quotes. You can also enter one of the following header keywords:
 - **Accept**—Accept request header
 - **Accept-Charset**—Accept-Charset request header
 - **Accept-Encoding**—Accept-Encoding request header
 - **Accept-Language**—Accept-Language request header

- **Authorization**—Authorization request header
- **Cache-Control**—Cache-Control general header
- **Connection**—Connection general header
- **Content-MD5**—Content-MD5 entity header
- **Expect**—Expect request header
- **From**—From request header
- **Host**—Host request header
- **If-Match**—If-Match request header
- **Pragma**—Pragma general header
- **Referer**—Referer request header
- **Transfer-Encoding**—Transfer-Encoding general header
- **User-Agent**—User-Agent request header
- **Via**—Via general header
- **header-value** *field_value*—Specifies the value assigned to the header field. Enter a text string with a maximum of 255 alphanumeric characters. If the value string includes spaces, enclose the string with quotes.

For example, to configure the Accept-Encoding HTTP header with a field value of IDENTITY, enter:

```
host1/Admin(config-probe-http) # header Accept-Encoding header-value
IDENTITY
```

To remove the header configuration for the probe, use the **no** form of the **header** command. For example, to remove a header with the Accept-Encoding field name, enter:

```
host1/Admin(config-probe-http) # no header Accept-Encoding
```

Configuring the HTTP Method for the Probe

By default, the HTTP request method is a GET with the URL “/”. If you do not configure a URL, the probe functions as a TCP probe.

You can configure the HTTP method and URL used by the probe by using the **request method** command. The syntax of this command is as follows:

```
request method { get | head } url path
```

The keywords and arguments are as follows:

- **get** | **head**—Configures the HTTP request method. The keywords are as follows:
 - **get** —The HTTP GET request method directs the server to get the page. This method is the default.
 - **head** —The HTTP HEAD request method directs the server to get only the header for the page.
- **url** *path*—Specifies the URL path. The *path* argument is a character string of up to 255 alphanumeric characters. The default path is “/”.

For example, to configure the HEAD HTTP method and the `/digital/media/graphics.html` URL used by an HTTP probe, enter:

```
host1/Admin(config-probe-http) # request method head url  
/digital/media/graphics.html
```

To reset the HTTP method for the probe to GET, use the **no request method** command. For example, enter:

```
host1/Admin(config-probe-http) # no request method head url  
/digital/media/graphics.html
```

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

```
expect status min_number max_number
```

The arguments and options are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure an expected status code of 200 that indicates that the HTTP request was successful, enter:

```
host1/Admin(config-probe-http) # expect status 200 200
```

To configure a range of expected status codes from 200 to 210, enter:

```
host1/Admin(config-probe-http) # expect status 200 210
```

To configure multiple ranges of expected status codes from 200 to 202 and 204 to 205, you must configure each range separately. For example, enter:

```
host1/Admin(config-probe-http) # expect status 200 202  
host1/Admin(config-probe-http) # expect status 204 205
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 200, enter:

```
host1/Admin(config-probe-http) # no expect status 200 200
```

To remove a range of expected status codes, enter the range when using the **no expect status** command. For example, to remove a range of 200 to 202 from a range of 200 to 210, enter:

```
host1/Admin(config-probe-http) # no expect status 200 202
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (200 to 202 and 204 to 205), enter:

```
host1/Admin(config-probe-http) # no expect status 200 202  
host1/Admin(config-probe-http) # no expect status 204 205
```

Configuring an MD5 Hash Value

By default, no MD5 hash value is configured on the ACE. To configure the ACE to dynamically generate the hash value or manually configure the value, use the **hash** command. If you do not use this command to configure the hash value, the ACE does not calculate a hash value on the HTTP data returned by the probe. The syntax of this command is as follows:

```
hash [value]
```

When you enter this command with no argument, the ACE generates the hash on the HTTP data returned by the first successful probe. If subsequent HTTP server hash responses match the generated hash value, the ACE marks the server as passed.

If a mismatch occurs due to changes to the HTTP data, the probe fails and the **show probe ... detail** command displays an MD5 mismatch error in the Last disconnect error field. To clear the reference hash and have the ACE recalculate the hash value at the next successful probe, change the URL or method by using the **request method** command. For more information, see the [“Configuring the HTTP Method for the Probe”](#) section.

The optional *value* argument is the MD5 hash value that you want to manually configure. Enter the MD5 hash value as a hexadecimal string with exactly 32 characters (16 bytes).



Note

The server response must include the Content-Length header for the **hash** command to function. Otherwise, the probe does not attempt to parse the hash value.

You can configure the **hash** command on a probe using the HEAD method, however there is no data to hash and has no effect causing the probe to always succeed.

For example, to configure the ACE to generate the hash on the HTTP data returned by the first successful probe, enter:

```
host1/Admin(config-probe-http) # hash
```

To manually configure a hash value, enter:

```
host1/Admin(config-probe-http) # hash 0123456789abcdef0123456789abcdef
```

To configure the ACE to no longer compare the referenced hash value to the computed hash value, enter:

```
host1/Admin(config-probe-http) # no hash
```

To configure the ACE to no longer use a manually configured hash value, enter:

```
host1/Admin(config-probe-http) # no hash
0123456789abcdef0123456789abcdef
```

Configuring an HTTPS Probe

An HTTPS probe is similar to an HTTP probe except that it uses SSL to generate encrypted data. HTTPS probes are hardware assisted, which causes the ACE to send them from the data plane instead of the control plane. This feature causes the ACE to use the routing table (which may bypass the real server IP address) to direct HTTPS probes to their destination regardless of whether you specify the **routed** option or not in the **ip address** command. For more information about the **ip address** command, see the “[Configuring the Destination IP Address](#)” section. Also, ACLs may impact HTTPS probes if you apply them incorrectly. For more information about ACLs, see the *Cisco 4700 Series Application Control Engine Appliance Security Configuration Guide*.



Note

The server response must include the Content-Length header for the **expect regex** or **hash** command to function. Otherwise, the probe does not attempt to parse the regex or hash value.

You can create an HTTPS probe and access its configuration mode by using the **probe https** command. The syntax of this command is as follows:

```
probe https name
```

For the *name* argument, enter an identifier for the HTTPS probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define an HTTPS probe named PROBE5 and access its mode, enter:

```
host1/Admin(config) # probe https PROBE5
host1/Admin(config-probe-https) #
```

To configure attributes for an HTTPS probe, see the following topics:

- [Configuring the Cipher Suite for the HTTPS Probe](#)
- [Configuring the Supported SSL or TLS Version](#)

After you create an HTTPS probe, you can configure the general probe attributes described in the “[Configuring General Probe Attributes](#)” section. You can also configure the HTTP probe attributes described in the “[Configuring an HTTP Probe](#)” section.

Configuring the Cipher Suite for the HTTPS Probe

By default, the HTTPS probe accepts any of the RSA configured cipher suites. You can configure the probe to expect a specific type of RSA cipher suite from the back-end server by using the **ssl cipher** command. The syntax of this command is as follows:

```
ssl cipher RSA_ANY | cipher_suite
```

The keyword and argument are as follows:

- **RSA_ANY**—Specifies that any of the RSA cipher suites from those allowed on the ACE is accepted from the server. This is the default setting.
- *cipher_suite*—RSA cipher suite that the probe expects from the back-end server. Enter one of the following keywords:
 - **RSA_EXPORT1024_WITH_DES_CBC_SHA**
 - **RSA_EXPORT1024_WITH_RC4_56_MD5**
 - **RSA_EXPORT1024_WITH_RC4_56_SHA**
 - **RSA_EXPORT_WITH_DES40_CBC_SHA**
 - **RSA_EXPORT_WITH_RC4_40_MD5**
 - **RSA_WITH_3DES_EDE_CBC_SHA**
 - **RSA_WITH_AES_128_CBC_SHA**
 - **RSA_WITH_AES_256_CBC_SHA**
 - **RSA_WITH_DES_CBC_SHA**
 - **RSA_WITH_RC4_128_MD5**
 - **RSA_WITH_RC4_128_SHA**

For example, to configure the HTTPS probes with the RSA_WITH_RC4_128_SHA cipher suite, enter:

```
host1/Admin(config-probe-https)# ssl cipher RSA_WITH_RC4_128_SHA
```

To reset the default behavior of the HTTPs probes accepting any RSA cipher suite, enter:

```
host1/Admin(config-probe-https)# no ssl cipher
```

Configuring the Supported SSL or TLS Version

The version in the ClientHello message sent to the server indicates the highest supported version. By default, the probe supports **all** as the SSL version. You can configure the version of SSL that the probe supports by using the **ssl version** command in probe HTTPS configuration mode.



Note

For hardware-assisted SSL (HTTPS) probes, the ACE uses the **all** option for the default SSL version and uses the routing table (which may bypass the real server IP address) to direct HTTPS probes to their destination regardless of whether you specify the **routed** option in the **ip address** command.

Additionally, hardware-assisted probes are subject to the same key-pair size limitations as SSL termination. The maximum size of a public key in a server SSL certificate that the ACE can process is 2048 bits.

The syntax of this command is as follows:

```
ssl version {all | SSLv3 | TLSv1}
```

The keywords are as follows:

- **all**—(Default) Specifies all SSL versions.
- **SSLv3**—Specifies SSL version 3.
- **TLSv1**—Specifies TLS version 1.

For example, to configure all SSL versions, enter:

```
host1/Admin(config-probe-https)# ssl version all
```

To reset the default setting, enter:

```
host1/Admin(config-probe-https)# no ssl version
```

Configuring an FTP Probe

An FTP probe establishes a TCP connection to the server. After the ACE receives the service ready message from the server, the ACE performs one of the following actions:

- Issues a **quit** command if the probe is configured for a graceful close.
- Resets the connection for forceful terminations

You can create an FTP probe and access its configuration mode by using the **probe ftp** command. The syntax of this command is as follows:

```
probe ftp name
```

For the *name* argument, enter the identifier of the FTP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define an FTP probe named PROBE8 and access its mode, enter:

```
host1/Admin(config)# probe ftp PROBE8  
host1/Admin(config-probe-ftp)#
```

The “[Configuring the Status Code from the Destination Server](#)” section describes how to configure status codes for the probe.

You can also configure the attributes described in the “[Configuring General Probe Attributes](#)” and “[Configuring a TCP Probe](#)” sections.

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

```
expect status min_number max_number
```

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure an expected status code of 200 that indicates that the request was successful, enter:

```
host1/Admin(config-probe-ftp)# expect status 200 200
```

To configure a range of expected status codes from 200 to 201, enter:

```
host1/Admin(config-probe-ftp)# expect status 200 201
```

To configure multiple ranges of expected status codes from 200 to 201 and 230 to 250, you must configure each range separately. For example, enter:

```
host1/Admin(config-probe-ftp)# expect status 200 201  
host1/Admin(config-probe-ftp)# expect status 230 250
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 200, enter:

```
host1/Admin(config-probe-ftp)# no expect status 200 200
```

To remove a range of expected status codes, enter the range when using the **no expect status** command. For example, to remove a range of 200 to 201, enter:

```
host1/Admin(config-probe-ftp)# no expect status 200 201
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (200 to 201 and 230 to 250), enter:

```
host1/Admin(config-probe-ftp)# no expect status 200 201  
host1/Admin(config-probe-ftp)# no expect status 230 250
```

Configuring a Telnet Probe

A Telnet probe establishes a connection to the server and verifies that a greeting from the application was received. You can create a Telnet probe and access its configuration mode by using the **probe telnet** command. The syntax of this command is as follows:

probe telnet *name*

For the *name* argument, enter an identifier for the Telnet probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a Telnet probe named PROBE6 and access its mode, enter:

```
host1/Admin(config)# probe telnet PROBE6  
host1/Admin(config-probe-telnet)#
```

You can also configure the attributes described in the [“Configuring General Probe Attributes”](#) and [“Configuring a TCP Probe”](#) sections.

Configuring a DNS Probe

A DNS probe sends a request to a DNS server giving it a configured domain (by default, the domain is www.cisco.com). To determine if the server is up, the ACE must receive one of the configured IP addresses for that domain. You can create a DNS probe and access its configuration mode by using the **probe dns** command. The syntax of this command is as follows:

probe dns *name*

For the *name* argument, enter an identifier for the DNS probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a DNS probe named PROBE7 and access its mode, enter:

```
host1/Admin(config)# probe dns PROBE7  
host1/Admin(config-probe-dns)#
```

To configure attributes for a DNS probe, see the following topics:

- [Configuring the Domain Name](#)
- [Configuring the Expected IP Address](#)

You can also configure the attributes described in the [“Configuring General Probe Attributes”](#) section.

Configuring the Domain Name

The DNS probe sends a domain name for the DNS server to resolve. By default, the probe uses the `www.cisco.com` domain. You can configure the domain name that the probe sends to the server by using the **domain** command. The syntax of this command is as follows:

```
domain name
```

The *name* argument is the domain that the probe sends to the DNS server. Enter an unquoted text string with a maximum of 255 alphanumeric characters.

For example, to configure the domain name of `support.cisco.com`, enter:

```
host1/Admin(config-probe-dns) # domain support.cisco.com
```

To reset the domain to `www.cisco.com`, use the **no domain** command. For example, enter:

```
host1/Admin(config-probe-dns) # no domain
```

Configuring the Expected IP Address

When a DNS probe sends a domain name resolve request to the server, it verifies the returned IP address by matching the received IP address with the configured addresses. You can configure the IP address that the ACE expects as a server response to a DNS request by using the **expect address** command. The syntax of this command is as follows:

```
expect address ip_address
```

The *ip_address* argument is the expected returned IP address. Enter a unique IPv4 address in dotted-decimal notation (for example, `192.8.12.15`).

You can specify multiple IP addresses with this command by entering the command with a different address separately. For example, to configure an expected IP address of `192.8.12.15` and `192.8.12.23`, enter:

```
host1/Admin(config-probe-dns) # expect address 192.8.12.15  
host1/Admin(config-probe-dns) # expect address 192.8.12.23
```

To remove an IP address, use the **no expect address** command. For example, enter:

```
host1/Admin(config-probe-dns) # no expect address 192.8.12.15
```

Configuring an SMTP Probe

SMTP probes initiates an SMTP session by logging into the server, sends a HELLO message, and then disconnects from the server. You can create an SMTP probe and access its configuration mode by using the **probe smtp** command. The syntax of this command is as follows:

```
probe smtp name
```

For the *name* argument, enter the identifier of the SMTP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a SMTP probe named PROBE10 and access its mode, enter:

```
host1/Admin(config)# probe smtp PROBE10  
host1/Admin(config-probe-smtp)#
```

After you create an SMTP probe, you can configure the status codes as described in the [“Configuring the Status Code from the Destination Server”](#) section.

You can also configure the attributes described in the [“Configuring General Probe Attributes”](#) section, and configure connection termination as described in the [“Configuring the Termination of the TCP Connection”](#) section.

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

```
expect status min_number max_number
```

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.

- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure a single expected status code of 211, enter:

```
host1/Admin(config-probe-smtp) # expect status 211 211
```

To configure a range of expected status codes from 211 to 250, enter:

```
host1/Admin(config-probe-smtp) # expect status 211 250
```

To configure multiple ranges of expected status codes from 211 and 250 and 252 to 254, you must configure each range separately as follows:

```
host1/Admin(config-probe-smtp) # expect status 211 250
host1/Admin(config-probe-smtp) # expect status 252 254
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 211, enter:

```
host1/Admin(config-probe-smtp) # no expect status 211 211
```

To remove a range of expected status codes, enter the range when using the **no expect status** command. For example, to remove a range of 211 to 250, enter:

```
host1/Admin(config-probe-smtp) # no expect status 211 250
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (211 and 250 and 252 to 254), enter:

```
host1/Admin(config-probe-smtp) # no expect status 211 250
host1/Admin(config-probe-smtp) # no expect status 252 254
```

Configuring an IMAP Probe

An Internet Message Access Protocol (IMAP) probe makes a server connection and sends user credential (login, password, and mailbox) information. The ACE can send a configured command. Based on the server response, the ACE marks the probe as passed or failed.

You can create an IMAP probe and access its configuration mode by using the **probe imap** command. The syntax of this command is as follows:

```
probe imap name
```

For the *name* argument, enter the identifier of the IMAP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define an IMAP probe named PROBE11 and access its mode, enter:

```
host1/Admin(config)# probe imap PROBE11  
host1/Admin(config-probe-imap) #
```

You can configure attributes for an IMAP probe, as described in the following topics:

- [Configuring the Username Credentials](#)
- [Configuring the Mailbox](#)
- [Configuring the Request Command for the Probe](#)

You can also configure the general attributes described in the “[Configuring General Probe Attributes](#)” section and configure connection termination as described in the “[Configuring the Termination of the TCP Connection](#)” section.

Configuring the Username Credentials

The credentials for an IMAP probe are the username and password used for authentication on the server. You can configure the credentials for the probe by using the **credentials** *username* command. The syntax of this command is as follows:

```
credentials username password
```

The arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—Password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-imap) # credentials ENG1 TEST
```

To delete the username credentials for the probe, use the **no credentials** *username* command. For example, to delete the username ENG1, enter:

```
host1/Admin(config-probe-imap) # no credentials ENG1
```

Configuring the Mailbox

You can configure the name of the mailbox from which the probe retrieves e-mail by using the **credentials mailbox** command. The syntax of this command is as follows:

```
credentials mailbox name
```



Note

You must configure the credentials for an IMAP probe using the **credentials** command before you configure the mailbox or the ACE will ignore the specified user mailbox name. See the [“Configuring the Username Credentials”](#) section.

The **mailbox** *name* keyword and argument specify the user mailbox name from which to retrieve e-mail for an IMAP probe. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the user mailbox LETTERS, enter:

```
host1/Admin(config-probe-imap) # credentials mailbox LETTERS
```

To delete the mailbox for the probe, use the **no credentials mailbox** command. For example, enter:

```
host1/Admin(config-probe-imap) # no credentials mailbox
```

Configuring the Request Command for the Probe

You can configure the request command used by an IMAP probe by using the **request command** command. The syntax of this command is as follows:

```
request command command
```



Note

You must configure the name of the mailbox using the **credentials mailbox** command before you configure the request command used by an IMAP probe or the ACE will ignore the specified request command. See the [“Configuring the Mailbox”](#) section.

The *command* argument is the request command for the probe. Enter a text string with a maximum of 32 alphanumeric characters with no spaces.

For example, to configure the last request command for an IMAP probe, enter:

```
host1/Admin(config-probe-imap) # request command last
```

To remove the request command for the probe, use the **no request** command. For example, enter:

```
host1/Admin(config-probe-imap) # no request
```

Configuring a POP3 Probe

You can configure Post Office Protocol 3 (POP3) probes to initiate a session and send the configured credentials. The ACE can also send a configured command. Based on the server response, the ACE marks the probe as passed or failed.

You can create a POP probe and access its configuration mode by using the **probe pop** command. The syntax of this command is as follows:

```
probe pop name
```

For the *name* argument, enter an identifier for the POP probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a POP probe named PROBE12 and access its mode, enter:

```
host1/Admin(config) # probe pop PROBE12  
host1/Admin(config-probe-pop) #
```

To configure attributes for a POP probe, see the following topics:

- [Configuring the Credentials for a Probe](#)
- [Configuring the Request Command for the Probe](#)

You can also configure the general attributes described in the “[Configuring General Probe Attributes](#)” section and configure connection termination as described in the “[Configuring the Termination of the TCP Connection](#)” section.

Configuring the Credentials for a Probe

The credentials for a probe are the username and password used for authentication on the server. You can configure the credentials for the probe by using the **credentials** command. The syntax of this command is as follows:

```
credentials username [password]
```

The arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—(Optional) Password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-pop) # credentials ENG1 TEST
```

To delete the credentials for the probe, use the **no credentials** command. For example, enter:

```
host1/Admin(config-probe-pop) # no credentials
```

Configuring the Request Command for the Probe

You can configure the request method used by a POP probe by using the **request command** command. The syntax of this command is as follows:

request command *command*

The *command* argument is the request method command for the probe. Enter a text string with a maximum of 32 alphanumeric characters with no spaces.

For example, to configure the last request command for a POP probe, enter:

```
host1/Admin(config-probe-pop) # request method last
```

To remove the request command for the probe, use the **no request** command. For example, enter:

```
host1/Admin(config-probe-pop) # no request
```

Configuring a SIP Probe

You can use a SIP probe to establish a TCP or UDP connection and send an OPTIONS request packet to the user agent on the server. The ACE can compare the response with the configured response code or expected string, or both, to determine the probe has succeeded.

For example, if you configure an expected string and status code and the ACE finds them both in the response, the server is marked as passed. However, if the ACE does not receive either the server response string or the expected status code, it marks the server as failed.

**Note**

If you do not configure an expected status code, any response from the server is marked as failed.

You can create a SIP probe and access its configuration mode by using the **probe sip {tcp | udp} name** command. The syntax of this command is as follows:

```
probe sip {tcp | udp} name
```

The keywords and argument are as follows:

- **tcp**—Creates the probe for a TCP connection.
- **udp**—Creates the probe for a UDP connection.
- *name*—Identifier of the probe. Enter an unquoted text string with a maximum of 64 alphanumeric characters.

For example, to define a SIP probe using TCP named probe13 and access its mode, enter:

```
host1/Admin(config)# probe sip tcp probe13
host1/Admin(config-probe-sip-tcp)#
```

To define a SIP probe using UDP named probe14 and access its mode, enter:

```
host1/Admin# probe sip udp probe14
host1/Admin(config-probe-sip-udp)#
```

You can configure most general probe attributes described in the “[Configuring General Probe Attributes](#)” section. If the probe uses a:

- TCP connection, as configured through the **tcp** keyword, you can configure the TCP attributes in the “[Configuring a TCP Probe](#)” section.
- UDP connection, as configured through the **udp** keyword, you can configure the UDP attributes in the “[Configuring a UDP Probe](#)” section.



Note

The send data option of UDP probes is not applicable to SIP UDP probes.

You can also use the additional commands to configure attributes for a SIP probe. The following sections describes how to configure additional probe attributes:

- [Configuring the Request Method for the Probe](#)
- [Configuring the Status Code from the Destination Server](#)

Configuring the Request Method for the Probe

By default, the SIP request method is the OPTIONS method. Currently, this is the only method available for SIP probes. You can configure the OPTIONS request method that is used by the probe by using the **request method options** command. The syntax of this command is as follows:

request method options

For example, to configure the OPTIONS method, enter:

```
host1/Admin(config-probe-sip-tcp)# request method options
```

To reset the method for the probe to OPTIONS, use the **no request method** command. For example, enter:

```
host1/Admin(config-probe-sip-tcp)# no request method
```

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, there are no status codes configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

expect status *min_number max_number*

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For SIP, the expected status code is 200, which indicates a successful probe. For example, to configure an expected status code of 200 that indicates that the request was successful, enter:

```
host1/Admin(config-probe-sip-tcp)# expect status 200 200
```

Configuring an RTSP Probe

You can configure an RTSP probe to establish a TCP connection and send a request packet to the server. The ACE compares the response with the configured response code to determine whether the probe has succeeded. When configuring these probes, you use the **probe rtsp *name*** command to create the probe and access probe configuration mode.

For example, to define an RTSP probe named `probe15` and access its mode, enter:

```
host1/Admin(config)# probe rtsp probe15
host1/Admin(config-probe-rtsp)#
```

After you create an RTSP probe, you can configure the general probe attributes described in the “[Configuring General Probe Attributes](#)” section. You can also configure the ACE to terminate a TCP connection by sending a RST and an expected response string as described in the “[Configuring a TCP Probe](#)” section.

You can also use the additional commands to configure attributes for an RTSP probe. The following topics describe how to configure additional probe attributes:

- [Configuring the Request Method](#)
- [Configuring the Header Field for the RTSP Probe](#)
- [Configuring the Status Code from the Destination Server](#)

Configuring the Request Method

By default, the RTSP request method is the OPTIONS method. You can also configure the DESCRIBE method. You can configure the request method that is used by the probe by using the **request method** command. The syntax of this command is as follows:

```
request method { options | describe url url_string }
```

The keywords and arguments are as follows:

- **options**—Configures the OPTIONS request method. This is the default method. The ACE uses the asterisk (*) request URL for this method.
- **describe url *url_string***—Configures the DESCRIBE request method. The *url_string* is the URL request for the RTSP media stream on the server. Enter a URL string with a maximum of 255 characters.

For example, to configure an RTSP probe to use the URL for `rtsp://media/video.smi`, enter:

```
host1/Admin(config-probe-rtsp) # request method describe url
rtsp://192.168.10.1/media/video.smi
```

For example, to configure an RTSP probe to use the PATH for `rtsp://media/video.smi`, enter:

```
host1/Admin(config-probe-rtsp) # request method describe path
/media/video.smi
```

In the example shown above, the IP address is taken from the probe target IP address.

To reset the default OPTIONS request method, use the **no request method** or the **request method options** command. For example, enter:

```
host1/Admin(config-probe-rtsp) # no request method
```

Configuring the Header Field for the RTSP Probe

You can configure a header field value for the probe by using the **header** command. The syntax of this command is as follows:

```
header { require | proxy-require } header-value value
```

The keywords and arguments are as follows:

- **require**—Specifies the Require header.
- **proxy-require**—Specifies the Proxy-Require header.
- **header-value** *value*—Specifies the header value. For the value, enter an alphanumeric string with no spaces and a maximum of 255 characters.

For example, to configure the REQUIRE header with a field value of `implicit-play`, enter:

```
host1/Admin(config-probe-rtsp) # header require header-value
implicit-play
```

To remove the header configuration for the probe, use the **no** form of the **header** command. For example, to remove a Require header, enter:

```
host1/Admin(config-probe-rtsp) # no header require
```

To remove a Proxy-Require header, enter:

```
host1/Admin(config-probe-rtsp) # no header proxy-require
```

Configuring the Status Code from the Destination Server

When the ACE receives a response from the server, it expects a status code to mark a server as passed. By default, no status codes are configured on the ACE. If you do not configure a status code, any response code from the server is marked as failed.

You can configure a single status code or a range of code responses that the ACE expects from the probe destination by using the **expect status** command. You can specify multiple status code ranges with this command by entering the command with different ranges separately. The syntax of this command is as follows:

```
expect status min_number max_number
```

The arguments are as follows:

- *min_number*—Single status code or the lower limit of a range of status codes. Enter an integer from 0 to 999.
- *max_number*—Upper limit of a range of status codes. Enter an integer from 0 to 999. When configuring a single code, reenter the *min_number* value.

For example, to configure an expected status code of 200 that indicates that the request was successful, enter:

```
host1/Admin(config-probe-rtsp) # expect status 200 200
```

To configure a range of expected status codes from 100 to 200, enter:

```
host1/Admin(config-probe-rtsp) # expect status 100 200
```

To configure multiple ranges of expected status codes from 100 to 200 and from 250 to 305, you must configure each range separately. For example, enter:

```
host1/Admin(config-probe-rtsp) # expect status 100 200  
host1/Admin(config-probe-rtsp) # expect status 250 305
```

To remove a single expected status code, use the **no expect status** command. For example, to remove the expected status code of 200, enter:

```
host1/Admin(config-probe-rtsp) # no expect status 200 200
```

To remove a range of expected status codes, enter the range using the **no expect status** command. For example, to remove a range of 250 to 302 from a range of 250 to 305, enter:

```
host1/Admin(config-probe-rtsp) # no expect status 250 305
```

To remove multiple ranges of expected status codes, you must remove each range separately. For example, if you have set two different ranges (100 to 200 and 250 to 305), enter:

```
host1/Admin(config-probe-rtsp) # no expect status 100 200
host1/Admin(config-probe-rtsp) # no expect status 250 305
```

Configuring a RADIUS Probe

A RADIUS probe sends a query using a configured username, password, and shared secret to a RADIUS server. If the server is up, it is marked as passed. If you configure a Network Access Server (NAS) address, the ACE uses it in the outgoing packet. Otherwise, the ACE uses the IP address associated with the outgoing interface as the NAS address.

You can create the RADIUS probe and access its configuration mode by using the **probe radius** command. The syntax of this command is as follows:

```
probe radius name
```

For the *name* argument, enter an identifier of the RADIUS probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a RADIUS probe named PROBE and access its mode, enter:

```
host1/Admin(config) # probe radius PROBE
host1/Admin(config-probe-radius) #
```

To configure probe attributes for a RADIUS probe, see the following topics:

- [Configuring the Credentials and Shared Secret for a Probe](#)
- [Configuring the Network Access Server IP Address](#)

You can also configure the general attributes described in the “[Configuring General Probe Attributes](#)” section.

Configuring the Credentials and Shared Secret for a Probe

The credentials for a probe are the username and password used for authentication on the server and an optional shared secret to allow probe access to the RADIUS server. You can configure the credentials for the probe by using the **credentials** command. The syntax of this command is as follows:

```
credentials username password [secret shared_secret]
```

The keywords and arguments are as follows:

- *username*—User identifier used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- *password*—Password used for authentication. Enter an unquoted text string with a maximum of 64 alphanumeric characters.
- **secret** *shared_secret*—(Optional) Specifies the shared secret. Enter the shared secret as a case-sensitive string with no spaces and a maximum of 64 alphanumeric characters.

For example, to configure the username ENG1 and a password TEST, enter:

```
host1/Admin(config-probe-radius)# credentials ENG1 TEST
```

To delete the credentials for the probe, use the **no credentials** command. For example, enter:

```
host1/Admin(config-probe-radius)# no credentials
```

Configuring the Network Access Server IP Address

If a Network Access Server (NAS) address is not configured for the RADIUS probe, the ACE uses the IP address associated with the outgoing interface as the NAS address. You can configure an NAS address by using the **nas ip address** command. The syntax of this command is as follows:

```
nas ip address ip_address
```

The *ip_address* argument is the NAS IP address. Enter a unique IPv4 address in dotted-decimal notation (for example, 192.8.12.15).

For example, to configure a NAS address of 192.8.12.15, enter:

```
host1/Admin(config-probe-radius)# nas ip address 192.8.12.15
```

To remove the NAS IP address, use the **no nas ip address** command. For example, enter:

```
host1/Admin(config-probe-radius)# no nas ip address
```

Configuring an SNMP-Based Server Load Probe

An SNMP-based server load probe establishes a UDP connection and allows you to configure a maximum of eight SMNP OID queries to probe the server. The ACE weighs and averages the load information that is retrieved and uses it as input to the least-loaded algorithm for load-balancing decisions. If the retrieved value is within the configured threshold, the server is marked as passed. If the threshold is exceeded, the server is marked as failed.

When configuring these probes, you use the **probe snmp name** command to create the probe and access probe configuration mode.

For example, to define an SNMP probe named probe18 and access its mode, enter:

```
host1/Admin(config)# probe snmp probe18
host1/Admin(config-probe-snmp)#
```

You can configure the general attributes described in the “[Configuring General Probe Attributes](#)” section. You can also use the additional commands to configure attributes for an SNMP probe. The following topics describe how to configure additional probe attributes:

- [Configuring the Community String](#)
- [Configuring the SNMP Version](#)
- [Configuring the OID String](#)
- [Configuring the OID Value Type](#)
- [Configuring the OID Threshold](#)
- [Configuring the OID Weight](#)

Configuring the Community String

The ACE probes access the server through its community string. By default, the community string is not set. You can configure the community string by using the **community** command. The syntax of the command is as follows:

```
community text
```

The *text* argument is the name of the SNMP community string for the server. Enter a text string with a maximum of 255 alphanumeric characters.

For example, to configure the private community string, enter:

```
host1/Admin(config-probe-snmp) # community private
```

To remove the community string, enter:

```
host1/Admin(config-probe-snmp) # no community
```

Configuring the SNMP Version

The version in the SNMP OID query sent to the server indicates the supported SNMP version. By default, the probe supports SNMP version 1.

You can configure the version of SNMP that the probe supports by using the **version** command. The syntax of this command is as follows:

```
version {1 | 2c}
```

The keywords are as follows:

- **1**—Specifies that the probe supports SNMP version 1 (default).
- **2c**—Specifies that the probe supports SNMP version 2c.

For example, to configure SNMP version 2c, enter:

```
host1/Admin(config-probe-snmp) # version 2c
```

To reset the default setting of SNMP version 1, enter:

```
host1/Admin(config-probe-snmp) # no version
```

Configuring the OID String

When the ACE sends a probe with an SNMP OID query, the ACE uses the retrieved values as input to the least-loaded algorithm for load-balancing decisions. Least-loaded load balancing bases the server selection on the server with the lowest load value. You can configure a maximum of eight OIDs.

To configure the OID string and access probe SNMP OID configuration mode, use the **oid** command in probe SNMP configuration mode. The syntax of the command is as follows:

oid *string*

The *string* argument is the OID that the probe uses to query the server for a value. Enter an unquoted string with a maximum of 255 alphanumeric characters in dotted-decimal notation. The OID string is based on the server type. The dots (.) in the string count as characters. For example, if the OID string is 10.0.0.1.1, then the character count is 10.

Accessing probe-snmp-oid configuration mode allows you to configure the threshold, the OID value type, and the weight assigned to the OID, as described in the following sections.



Note

If you configure more than one OID and they are used in a load-balancing decision, you must configure a weight value.

For example, to configure the OID string .1.3.6.1.4.1.2021.10.1.3.1 for a 1-minute average of CPU load on a Linux server and access probe-snmp-oid configuration mode, enter:

```
host1/Admin(config-probe-snmp) # oid .1.3.6.1.4.1.2021.10.1.3.1  
host1/Admin(config-probe-snmp-oid) #
```

To remove the OID string, enter:

```
host1/Admin(config-probe-snmp) # no oid .1.3.6.1.4.1.2021.10.1.3.1
```

Configuring the OID Value Type

By default, the retrieved OID value type is a percentile value. To configure the OID value type as absolute and define its maximum expected value, use the **type absolute max** command in probe SNMP OID configuration mode. The syntax of this command is as follows:

```
type absolute max integer
```

The *integer* argument specifies the maximum expected absolute value for the OID. Enter an integer from 1 to 4294967295. By default, the OID value is a percentile value.



Note

When you configure the **type absolute max** command, we recommend that you also configure the value for the **threshold** command because the default threshold value is set to the integer value specified in the **type absolute max** command.

For example, to configure an absolute value type with its maximum expected value of 65535, enter:

```
host1/Admin(config-probe-snmp-oid) # type absolute max 65535
```

To reset the OID value type to a percentile value, enter:

```
host1/Admin(config-probe-snmp) # no type
```



Note

The **no type** command resets the OID type to percentile and sets the **threshold** command to a value of 100.

Configuring the OID Threshold

The OID threshold specifies the value to take the server out of service.

- When the OID value is based on a percentile, the default threshold value is 100.
- When the OID is based on an absolute value, the threshold range is based on what you specified in the **type absolute max** command (see the [“Configuring the OID Threshold”](#) section).

To configure the threshold, use the **threshold** command in probe SNMP OID configuration mode. The syntax of this command is as follows:

threshold *integer*

The *integer* argument specifies the threshold value to take the server out of service.

- When the OID value is based on a percentile, enter an integer from 1 to 100, with a default value of 100.
- When the OID is based on an absolute value, the threshold range is from 1 to the maximum value that you specified in the **type absolute max** command.

For example, to configure a threshold of 50, enter:

```
host1/Admin(config-probe-snmp-oid)# threshold 50
```

To reset the OID threshold to its default value, enter:

```
host1/Admin(config-probe-snmp)# no threshold
```

Configuring the OID Weight

You must specify an OID weight when you configure more than one OID and they need to be used in a load-balancing decision. To configure the weight for the OID, use the **weight** command in probe-snmp-oid configuration mode. The syntax of this command is as follows:

weight *integer*

The *integer* argument specifies the weight for the OID. Enter an integer from 1 to 16000. By default, an equal weight is given to each configured OID.



Note

If you configure more than one OID and they are used in a load-balancing decision, you must configure a weight value.

For example, to configure the weight of 10000, enter:

```
host1/Admin(config-probe-snmp-oid)# weight 10000
```

To reset the default behavior an equal weight given to each configured OID, enter:

```
host1/Admin(config-probe-snmp)# no weight
```

Configuring a Scripted Probe

Scripted probes allow you to run a script to execute the probe that you created for health monitoring. You can author specific scripts with features not present in standard health probes. To configure a scripted probe, you need to do the following:

- Copy the script file to the ACE disk0: file system
- Load the script file
- Associate the script with the scripted probe

The ACE allows the configuration of 256 unique script files.

You can also use the Cisco-supplied scripts located in the `probe: directory` in the ACE. For more information about these scripts, see the “[Scripts Overview](#)” section in [Appendix A, “Using TCL Scripts with the ACE”](#).



Note

The ACE can simultaneously execute only 200 scripted probe instances. When this limit is exceeded, the **show probe detail** command displays the “Out-of-Resource: Max. script-instance limit reached” error message in the Last disconnect err field and the out-of-sockets counter increments.

For information about copying and loading a script file on the ACE, see [Appendix A, “Using TCL Scripts with the ACE”](#).

You can create a scripted probe and access the scripted probe configuration mode by using the **probe scripted** command. The syntax of this command is as follows:

```
probe scripted name
```

For the *name* argument, enter the identifier of the scripted probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to define a scripted probe named PROBE19 and access its mode, enter:

```
host1/Admin(config)# probe scripted PROBE19
host1/Admin(config-probe-scrptd)#
```

To configure the scripted probe attributes, see the “[Associating a Script with a Probe](#)” section.

You can also configure the general commands described in the “[Configuring General Probe Attributes](#)” section.

Associating a Script with a Probe

Scripted probes run probes from a configured script to perform health probing. You can also configure arguments that are passed to the script. Before you can associate a script file with a probe, you must copy and load the script on the ACE. For information about copying and loading a script, see [Appendix A, “Using TCL Scripts with the ACE”](#).

Use the script command to specify the name of the script file and the arguments to be passed to the script.

The syntax of this command is as follows:

```
script script_name [script_arguments]
```

The arguments are as follows:

- *script_name*—Name of the script. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters.
- *script_arguments*—(Optional) Data sent to the script. Enter a text string with a maximum of 255 alphanumeric characters including spaces and quotes. Separate each argument by a space. If a single argument contains spaces, enclose the argument string in quotes.

For example, to configure the script name of PROBE-SCRIPT and arguments of ??, enter:

```
host1/Admin(config-probe-scrptd)# script PROBE-SCRIPT ??
```

To remove the script and its arguments from the configuration, use the **no script** command. For example, enter:

```
host1/Admin(config-probe-scrptd)# no script
```

Example of a UDP Probe Load-Balancing Configuration

The following example shows a running configuration that load balances DNS traffic across multiple real servers, and transmits and receives UDP data that spans multiple packets. The configuration uses a UDP health probe. The UDP probe configuration appears in bold in the example.

```
access-list ACL1 line 10 extended permit ip any any

probe udp UDP
  interval 5
  passdetect interval 10
  description THIS PROBE IS INTENDED FOR LOAD BALANCING DNS TRAFFIC
  port 53
  send-data UDP_TEST

rserver host SERVER1
  ip address 192.168.252.245
  inservice
rserver host SERVER2
  ip address 192.168.252.246
  inservice
rserver host SERVER3
  ip address 192.168.252.247
  inservice

serverfarm host SFARM1
  probe UDP
  rserver SERVER1
    inservice
  rserver SERVER2
    inservice
  rserver SERVER3
    inservice

class-map match-all L4UDP-VIP_114:UDP_CLASS
  2 match virtual-address 192.168.120.114 udp eq 53
policy-map type loadbalance first-match L7PLBSF_UDP_POLICY
  class class-default
    serverfarm SFARM1
policy-map multi-match L4SH-Gold-VIPs_POLICY
  class L4UDP-VIP_114:UDP_CLASS
    loadbalance vip inservice
    loadbalance policy L7PLBSF_UDP_POLICY
    loadbalance vip icmp-reply
  nat dynamic 1 vlan 120
  connection advanced-options 1SECOND-IDLE
```

```
interface vlan 120
  description Upstream VLAN_120 - Clients and VIPs
  ip address 192.168.120.1 255.255.255.0
  fragment chain 20
  fragment min-mtu 68
  access-group input ACL1
  nat-pool 1 192.168.120.70 192.168.120.70 netmask 255.255.255.0 pat
  service-policy input L4SH-Gold-VIPs_POLICY
  no shutdown
ip route 10.1.1.0 255.255.255.0 192.168.120.254
```

Configuring KAL-AP

A keepalive-appliance protocol (KAL-AP) on the ACE allows communication between the ACE and the Global Site Selector (GSS), which send KAL-AP requests, to report the server states and loads for global-server load-balancing (GSLB) decisions. The ACE uses KAL-AP through a UDP connection to calculate weights and provide information for server availability to the KAL-AP device. The ACE acts as a server and listens for KAL-AP requests. When KAL-AP is initialized on the ACE, the ACE listens on the standard 5002 port for any KAL-AP requests. You cannot configure any other port.

The ACE supports VIP-based and tag-based KAL-AP probes. For a VIP-based KAL-AP, when the ACE receives a `kal-ap-by-vip` request, it verifies whether the VIP addresses are active in all Layer 3 class maps that are configured with the addresses. The ACE ignores all other protocol-specific information for the VIP addresses. For each Layer 3 class map, the ACE locates the associated Layer 7 policies and associated real servers in server farms. The ACE determines the total number of servers associated with these VIPs and those servers in the Operational state.

The ACE calculates a load number from 0 to 255 and reports the server availability of the VIP to the KAL-AP device. A load value of 0 indicates that the VIP address is not available. This value is also sent in the case of any VIP lookup failures. A load value of 1 is reserved to indicate that the VIP is offline and not available for use. Valid load values are from 2 to 255. A load value of 2 indicates that the VIP is least loaded and a load value of 255 indicates that the VIP is fully loaded. For example, if the total number of servers is 10 and only 5 are operational, the load value is 127.

**Note**

If the same real server is associated with more than one server farm, the ACE includes it twice in the calculation.

For a tag-based KAL-AP, a domain associated with a VIP address corresponds to a tag in the ACE. When the ACE receives a kal-ap-by-tag request, the process is similar to VIP-based KAL-AP probes. The load calculation considers the Layer 3 class map, server farm, and real server objects.

Under a domain, all other objects are ignored during the load calculation. The calculation for the domain is similar to a VIP address. The ACE gathers the server availability information for any Layer 3 VIP addresses within the domain. The ACE considers all of the server farms associated with the domain. If the real servers are in a domain, the ACE adds them to the current total and then performs a division to determine their availability as tag objects. The ACE reports this final number in the KAL-AP response.

This section contains the following topics:

- [Enabling KAL-AP on the ACE](#)
- [Configuring a KAL-AP VIP Address](#)
- [Configuring KAL-AP Tags as Domains](#)
- [Configuring Secure KAL-AP](#)
- [Displaying Global-Server Load-Balancing Load Information](#)
- [Displaying Global-Server Load-Balancing Statistics](#)

Enabling KAL-AP on the ACE

To enable KAL-AP on the ACE, you must configure a management class map and policy map, and apply it to the appropriate interface. The KAL-AP server listens on the standard 5002 port to all KAL-AP requests.

You can configure the class map for KAL-AP over UDP management access by using the **match protocol kalap-udp** command in the class map management configuration mode. The syntax of this command is as follows:

```
match protocol kalap-udp any | [source-address ip_address subnet_mask]
```

The keywords and arguments are as follows:

- **any**—Specifies any client source address for the management traffic classification.
- **source-address**—Specifies a client source host IP address and subnet mask as the network traffic matching criteria. As part of the classification, the ACE implicitly obtains the destination IP address from the interface on which you apply the policy map.
- *ip_address*—Source IP address of the client. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).
- *mask*—Subnet mask of the client entry in dotted-decimal notation (for example, 255.255.255.0).

For example, to specify a KAL-AP class map from any source IP address, enter:

```
host1/Admin(config)# class-map type management KALAP-CM
host1/Admin(config-cmap-mgmt)# match protocol kalap-udp any
host1/Admin(config-cmap-mgmt)# exit
host1/Admin(config)#
```

To remove the class map, enter:

```
host1/Admin(config-cmap-mgmt)# no match protocol kalap-udp
source-address any
```

After you create the KAL-AP class map, create a KAL-AP management policy map and apply the class map to it. To create the policy map and access policy map management configuration mode, use the **policy-map type management** command in configuration mode. For example, to create the KALAP-MGMT management policy map and apply the KALAP-CM class map to it, enter:

```
host1/Admin(config)# policy-map type management KALAP-MGMT
host1/Admin(config-pmap-mgmt)# class KALAP-CM
host1/Admin(config-cmap-mgmt)# permit
host1/Admin(config-cmap-mgmt)# exit
host1/Admin(config)#
```

To apply the policy map to an interface, use the **interface vlan** command in configuration mode. For example, to apply the KALAP-MGMT policy map to VLAN interface 10, enter:

```
host1/Admin(config)# interface vlan 10
host1/Admin(config-if)# ip address 10.1.0.1 255.255.255.0
host1/Admin(config-if)# service-policy input KALAP-MGMT
host1/Admin(config-if)# no shutdown
host1/Admin(config-if)# exit
host1/Admin(config)#
```

**Note**

When you modify or remove a KAL-AP policy, you must clear the existing KAL-AP connections manually.

Configuring a KAL-AP VIP Address

You can configure VIP-based KAL-AP by configuring a Layer 3/4 class map that contains a VIP address match statement. You can define a 3-tuple flow of VIP address, protocol, and port as matching criteria by using the **match virtual-address** command in class map configuration mode. You can configure multiple match criteria statements to define the VIPs for server load balancing. The syntax of this command is as follows:

```
[line_number] match virtual-address vip_address {[mask] | any | {tcp | udp
  {any | eq port_number | range port1 port2}} | protocol_number}
```

For information on the keywords and arguments, see the “[Defining VIP Address Match Criteria](#)” section in [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).

**Note**

For KAL-AP, the ACE verifies whether the VIP addresses are active in all Layer 3 class maps that are configured with the addresses. It ignores all other protocol-specific information for the VIP addresses.

For example, to create a class map VIP-20 that matches traffic destined to VIP address 10.10.10.10 with a wildcard value for the IP protocol value (TCP or UDP), enter:

```
host1/Admin(config)# class-map VIP-20
host1/Admin(config-cmap)# match virtual-address 10.10.10.10 any
```

To remove the VIP match statement from the class map, enter:

```
host1/Admin(config-cmap)# no match virtual-address 10.10.10.10 any
```

Configuring KAL-AP Tags as Domains

You can configure KAL-AP tags as domains by using the **domain** command in configuration mode. You can configure a maximum of 64 KAL-AP tag domains per context. The syntax of this command is as follows:

```
domain name
```

The *name* is the name of the KAL-AP tag.

**Note**

For the domain load calculation, the ACE considers the Layer 3 class map, server farm, and real server objects. All other objects under the domain are ignored during the calculation.

For example, to configure KAL-AP-TAG1 as a domain, enter:

```
host1/Admin(config)# domain KAL-AP-TAG1
```

After you create the domain, use the **add-object class-map** command in domain configuration mode to add each class map that you want to associate with the tag domain. For example, to add the VIP-20 and VIP-71 class maps to the tag domain, enter:

```
host1/Admin(config-domain)# add-object class-map VIP-20  
host1/Admin(config-domain)# add-object class-map VIP-71
```

To remove the domain, enter:

```
host1/Admin(config)# no domain KAL-AP-TAG1
```

For more information about configuring class maps, see [Chapter 3, “Configuring Traffic Policies for Server Load Balancing.”](#) For more information about configuring domains, see the *Cisco 4700 Series Application Control Engine Appliance Virtualization Configuration Guide*.

Configuring Secure KAL-AP

The ACE supports secure KAL-AP for MD5 encryption of data between it and the GSS. For encryption, you must configure a shared secret as a key for authentication between the GSS and the ACE context.

To configure secure KAL-AP on the ACE, access KAL-AP UDP configuration mode through the **kalap udp** command in configuration mode. The syntax of this command is as follows:

kalap udp

For example, enter:

```
host1/Admin(config)# kalap udp
host1/Admin(config-kalap-udp)#
```

To remove the KAL-AP configuration and all VIP entries, enter the following command:

```
host1/Admin(config)# no kalap udp
```

In this mode, you enable secure KAL-AP by configuring the VIP address to the GSS and the shared secret through the **ip address** command. The syntax of this command is as follows:

ip address ip_address encryption md5 secret

The keywords and arguments are as follows:

- *ip_address*—The VIP address for the GSS. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).
- **encryption**—Specifies the encryption method.
- **md5**—Specifies the MD5 encryption method.
- *secret*—Shared secret between the KAL-AP device and the ACE. Enter the shared secret as a case-sensitive string with no spaces and a maximum of 31 alphanumeric characters.

For example, to enable secure KAL-AP and configure the VIP address for the GSS and shared secret, enter:

```
host1/Admin(config-kalap-udp)# ip address 10.1.0.1 encryption md5
andromeda
```

To disable secure KAL-AP, use the **no** form of the **ip address** command. For example, enter:

```
host1/Admin(config-kalap-udp)# no ip address 10.1.0.1
```

Displaying Global-Server Load-Balancing Load Information

You can display the latest load information for a VIP address, domain name, or VIP tag name provided to the KAL-AP request by using the **show kalap udp load** command in Exec mode. The syntax of the command is as follows:

```
show kalap udp load {all | domain name | vip ip_address}
```

The keywords and arguments are as follows:

- **all**—Displays the latest load information for all VIP addresses and domains.
- **domain *name***—Displays the latest load information for the specified domain name.
- **vip *ip_address***—Displays the latest load information for the specified VIP address. Enter the IP address in dotted-decimal notation (for example, 192.168.11.1).

The output fields for the **show kalap udp load** command display the VIP address, or domain name, its load value, and the time stamp.

For example, to display the latest load information for all VIP addresses and domains, enter:

```
host1/Admin# show kalap udp load all
```

For example, to display the latest load information to the KAL-AP request for VIP address 10.10.10.10, enter:

```
host1/Admin# show kalap udp load vip 10.10.10.10
```

To display the latest load information to the KAL-AP request for domain KAL-AP-TAG1, enter:

```
host1/Admin# show kalap udp load domain KAL-AP-TAG1
```

Displaying Global-Server Load-Balancing Statistics

You can display the global-server load-balancing statistics per context by using the **show stats kalap** command in Exec mode. The syntax of the command is as follows:

```
show stats kalap
```

For example, enter:

```
host1/Admin# show stats kalap
```

Table 4-2 lists the output fields displayed by this command.

Table 4-2 *Field Descriptions for the show stats kalap Command*

Field	Descriptionh
Total bytes received	Total number of bytes received.
Total bytes sent	Total number of bytes sent.
Total requests received	Total number of requests received.
Total responses sent	Total number of responses sent.
Total requests successfully received	Total number of requests successfully received.
Total responses successfully sent	Total number of responses successfully sent.
Total secure requests received	Total number of secure requests received.
Total secure responses sent	Total number of secure responses sent.
Total requests with errors	Total number of requests with errors.
Total requests with parse errors	Total number of requests with parse errors.
Total response transfer errors	Total number of response transfer errors.

You can clear the global-server load-balancing statistics per context by using the **clear stats kalap** command in Exec mode. For example, enter:

```
host1/Admin# clear stats kalap
```

Displaying Probe Information

You can display configuration information and statistics for a probe by using the **show probe** command in Exec mode. The syntax of this command is as follows:

```
show probe [probe_name] [detail]
```

The argument and option are as follows:

- *probe_name*—(Optional) Information for the specified probe name.
- **detail**—(Optional) Displays detailed probe configuration and statistic information.

If you do not enter a probe name, this command shows a summary of information for all configured probes. For example, enter:

```
host1/Admin# show probe
```

You can also display configuration information for all probes by using the **show running-config probe** command.

For example, enter:

```
host1/Admin# show running-config probe
```

[Table 4-3](#) describes the fields in the **show probe** command output including additional output provided by the **detail** option.



Note

Probe instances will not be displayed in the **show probe** command output for any type of probe (real server, server farm, real server in a server farm, predictor, or probe configured on the active FT group member when the real server is out-of-service. In this case, only the associations will be listed in the show probe command output. This occurs to maintain consistency between probe instances with no port inheritance and probe instances with port inheritance (see the [“Port Number Inheritance for Probes”](#) section).

Table 4-3 *Field Descriptions for the show probe Command*

Field	Description
Probe	Name of the probe.
Type	Probe type.

Table 4-3 *Field Descriptions for the show probe Command (continued)*

Field	Description
State	Whether the probe is active or inactive.
Description	Configured description for the probe (detail option output).
Port	Port number that the probe uses. By default, the probe uses the port number based on its type. If the probe's port number is inherited (see the “Port Number Inheritance for Probes” section), the inherited port number appears in this field.
Address	Destination address for the probe.
Addr type	Address type.
Interval	Time interval in seconds that the ACE sends probes to a server marked as passed.
Pass intvl	Time period in seconds to send a probe to a failed server.
Pass count	Consecutive number of passed probes before marking the server as passed.
Fail count	Consecutive number of failed probes before marking the server as failed.
Recv timeout	Time period in seconds to receive a server response to the probe.
DNS domain	Domain name configured for the probe (detail option output for a DNS probe).
HTTP method	HTTP method and URL used by the probe, GET or HEAD (detail option output for HTTP and HTTPS probes).
HTTP URL	URL used by the probe with the HTTP method (detail option output for HTTP and HTTPS probes).
RTSP method	RTSP method and URL used by the probe (detail option output for RTSP probes).
RTSP URL	URL used by the probe with the RTSP method (detail option output for RTSP probes).

Table 4-3 *Field Descriptions for the show probe Command (continued)*

Field	Description
IMAP mailbox	Mailbox username where the probe retrieves e-mail (detail option output for IMAP probes).
IMAP/POP command	Request method command for the probe (detail option output for IMAP and POP probes).
NAS address	Network Access Server (NAS) address for the RADIUS server (detail option output for RADIUS probes).
Script filename	Filename for the script (detail option output for scripted probes).
Conn termination	TCP connection termination type, GRACEFUL or FORCED (detail option output for ECHO TCP, Finger, FTP, HTTP, HTTPS, IMAP, POP, SMTP, TCP, and Telnet probes).
Expect/Search offset	Number of characters into the received message or buffer to start searching for the expect regex expression (detail option output for HTTP, HTTPS, RTSP, SIP, TCP, and UDP probes).
Request-method	Request method for SIP probes displayed in the detail option output. Currently, the OPTIONS method is the only method available for SIP probes.
Expect regex	Configured expected response data from the probe destination (detail option output for HTTP, HTTPS, RTSP, SIP, TCP, and UDP probes).
Open timeout	Time interval in seconds that the probe waits to open and establish the connection with the server (detail option output for Finger, FTP, HTTP, HTTPS, IMAP, POP, scripted, RTSP, SMTP, TCP, and Telnet probe).
Send data	ASCII data that the probe sends (detail option output for ECHO, Finger, HTTP, HTTPS, RTSP, TCP, and UDP probes).
Version	SNMP version in the SNMP OID query sent to the server that indicates the supported version (detail option output for SNMP probes).

Table 4-3 Field Descriptions for the show probe Command (continued)

Field	Description
Community	SNMP community string (detail option output for SNMP probes).
OID string	The configured OID (detail option output for SNMP probes).
Type	The OID value type, absolute or percentile, for the retrieved OID value (detail option output for SNMP probes).
Max value	The maximum expected load value for the OID load type (detail option output for SNMP probes).
Weight	The load weight for the OID (detail option output for SNMP probes).
Threshold	The threshold setting for the OID. When the threshold is exceeded, the OID is taken out of service (detail option output for SNMP probes).
probe results	
associations	Real server association for the probe.
ip-address	Destination or source address for the probe.
port	Port number for the probe.
porttype	Source of the probe's port number. This field identifies whether the probe's port number is inherited (see the “Port Number Inheritance for Probes” section). Possible values are: PROBE, REAL, VIP, or DEFAULT. Note A value of “--” is displayed for a server farm predictor method, a probe assigned to a standalone real server, or a probe configured on the active FT group member in a redundant configuration
probes	Total number of probes.
failed	Total number of failed probes.
passed	Total number of passed probes.

Table 4-3 *Field Descriptions for the show probe Command (continued)*

Field	Description
health	Health of the probe. Possible values are PASSED or FAILED.
Additional detail option output for scripted probes:	
Socket state	Socket state.
No. Passed states	Number of passed states.
No. Failed states	Number of failed states.
No. Probes skipped	Number of skipped probes. A skipped probe occurs when the ACE does not send out a probe because the scheduled interval to send a probe is shorter than it takes to complete the execution of the probe; the send interval is shorter than the open timeout or receive timeout interval. When a probe is skipped or an internal error is displayed by the show probe detail command, the state of the probe does not change. If it fails, it remains as failed.
Last status code	Last exit code (see Table A-7).
Last disconnect err	Message for the exit code for a scripted probe (see Table A-7) or an internal error.
Last probe time	Time stamp for the last probe.
Last fail time	Time stamp for the last failed probe.
Last active time	Time stamp for the last active time.
Internal error	Counter for the number of internal errors encountered.

Table 4-4 list the possible disconnect errors that can appear in the **show probe** output. For a list of disconnect messages for scripted probes, see Table A-7.

Table 4-4 ACE Probe Disconnect Errors

Probe Type	Error Message
All probe types	Unrecognized or invalid probe request.
	Connect error.
	Connection reset by server.
	Connection refused by server.
	Authentication failed.
	Unrecognized or invalid response.
	Out of memory, packets discarded.
	Server open timeout (no SYN ACK).
	Server reply timeout (no reply).
	Graceful disconnect timeout (no FIN ACK).
	Received Out-Of-Band data.
	User defined Reg-Exp was not found in host response.
	Expect status code mismatch.
Received invalid status code.	

Table 4-4 ACE Probe Disconnect Errors (continued)

Probe Type	Error Message
ICMP	ICMP Internal error.
	ICMP Internal error: Write failure.
	ICMP Internal error: Received bad FD.
	Host Unreachable, no route found to destination.
	ARP not resolved for dest-ip (destination IP address).
	Network down.
	Egress interface has no ip addr (IP address).
	ICMP Internal error: Data entry being modified.
	ICMP Internal error: No space, transmit path is full.
	ICMP Host unreachable.
	ICMP Dest unreachable.
	ICMP Time exceeded.
	ICMP Redirect.
	Received ICMP Echo Request.
	Received ICMP Stale pkt.
	Unexpected ICMP pkt type received.
	ICMP Pkt received is too short.
ICMP Pkt received is too long.	
HTTP/HTTPS	MD5 mismatch.
HTTPS	Invalid server greeting.
	Internal error: Failed to build a server query.

Table 4-4 ACE Probe Disconnect Errors (continued)

Probe Type	Error Message
SNMP	Last Disconnect Error: Sum of weights don't add up to max weight value.
	Last Disconnect Error: ASN encoding failed for the configured SNMP OID.
	Last Disconnect Error: Server load hit max value for type percentile.
	Last Disconnect Error: Server load hit max value for type absolute.
	Last Disconnect Error: Server load hit the threshold value.
	Last Disconnect Error: Failed to parse the PDU reply sent by the server.
	Last Disconnect Error: Unrecognized or invalid response.

To display the global statistics for a probe type, use the **show stats probe type** command in Exec mode. The syntax of this command is as follows:

```
show stats probe type probe_type
```

To view a list of probe types, enter:

```
host1/Admin# show stats probe type ?
```

For example, to view the global statistics for all DNS probes, enter:

```
host1/Admin# show stats probe type dns
```

Table 4-5 describes the fields in the **show stats probe type** command output.

Table 4-5 *Field Descriptions for the show stats probe type command*

Field	Description
Total probes sent	Total number of probes sent.
Total send failures	Total number of send failures. These failures are due to internal errors.
Total probes passed	Total number of passed probes.
Total probes failed	Total number of failed probes.
Total connect errors	Total number of connection errors.
Total conns refused	Total number of connections refused.
Total RST received	Total number of resets received.
Total open timeouts	Total number of open timeouts for the specified probe type.
Total receive timeouts	Total number of timeouts received.

Clearing Probe Statistics

This section describes the commands that you use to clear probe statistics, either for individual probes or for all probes in a context. It contains the following topics:

- [Clearing Statistics for Individual Probes](#)
- [Clearing All Probe Statistics in a Context](#)

Clearing Statistics for Individual Probes

You can clear the statistics displayed through the **show probe** command for a specific probe by using the **clear probe** command in Exec mode. The syntax of this command is as follows:

```
clear probe name
```

The *name* argument is the name of a configured probe.

For example, to clear the statistics for the DNS1 probe, enter:

```
host1/Admin# clear probe DNS1
```

**Note**

If you have redundancy configured, then you need to explicitly clear load-balancing statistics on both the active and the standby ACEs. Clearing statistics on the active appliance only will leave the standby appliance's statistics at the old values.

Clearing All Probe Statistics in a Context

You can clear all probe statistics in the current context by using the **clear stats probe** command in Exec mode. The syntax of this command is as follows:

```
clear stats probe
```

For example, enter:

```
host1/Admin# clear stats probe
```

**Note**

If you have redundancy configured, then you need to explicitly clear load-balancing statistics on both the active and the standby ACEs. Clearing statistics on the active appliance only will leave the standby appliance's statistics at the old values.

Where to Go Next

To learn how to use the Toolkit Command Language (TCL) to write probe scripts, see [Appendix A, “Using TCL Scripts with the ACE”](#). To configure stickiness (session persistence), see [Chapter 5, Configuring Stickiness](#). To configure firewall load balancing (FWLB), see [Chapter 6, Configuring Firewall Load Balancing](#).

■ Where to Go Next