



Using Toolkit Command Language (TCL) Scripts with the ACE

You can copy, upload, and execute Toolkit Command Language (TCL) scripts on the ACE. TCL is a widely used scripting language within the networking community. TCL also has large libraries of developed scripts that can easily be found from various sites. Using TCL scripts, you can write TCL scripts for customized health probes. The ACE also supports UDP socket functions.

**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.

This chapter provides information on scripts and contains the following sections:

- [Scripts Overview](#)
- [Probe Script Quick Start](#)
- [Copying and Loading Scripts on the ACE](#)
- [Configuring Health Probes for Scripts](#)
- [Writing Probe Scripts](#)
- [Displaying Script Information](#)
- [Debugging Probe Scripts](#)

Scripts Overview

The ACE supports several specific types of health probes (for example HTTP, TCP, or ICMP health probes) when you need to use a diverse set of applications and health probes to administer your network. The basic health probe types supported in the current ACE software release may not support the specific probing behavior that your network requires. To support a more flexible health-probing functionality, the ACE allows you to upload and execute TCL scripts on the ACE.

The TCL interpreter code in the ACE is based on Release 8.44 of the standard TCL distribution. You can create a script to configure health probes. Script probes operate similar to other health probes available in the ACE software. As part of a script probe, the ACE executes the script periodically, and the exit code that is returned by the executing script indicates the relative health and availability of specific real servers. For information on health probes, see [Chapter 4, Configuring Health Monitoring](#).

For your convenience, the following sample scripts for the ACE are available to support the TCL feature and are supported by Cisco TAC:

- CHECKPORT_STD_SCRIPT
- ECHO_PROBE_SCRIPT
- FINGER_PROBE_SCRIPT
- FTP_PROBE_SCRIPT
- HTTP_PROBE_SCRIPT
- HTTPCONTENT_PROBE
- HTTPHEADER_PROBE
- HTTPPROXY_PROBE
- IMAP_PROBE
- LDAP_PROBE
- MAIL_PROBE
- POP3_PROBE
- PROBENOTICE_PROBE
- RTSP_PROBE

- SSL_PROBE_SCRIPT
- TFTP_PROBE

The ace_scripts.tgz zip file contains these scripts and is located at this URL:

<http://www.cisco.com/cgi-bin/tablebuild.pl/cat6500-ace>

To unzip this file, use the **gunzip** command in Exec mode, as described in the “[Unzipping and Untarring ACE Sample Scripts](#)” section.

Probe Script Quick Start

Before you can run a probe script, you must copy the script onto the ACE, configure a script probe, and then associate the script with the probe. [Table A-1](#) provides steps to copy and load a script on the ACE, and configure an associated scripted probe.

Table A-1 Probe Script Quick Start

Task and Command Example

1. Copy the script into the disk0: directory on the ACE. For example, to copy a script from an FTP server to the ACE and rename it to ACETCL, enter:

```
host1/Admin# copy ftp://192.168.1.1/test1/ECHO_PROBE_SCRIPT
disk0:ACETCL
Enter username:
```

At the prompt, you must provide a username for the server.

Note The filename that you assign the script must be unique across the contexts. You will use this filename when you load the script into the ACE memory and configure the probe.

2. If you are operating in multiple contexts, observe the CLI prompt to verify that you are operating in the desired context. If necessary, change to, or directly log in to, the correct context.

```
host1/Admin# changeto C1
host1/C1#
```

The rest of the examples in this table use the Admin context for illustration purposes, unless otherwise specified. For details on creating contexts, refer to the *Cisco Application Control Engine Module Administration Guide*.

Table A-1 Probe Script Quick Start (continued)**Task and Command Example**

3. Enter configuration mode by entering **config**.

```
host1/Admin# config
Enter configuration commands, one per line. End with CNTL/Z
host1/Admin(config)#
```

4. Load the script into the ACE memory:

```
host1/Admin(config)# script file 22 ACETCL
```

5. Create a scripted probe:

```
host1/Admin(config)# probe scripted test1
host1/Admin(config-probe-scripted)# interval 10
host1/Admin(config-probe-scripted)# script ACETCL
host1/Admin(config-probe-scripted)# exit
```

6. Create a real server and a server farm. Then associate the probe and real server to the server farm:

```
host1/Admin(config)# rserver host test
host1/Admin(config-rserver-host)# ip address 10.1.0.105
host1/Admin(config-rserver-host)# inservice
host1/Admin(config-rserver-host)# exit
host1/Admin(config)# serverfarm host tests
host1/Admin(config-sfarm-host)# probe test1
host1/Admin(config-sfarm-host)# rserver test
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# Ctrl-z
```

7. At this point, the script probe should be running. You can use the **show probe detail** command in Exec mode to ensure that the script is running.

8. To stop the script probe:

```
host1/Admin# config
host1/Admin(config)# serverfarm host test
host1/Admin(config-sfarm-host)# no probe test1
host1/Admin(config-sfarm-host)# exit
```

Copying and Loading Scripts on the ACE

You load scripts onto the ACE through script files. A script file contains only one script. The ACE allows the configuration of 256 script files.

When using scripts on the ACE, the following considerations apply:

- Each script is always identified by its unique name as defined when copying the script file into the ACE disk0: filesystem. The script name must be unique across contexts.
- During probe configuration, you can assign a script to a probe. If the script is unavailable at that time, the probe attempts to execute the script and returns an error code. If this situation occurs, a syslog message displays to indicate the probe failure and why the probe failed. If the script is unavailable due to an error when loading the script, a syslog message would indicate the script load failure. You can also use the `show script` command to display the exit codes. For a list of exit codes, see [Table A-7](#).
- To change a script that is already loaded into memory, you must unload and then reload the script. For information on loading a script file, see the [“Loading Scripts into the ACE Memory”](#) section. For information on reloading a script, see the [“Removing Scripts from ACE Memory”](#) section.

After the script is changed in memory, the ACE applies the changes automatically the next time that the script executes. The command line arguments specified during probe configuration still apply after the reloading of the script.

**Note**

Because the ACE does not replicate probe scripts to the standby in a redundant configuration, you must copy the scripts from the active ACE to the standby ACE. Otherwise, configuration synchronization does not work properly.

This section contains the following topics:

- [Copying Scripts to the ACE](#)
- [Unzipping and Untarring ACE Sample Scripts](#)
- [Loading Scripts into the ACE Memory](#)
- [Removing Scripts from ACE Memory](#)
- [Reloading Modified Scripts in ACE Memory](#)

Copying Scripts to the ACE

To copy a script from a server to the ACE disk0: filesystem, use the **copy** command in Exec mode. Note that you can also copy the file from the supervisor to the ACE.

Because of virtualization, by default, a script file is copied into the directory for context that you are currently accessing. Thus, a script file in one context cannot be seen from another context. For details about virtualization, refer to the *Cisco Application Control Engine Module Virtualization Configuration Guide*. The syntax of this command is:

```
copy [ftp://server/path | tftp://server[:port]/path |
      sftp://[username@]server/path] disk0:filename
```

The keywords and arguments are:

- **ftp://server/path**—Specifies the File Transfer Protocol (FTP) network server and the source location of the script file including its filename.
- **sftp://[username@]server[/path]**—Specifies the Secure Trivial File Transfer Protocol (SFTP) network server and the source location of the script file including its filename.
- **tftp://server[:port]/path**—Specifies the Trivial File Transfer Protocol (TFTP) network server and the source location of the script file including its filename.
- **disk0:filename**—Specifies the destination filename for the script on the ACE disk0: filesystem. If you do not enter a filename, you are prompted to enter a filename or accept the source filename. You will use this filename when you load the script into the ACE memory and configure the probe.



Note The filename that you assign to the script must be unique across the contexts.

For example, to copy a script from an FTP server to the ACE, enter:

```
host1/Admin# copy ftp://192.168.1.1/test1/FTP_PROBE_SCRIPT
disk0:ftp1.tcl
Enter username:
```

At the prompt, you must provide a username for the server.

You can also copy the zipped sample scripts file for the ACE onto disk0:. After you copy the zipped file, use the **gunzip** command in Exec mode to unzip its contents. For information on using this command, see the “[Unzipping and Untarring ACE Sample Scripts](#)” section.

Unzipping and Untarring ACE Sample Scripts

Sample scripts for the ACE are available to support the TCL feature. All of these scripts are provided in a zipped file which contains a .tar file. After you copy the zip file to the ACE, you need to unzip it and then untar it. When you untar the file, the ACE automatically creates an ace_scripts directory and places all of the individual scripts in it.

To unzip the sample scripts file, use the **gunzip** command in Exec mode. The syntax for this command is:

```
gunzip disk0:[path/]filename.tgz
```

The *filename* argument is the name of the zipped scripts file.

For example, to unzip the ace_scripts.tgz scripts file, enter:

```
host1/Admin# gunzip disk0:ace_scripts.tgz
```

The ACE unzips the file and places the ace_scripts.tar file in the disk0: filesystem.

To untar all of the script files from the ace_scripts.tar file, use the **untar** command in Exec mode. The syntax for the command is:

```
untar disk0:[path/]filename
```

The *filename* argument identifies the name of the .tar file in the disk0: filesystem. The filename must end with a .tar extension.

For example, to untar all of the script files into the ace_scripts directory in the disk0: filesystem, enter:

```
host1/Admin# untar disk0:ace_scripts.tar
```

To view the scripts in the ace_scripts directory, use the **dir** command in Exec mode. For example, enter:

```
host1/Admin# dir disk0:ace_scripts/
```

Before you can load a sample script into memory, you must copy the script out of the `ace_scripts` directory into the `disk0:` directory. Use the **copy disk0:** command. For example, to copy the `ftp1.tcl` script from the `ace_scripts` directory to the `disk0:` directory, enter:

```
host1/Admin# copy disk0:ace_scripts/ftp1.tcl disk0:ftp1.tcl
```

Loading Scripts into the ACE Memory

To load the script into memory on the ACE and enable it for use, use the **script file** command in configuration mode. The syntax of this command is:

```
script file index script_name
```

The arguments are:

- *index*—An index number for the script file. The index number will be associated with the script name. The number must be unique across the contexts. Enter a number from 1 to 255.
- *script_name*—The name of the script on the `disk0:` filesystem.



Note

To load a script into memory, the script must be in the `disk0:` directory. The ACE does not load script files in a `disk0:` subdirectory.

For example, to load a script into memory:

```
host1/Admin(config)# script file 22 ftp1.tcl
```

To run the script or create a health probe using that script, use the script name you configured; not the script file from which the script was loaded.

Removing Scripts from ACE Memory

After a script file has been loaded, the scripts in that file exist in the ACE independent of the file from which that script was loaded. To remove a script from memory and the running configuration, use the **no script file** command in configuration mode. The syntax of this command is:

```
no script file index
```

The *index* argument is an index number for the script file you want to remove from memory.

For example, to remove the script with index 22, enter:

```
host1/Admin(config)# no script file 22
```

Reloading Modified Scripts in ACE Memory

If a script file is subsequently modified, you can update the script in memory by reloading it. Reloading a script requires:

1. The removal of the script from memory by using the **no script file** command in configuration mode. For information on removing a script from memory, see the [“Removing Scripts from ACE Memory”](#) section.
2. The reloading of the modified script into memory by using the **script file** command in configuration mode. For information on loading a script into memory, see the [“Loading Scripts into the ACE Memory”](#) section.

After the script is reloaded into memory, the ACE applies the changes automatically in the next script execution. The command line arguments specified during probe configuration still apply after the reloading of the script.

For example, to reload a script 22, enter:

```
host1/Admin(config)# no script file 22
host1/Admin(config)# script file 22 ftp1.tcl
```

Configuring Health Probes for Scripts

You can create a scripted probe that the ACE periodically executes for each real server in any server farm associated with a probe. Depending upon the exit code of a script, the real server is considered passed or failed. For more information on exit codes, see the [“Exit Codes”](#) section.

To create a scripted probe, use the **probe scripted** *probe_name* command in configuration mode. This command enters a probe configuration mode that is similar to the existing ACE health probe modes (such as HTTP, TCP, DNS, SMTP, and so on).

The probe scripted configuration mode includes the **faildetect**, **interval**, **passdetect**, **open**, **priority**, and **receive** commands. The **script** *script_name* command can process up to 80 arguments that are passed to the script when it is run as part of the health probe function. When you configure each interval of time, an internal ACE scheduler schedules the health scripts. For more information on configuring scripted probes and the associated commands, see [Chapter 4, Configuring Health Monitoring](#).

After creating the scripted health probe, attach the probe to the server farm and the virtual server. For example:

```
host1/Admin(config)# serverfarm host tests
host1/Admin(config-sfarm-host)# probe test1
host1/Admin(config-sfarm-host)# rserver test
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
```

Writing Probe Scripts

Probe scripts test the health of a real server by creating a network connection to the server, sending data to the server, and checking the response. The flexibility of this TCL scripting environment makes the available probing functions possible.

Write the script as if you intend to perform only one probe. You must declare the result of the probe using the **exit** command. Depending upon the exit code of a script, the real server is considered passed or failed. For more information on exit codes, see the “[Exit Codes](#)” section.

A health script typically performs these actions:

- Opens a socket to an IP address.
- Sends one or more requests.
- Reads the responses.
- Analyzes the responses.
- Closes the socket.
- Exits the script by using an exit code for success or failure.

This section provides information to assist you when you write a probe script. The topics include:

- [TCL Script Commands Supported on the ACE](#)
- [Environment Variables](#)
- [Exit Codes](#)
- [Example for Writing a Probe Script](#)

TCL Script Commands Supported on the ACE

The ACE TCL script feature is based on the TCL 8.44 source distribution software. [Table A-2](#) lists the TCL commands that are supported by ACE.

Table A-2 *TCL Commands Supported by the ACE*

Command			
Generic TCL Commands¹			
append	array	binary	break
case	catch	concat	continue
encoding	error	eval	exit
expr	fblocked	for	foreach
format	gets	glob	global
if	incr	info	join
lappend	lindex	linsert	list
llength	lrange	lreplace	lsearch
lset	lsort	namespace	proc
regexp	regsub	rename	return
scan	set	split	string
subst	switch	unset	uplevel
upvar	variable	while	
Time-Related Commands			
after	clock	time	

Table A-2 *TCL Commands Supported by the ACE (continued)*

Command			
Socket Commands			
close	eof	fconfigure	fileevent
flush	read	socket	update
vwait			

1. The **puts** command can appear in a script, however, the ACE does not display its output.

[Table A-3](#) lists the TCL command not supported by the ACE.

Table A-3 *TCL Commands Not Supported by the ACE*

Generic TCL Commands			
auto_execok	auto_import	auto_load	auto_load_index
auto_qualify	cd	exec	file
fcopy	history	interp	load
open	package	pid	pwd
seek	source	tell	trace

Table A-4 lists the TCL command specific to the ACE.

Table A-4 ACE Specific TCL Commands

Command	Definition
<code>gset varname value</code>	<p>Allows you to preserve the state of a probe by setting a variable that is global to all probe threads running from the same script.</p> <p>Variables in a probe script are only visible within one probe thread. Each time a probe exits, all variables are gone. For example, if a probe script contains a 'gset x 1 ; incr x', variable x would increase by 1 for each probe attempt.</p> <ul style="list-style-type: none"> • To set the value of variable from script, set <i>var</i> or <i>\$var</i>. • To reset the value of variable from script, unset <i>var</i>. • To display the gset value, have the script place this value in the exit message . Then you can use the show script command to display the exit message. Make sure that the script does not overwrite the exit message with another value before it exits. For information on the show script command, see the “Displaying the Statistics for an Active Script” section.

The UDP command set allows Scotty-based TCL scripts to run on the ACE. Scotty is the name of a software package that allows you to implement site-specific network management software using high-level, string-based APIs. The TCL UDP command reference is located at this URL:

<http://wwwhome.cs.utwente.nl/~schoenw/scotty/>

Table A-5 lists the UDP commands used by the ACE.

Table A-5 UDP Commands

Command	Definition
udp_binary send <i>handle</i> [<i>host port</i>] <i>message</i>	Sends binary data containing a message to the destination specified by host and port. The <i>host</i> and <i>port</i> arguments may not be used if the UDP handle is already connected to a transport endpoint. If the UDP handle is not connected, you must use these optional arguments to specify the destination of the datagram.
udp bind <i>handle</i> readable [<i>script</i>] udp bind <i>handle</i> writable [<i>script</i>]	Allows binding scripts to a UDP handle. A script is evaluated once the UDP handle becomes either readable or writable, depending on the third argument of the udp bind command. The script currently bound to a UDP handle can be retrieved by calling the udp bind command without a <i>script</i> argument. Bindings are removed by binding an empty string.
udp close <i>handle</i>	Closes the UDP socket associated with handle.
udp connect <i>host port</i>	Opens a UDP datagram socket and connects it to a port on a remote host. A connected UDP socket only allows sending messages to a single destination. This usually allows shortening the code because there is no need to specify the destination address for each udp send command on a connected UDP socket. The command returns a UDP handle.
udp info [<i>handle</i>]	Without the <i>handle</i> argument, this command returns a list of all existing UDP handles. Information about the state of a UDP handle can be obtained by supplying a valid UDP handle. The result is a list containing the source IP address, the source port, the destination IP address and the destination port.
udp open [<i>port</i>]	Opens a UDP datagram socket and returns a UDP handle. The socket is bound to given port number or name. An unused port number is used if the <i>port</i> argument is missing.

Table A-5 UDP Commands (continued)

Command	Definition
udp receive <i>handle</i>	Receives a datagram from the UDP socket associated with the handle. This command blocks until a datagram is ready to be received.
udp send <i>handle [host port] message</i>	Sends ASCII data containing a message to the destination specified by host and port. The <i>host</i> and <i>port</i> arguments may not be used if the UDP handle is already connected to a transport endpoint. If the UDP handle is not connected, you must use these optional arguments to specify the destination of the datagram.

Environment Variables

Health probe scripts have access to many configured items through a predefined TCL array. The most common use of this array is to find the current real server IP addresses of the suspect during any particular launch of the script.

Whenever the ACE executes a script probe, a special array called `scriptprobe_env` is passed to the script. This array holds important parameters that may be used by the script.

[Table A-6](#) lists the members of the `scriptprobe_env` array.

Table A-6 Member list for the `scriptprobe_env` Array

Member name	Content
<code>realIP</code>	Suspect IP address
<code>realPort</code>	Suspect IP port
<code>intervalTimeout</code>	Configured probe interval in seconds
<code>openTimeout</code>	Configured socket open timeout for this probe (tbd)
<code>recvTimeout</code>	Configured socket receive timeout for this probe
<code>failedTimeout</code>	Configure failed timeout
<code>retries</code>	Configured retry count

Table A-6 Member list for the *scriptprobe_env* Array (continued)

Member name	Content
healthStatus	Current suspect health status
contextID	The ID for the context running this script
failedRetries	Consecutive successful retries on a failed server before marking it as passed
isRouted	Boolean to determine if this IP address is a routed address
pid	Process identifier of the TCL process
runID	Pointer to the event structure (em_event_t)

Exit Codes

The probe script uses exit codes to signify various internal conditions. The exit code information can help you troubleshoot your scripts if they do not operate correctly. A probe script indicates the relative health and availability of a real server using the exit code of the script. By calling exit 30001, a script indicates that the server successfully responded to the probe. Calling exit 30002 indicates that the server did not respond correctly to the health probe.

For example, if a probe script fails and exits with 30002, the corresponding server is marked as `PROBE_FAILED` and is temporarily disabled from the server farm. The ACE continues to probe the server. When the probe successfully reconnects and exits with 30001, the ACE marks the server status as `OPERATIONAL` and enables the server from the server farm again. The exit 30001 must occur the number of failedRetries times before the server is marked as `OPERATIONAL`. See the previous section on environmental variables for further information on the failedRetries member name.

These situations can cause a script to fail and mark the suspect PROBE_FAILED:

- **TCL errors**—Occurs when scripts contain errors that are caught by the TCL interpreter, for example, a syntax error.

The syntax error message is stored in the special variable **erroInfo** and can be viewed using the **show script** command in Exec mode. Another example of TCL errors would cause the TCL interpreter to abort and call panic.

- **A stopped script**—Caused by an infinite loop and wait indefinitely for a response. Each script must complete its task within the configured time interval. If the script does not complete its task, the script controller terminates the script, and the suspect is failed implicitly.
- **Error conditions**—Occurs when a connection timeout or a peer-refused connection is also treated as an implicit failure.

Table A-7 shows all exit codes used in the ACE.

Table A-7 ACE Exit Codes

Exit Code	Description/Message
30001	Probe successful (no message)
30002	Probe error: Server did not respond as expected
30003	Internal error: Fork failed for TCL script
30004	Internal error: Script probe terminated due to timeout
30005	Internal error: TCL interpreter PANIC (interpreter problem)
30006	Internal error: Script error
30007	Internal error: Script-file lookup failed or empty buffer
30008	Internal error: Failed to allocate memory for TCL_wt (worker thread) qnode
30009	Internal error: Unknown script error
30010	Internal error: Out of sockets for the TCL script
30011	Internal error: Unable to read persistent variable table
30012	Internal error: PData (probe data) pointer is null

Example for Writing a Probe Script

This example shows how a script is written to probe an HTTP server using a health script:

```
# get the IP address of the real server from a predefined global array
# scriptprobe_env
set ip $scriptprobe_env(realIP)
set port 80
set url "GET /index.html HTTP/1.0\n\n"

# Open a socket to the server. This creates a TCP connection to the
# real server
set exit_msg "opening socket"
set sock [socket $ip $port]
fconfigure $sock -buffering none -eofchar {}

# Wait for the response from the server and read that in variable line
set exit_msg "receiving response"
set line [ read $sock ]

# Parse the response
if { [ regexp "HTTP/1.. ([0-9\+]) " $line match status ] }

# Close the socket. Application MUST close the socket once the
# request/response is over.
# This allows other applications and tcl scripts to make
# a good use of socket resource. Health monitoring is allowed to open
# only 200 sockets simultaneously.
set exit_msg "closing socket"
close $sock

# decide the exit code to return to control module.
# If the status code is OK then script MUST do exit 30001
# to signal successful completion of a script probe.
# In this example any other status code means failure.
# User must do exit 30002 when a probe has failed.
if { $status == 200 } {
    set exit_msg "probe success"
    exit 30001
} else {
    set exit_msg "probe fail : can't find status code"
    exit 30002
}
```

Displaying Script Information

The following sections provide information for displaying scripted probe and script information:

- [Displaying ACE Script and Scripted Probe Configuration](#)
- [Displaying Scripted Probe Information](#)
- [Displaying Global Scripted Probe Statistics](#)
- [Displaying the Statistics for an Active Script](#)
- [Displaying the Script Contents](#)

Displaying ACE Script and Scripted Probe Configuration

To display configuration information for scripts and scripted probes on the ACE, use the **show running-config** command in Exec mode. For example, enter:

```
switch/Admin# show running-config
Generating configuration....

logging enable
boot system image:

script file 200 ftp1.tcl

probe scripted bts_test_probe
  interval 10
  receive 8
  script test.tcl

rserver host bts_test_rserver
  ip address 10.86.209.1
  probe bts_test_probe
  inservice

snmp-server user www Network-Monitor
snmp-server user admin Network-Monitor

context Admin
...
```

Displaying Scripted Probe Information

To display configuration and probe result information about scripted probes, use the **show probe** command in Exec mode. The syntax of this command is:

```
show probe scripted_probe_name [detail]
```

The argument and option are:

- *scripted_probe_name*—Displays the information for the specified scripted probe name
- **detail**—(Optional) Displays detailed probe information including configuration information and statistics

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

For example, to view the SCRIPT-1 scripted probe and detailed information, enter:

```
host1/Admin# show probe SCRIPT-1 detail
```

[Table A-8](#) describes the fields in the **show probe** command output for a scripted probe.

Table A-8 *Field Descriptions for the show probe Command for a Scripted Probe*

Field	Description
probe	Name of the probe.
type	Probe type.
description	Configured string that describes the probe.
port	Port number that the probe uses. By default, the probe uses the port number based on its type.
address	Not used for scripted probes.
addr type	Not used for scripted probes.
interval	Time interval in seconds that the ACE sends probes to a server marked as passed.

Table A-8 *Field Descriptions for the show probe Command for a Scripted Probe (continued)*

Field	Description
pass intvl	Time period in seconds to send a probe to a failed server.
pass count	Number of successful probe replies before enabling failed server.
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.
script filename	Script filename.
probe association	Serverfarm or real server association.
probe results	
probed-address	Destination or source address for the probe.
probes	Total number of probes.
failed	Total number of failed probes.
passed	Total number of passed probes.
health	Current health of probe: PASSED or FAILED.
Additional Detailed Output:	
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.
Last status code	Last exit code (see Table A-7).
Last disconnect err	Message for the exit code (see Table A-7).
Last probe time	Timestamp for the last probe.
Last fail time	Timestamp for the last failed probe.
Last active time	Timestamp for the last active time.
Internal error	Counter for the number of internal errors encountered.

Displaying Global Scripted Probe Statistics

To display the global statistics for all scripted probe, use the **show stats probe type scripted** command in Exec mode. For example, enter:

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

Table A-9 describes the fields in the **show stats probe type scripted** command output.

Table A-9 *Field Descriptions for the show stats probe type scripted command*

Field	Description
Total probes sent	Total number of probes sent by all scripted probes.
Total send failures	Total number of send failures for all scripted probes. These failures are due to internal errors. For more information, see the last disconnect error field displayed by the show probe command.
Total probes passed	Total number of passed probes for all scripted probes.
Total probes failed	Total number of failed probes for all scripted probes.
Total connect errors	Total number of connection errors for all scripted probes.
Total conns refused	Total number of connections refused for all scripted probes.
Total RST received	Total number of resets received by all scripted probes.
Total open timeouts	Total number of open timeouts for all scripted probes.
Total receive timeouts	Total number of time outs received by all scripted probes.

Displaying the Statistics for an Active Script

To display the statistics for a script file active on the ACE including exit codes and exit messages, use the **show script** command from Exec mode. The syntax for this command is:

```
show script script_name probe_name [rserver_name [port_name]]
           [serverfarm sfarm_name]
```

The arguments and options are:

- *script_name*—The name of the script
- *probe_name*—The name of the scripted probe associated the script
- *rserver_name* [*port_name*]—(Optional) Specifies the name of the real server and optional port number using the scripted probe
- **serverfarm** *sfarm_name*—The name of the server farm using the scripted probe

For example, to display the statistics for the ECHO_PROBE_SCRIPT script for the SCRIPT1 probe:

```
host1/Admin# show script ECHO_PROBE_SCRIPT SCRIPT1
```

[Table A-10](#) describes the fields in the **show script** command output.

Table A-10 Field Descriptions for the show script command

Field	Description
Script	Name of the script.
Scripted probe	Probe associated with the script.
Probe-association(s): (count= <i>number</i>)	Total number of real servers and server farms associated with the scripted probe.
Rserver/Serverfarm	Name of the real server or server farm for the script statistics.
Exit code	Current exit code for the script. See Table A-7 for information on exit codes.
Child PID	Child process identifier for the script.
Exit message	Value of the TCL gset variable from the script.

Table A-10 Field Descriptions for the show script command (continued)

Field	Description
Panic string	Indicates the internal problem with the TCL interpreter.
Internal error	Error code message associated with the exit code.
Last RunStart count/Last RunStart time	Number of times that the script successfully started and the last time stamp.
Last RunEnd count/Last RunEnd time	Number of times that the script successfully ended and the last time stamp.
Last Probe OK count/ Last Probe OK time	Number of times that the scripted probe passed and the last time stamp.
Last ProbeFail count/ Last ProbeFail time	Number of times that the scripted probe failed and the last time stamp.
Last ForkFail count/ Last ForkFail time	Number of times that the fork failed and the last time stamp.
Last Kill count/ Last Kill time	Number of times that the script probe failed due to timeout and the last time stamp.
Last Panic count/ Last Panic time	Number of times that there was an internal problem with the TCL interpreter and the last time stamp.
Last Nodecreate Error count/Last Nodecreate Error time	Number of times that the ACE ran out of memory for the TCL worker thread node and the last time stamp.
Last Unassociated Script count/ Last Unassociated Script time	Number of times that the script in memory could not be associated and the last time stamp.
Last Fail Internal/ Last Fail Internal	Number of times that the script had an internal syntax error and the last time stamp.
Last Socket-Limit count/ Last Socket-Limit time	Number of times that the ACE ran out of sockets for the TCL script and the last time stamp.
Last PV-Read count/ Last PV-Read Time	Number of times that the persistent variable table was read and the last time stamp.

Table A-10 Field Descriptions for the `show script` command (continued)

Field	Description
Last PData-Null count/Last PData-Null time	Number of times that the probe data pointer is null and the last time stamp.
Last Unknown count/ Last Unknown time	Number of times that the script passed an error code that is not recognized and last time stamp.

Displaying the Script Contents

To display the contents for a script file loaded on the ACE, use the **show script code** command from Exec mode. The syntax for this command is:

```
show script code script_name
```

The arguments and options are:

- **code**—Displays the code within the script file
- *script_name*—The name of the script

For example, to display the code within the ECHO_PROBE_SCRIPT script:

```
host1/Admin# show script code ECHO_PROBE_SCRIPT
```

Debugging Probe Scripts

To debug a script probe, you can do the following:

- Use the `EXIT_MSG` variable in the script. Each probe suspect contains its own `EXIT_MSG` variable. This variable allows you to trace the status of a script and check the status of the probe.

This example shows how to use the `EXIT_MSG` variable in a script:

```
set EXIT_MSG "before opening socket"
set s [ socket $ip $port]
set EXIT_MSG " before receive string"
gets $s
set EXIT_MSG "before close socket"
close $s
```

If a probe suspect fails when receiving the message, you should see `EXIT_MSG = before you receive the string`. You can view the `EXIT_MSG` variable with the **show script** command in Exec mode.

- Use the **show probe** command in Exec mode to view the current active probe suspects in the system. For more information on this command, see the [“Displaying Script Information”](#) section.
- Use the **show script** command in Exec mode to view:
 - The last exit status with the exit code number.
 - The internal error that is generated by the TCL compiler. When the script has a TCL runtime error, the TCL interpreter stops running the script and the ACE displays this error.
 - The `EXIT_MSG` variable value of the TCL **gset** command from the script.