



CHAPTER 2

Configuring Real Servers and Server Farms

This chapter describes the functions of real servers and server farms in load balancing and how to configure them on the ACE appliance. It contains the following major sections:

- [Configuring Real Servers](#)
- [Configuring a Server Farm](#)
- [Displaying Real Server Configurations and Statistics](#)
- [Clearing Real Server Statistics and Connections](#)
- [Displaying Server Farm Configurations and Statistics](#)
- [Clearing Server Farm Statistics](#)
- [Where to Go Next](#)

This chapter also provides information on the Asymmetric Server Normalization feature, as described in the “[Configuring Asymmetric Server Normalization](#)” section.

Configuring Real Servers

This section describes real servers and how to configure them. It contains the following topics:

- [Real Server Overview](#)
- [Managing Real Servers](#)
- [Real Server Configuration Quick Start](#)
- [Creating a Real Server](#)
- [Configuring a Real Server Description](#)
- [Configuring a Real Server IP Address](#)
- [Configuring Real Server Health Monitoring](#)
- [Configuring AND Logic for Real Server Probes](#)
- [Configuring Real Server Connection Limits](#)
- [Configuring Real Server Rate Limiting](#)
- [Configuring a Real Server Relocation String](#)
- [Configuring a Real Server Weight](#)
- [Placing a Real Server in Service](#)
- [Gracefully Shutting Down a Server](#)
- [Examples of Real Server Configurations](#)

Real Server Overview

Real servers are dedicated physical servers that you typically configure in groups called server farms. These servers provide services to clients, such as HTTP or XML content, streaming media (video or audio), TFTP or FTP uploads and downloads, and so on. You identify real servers with names and characterize them with IP addresses, connection limits, and weight values.

The ACE uses traffic classification maps (class maps) within policy maps to filter out interesting traffic and to apply specific actions to that traffic based on the SLB configuration. You use class maps to configure a virtual server address and definition. The load-balancing predictor algorithms (for example, round-robin, least connections, and so on) determine the servers to which the ACE sends connection requests. For information about configuring traffic policies for SLB, see [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).

Real Server Configuration Quick Start

[Table 2-1](#) provides a quick overview of the steps required to configure real servers. Each step includes the CLI command or a reference to the procedure required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following [Table 2-1](#).

Table 2-1 Real Server Configuration Quick Start

Task and Command Example

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

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

The rest of the examples in this table use the Admin context, unless otherwise specified. For details about creating contexts, see the *Cisco 4700 Series Application Control Engine Appliance Virtualization Configuration Guide*.

-
2. Enter configuration mode.

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

-
3. Configure a real server and enter real server configuration mode.

```
host1/Admin(config)# rserver SERVER1  
host1/Admin(config-rserver-host)#
```

Table 2-1 Real Server Configuration Quick Start (continued)

Task and Command Example	
4. (Recommended) Enter a description of the real server.	<pre>host1/Admin(config-rserver-host)# description accounting department server</pre>
5. Configure an IP address for the real server in dotted-decimal notation.	<pre>host1/Admin(config-rserver-host)# ip address 192.168.12.15</pre>
6. Assign one or more existing probes to the real server for health monitoring.	<pre>host1/Admin(config-rserver-host)# probe PROBE1</pre>
7. To prevent the real server from becoming overloaded, configure connection limits.	<pre>host1/Admin(config-rserver-host)# conn-limit max 20000000 min 15000000</pre>
8. If you plan to use the weighted round-robin or least connections predictor method, configure a weight for the real server.	<pre>host1/Admin(config-rserver-host)# weight 50</pre>
9. Place the real server in service.	<pre>host1/Admin(config-rserver-host)# inservice host1/Admin(config-rserver-host)# Ctrl-Z</pre>
10. Use the following command to display the real server configuration. Make any modifications to your configuration as necessary, then reenter the command to verify your configuration changes.	<pre>host1/Admin# show running-config rserver</pre>
11. (Optional) Save your configuration changes to flash memory.	<pre>host1/Admin# copy running-config startup-config</pre>

Creating a Real Server

You can configure a real server and enter real server configuration mode by using the **rserver** command in configuration mode. You can create a maximum of 4095 real servers. The syntax of this command is as follows:

```
rserver [host | redirect] name
```

The keywords, arguments, and options for this command are as follows:

- **host**—(Optional, default) Specifies a typical real server that provides content and services to clients.
- **redirect**—(Optional) Specifies a real server used to redirect traffic to a new location as specified in the *relocn-string* argument of the **webhost-redirect** command. See the “[Configuring a Real Server Relocation String](#)” section.
- *name*—Identifier for the real server. Enter an unquoted text string with no spaces and maximum of 64 alphanumeric characters.

**Note**

You can associate a real sever of type **host** only with a server farm of type **host**. You can associate a real server of type **redirect** only with a server farm of type **redirect**.

For example, to create a real server of type host, enter:

```
host1/Admin(config) # rserver server1
host1/Admin(config-rserver-host) #
```

To remove the real server of type host from the configuration, enter:

```
host1/Admin(config) # no rserver server1
```

To create a real server of type redirect, enter:

```
host1/Admin(config) # rserver redirect server2
host1/Admin(config-rserver-redir) #
```

To remove the real server of type redirect from the configuration, enter:

```
host1/Admin(config) # no rserver redirect server2
```

**Note**

The sections that follow apply to both real server types unless otherwise indicated.

Configuring a Real Server Description

You can configure a description for a real server by using the **description** command in either real server host, or real server redirect configuration mode. The syntax of this command is as follows:

description *string*

For the *string* argument, enter an alphanumeric text string and a maximum of 240 characters, including quotation marks (“ ”) and spaces.

For example, enter:

```
host1/Admin(config)# rserver server1  
host1/Admin(config-rserver-host)# description accounting server
```

To remove the real-server description from the configuration, enter:

```
host1/Admin(config-rserver-host)# no description
```

Configuring a Real Server IP Address

You can configure an IP address so that the ACE can access a real server of type **host**. You can configure an IP address by using the **ip address** command in real server host configuration mode. The syntax of this command is as follows:

ip address *ip_address*

For the *ip_address* argument, enter an IPv4 address in dotted-decimal notation (for example, 192.168.12.15). The IP address must be unique within the current context.

For example, to specify an address enter:

```
host1/Admin(config)# rserver server1  
host1/Admin(config-rserver-host)# ip address 192.168.12.15
```

To remove the real server IP address from the configuration, enter:

```
host1/Admin(config-rserver-host)# no ip address
```

Configuring Real Server Health Monitoring

To check the health and availability of a real server, the ACE periodically sends a probe to the real server. Depending on the server response, the ACE determines whether to include the server in its load-balancing decision. For more information about probes, see [Chapter 4, Configuring Health Monitoring](#).

You can assign one or more existing probes to a real server by using the **probe** command in real server host configuration mode. This command applies only to real servers of type **host**. The syntax of this command is as follows:

probe *name*

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

For example, enter:

```
host1/Admin(config)# rserver server1
host1/Admin(config-rserver-host)# probe probe1
```

To remove a real server probe from the configuration, enter:

```
host1/Admin(config-rserver-host)# no probe probe1
```

Configuring AND Logic for Real Server Probes

By default, real servers with multiple probes configured on them have an OR logic associated with them. This means that if one of the real server probes fails, the real server fails and enters the PROBE-FAILED state. To configure a real server to remain in the OPERATIONAL state unless all probes associated with it fail (AND logic), use the **fail-on-all** command in real server host configuration mode. This command is applicable to all probe types. For more information about probes, see [Chapter 4, Configuring Health Monitoring](#).



Note

You can configure the **fail-on-all** command also on server farms and real servers in server farms. See the [“Configuring AND Logic for Server Farm Probes”](#) section and the [“Configuring AND Logic for Real Server Probes in a Server Farm”](#) section.

The syntax of this command is as follows:

fail-on-all

For example, to configure the SERVER1 real server to remain in the OPERATIONAL state unless all associated probes fail, enter:

```
host1/Admin(config)# rserver SERVER1
host1/Admin(config-rserver-host)# ip address 192.168.12.15
host1/Admin(config-rserver-host)# probe HTTP_PROBE
host1/Admin(config-rserver-host)# probe ICMP_PROBE
host1/Admin(config-rserver-host)# fail-on-all
```

In this example, if HTTP_PROBE fails, the SERVER1 real server remains in the OPERATIONAL state. If both probes fail, the real server fails and enters the PROBE-FAILED state.

To remove the AND probe logic from the real server and return the behavior to the default of OR logic, enter:

```
host1/Admin(config-rserver-host)# no fail-on-all
```

Configuring Real Server Connection Limits

To prevent a real server from being overburdened and to conserve system resources, you can limit the maximum number of active connections to the server. You can set the maximum and minimum connection thresholds by using the **conn-limit** command in either real server host or real server redirect configuration mode. The syntax of this command is as follows:

```
conn-limit max maxconns min minconns
```

The keywords and arguments are as follows:

- **max** *maxconns*—Specifies the maximum allowable number of active connections to a real server. When the number of connections exceeds the *maxconns* threshold value, the ACE stops sending connections to the real server and assigns the real server a state of MAXCONNS until the number of connections falls below the configured *minconns* value. Enter an integer from 1 to 4000000. The default is 4000000.

- **min minconns**—Specifies the minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold. Enter an integer from 1 to 4000000. The default is 4000000. The *minconns* value must be less than or equal to the *maxconns* value.

Because the ACE has two network processors (NPs), the *maxconns* value for a real server is divided as equally as possible between the two NPs. If you configure an odd value for *maxconns*, one of the NPs will have a *maxconns* value that is one more than the other. With very small values for *maxconns*, a connection may be denied even though one of the NPs has the capacity to handle the connection.

Consider the scenario where you configure a value of 3 for the *maxconns* argument. One NP will have a value of 2 and the other NP will have a value of 1. If both NPs have reached their connection limits for that server, the next connection request for that server will be denied and the Out-of-rotation Count field of the **show serverfarm detail** command will increment by 1. Now, suppose that a connection is closed on the NP with the *maxconns* value of 2. If the next connection request to the ACE hits the NP with the *maxconns* value of 1 and it has already reached its limit, the ACE denies the connection, even though the other NP has the capacity to service the connection.

If you change the *minconns* value while there are live connections to a real server, the server could enter the MAXCONNS state without actually achieving the *maxconns* value in terms of the number of connections to it. Consider the following scenario where you configure a *maxconns* value of 10 and a *minconns* value of 5. Again, the ACE divides the values as equally as possible between the two NPs. In this case, both NPs would have a *maxconns* value of 5, while NP1 would have a *minconns* value of 3 and NP2 would have a *minconns* value of 2.

Suppose that the real server now has 10 live connections to it. Both NPs and the server have reached the MAXCONNS state. If four connections to the real server are closed leaving six live connections, both NPs (and, hence, the real server) would still be in the MAXCONNS state with three connections each. Remember, for an NP to come out of the MAXCONNS state, the number of connections to it must be less than the *minconns* value.

If you change the server's *minconns* value to 7, NP1 would enter the OPERATIONAL state because the number of connections (three) is one less than the *minconns* value of 4. NP2 would still be in the MAXCONNS state (*minconns* value = number of connections = 3). NP1 could process two more connections for a total of only eight connections to the real server, which is two less than the server's *maxconns* value of 10.

You can also specify minimum and maximum connections for a real server in a server farm configuration. The total of the minimum and maximum connections that you configure for a server in a server farm cannot exceed the minimum and maximum connections you set globally in real server configuration mode. For details about configuring real server maximum connections in a server farm configuration, see the [“Configuring Connection Limits for a Real Server in a Server Farm”](#) section.

For example, enter:

```
host1/Admin(config)# rserver server1
host1/Admin(config-rserver-host)# conn-limit max 5000 min 4000
```

To reset the real-server maximum connection limit and minimum connection limit to the default value of 4000000, enter:

```
host1/Admin(config-rserver-host)# no conn-limit
```

Configuring Real Server Rate Limiting

In addition to preserving system resources by limiting the total number of active connections to a real server (see the [“Configuring Real Server Connection Limits”](#) section), the ACE allows you to limit the connection rate and the bandwidth rate of a real server. The connection rate is the number of connections per second received by the ACE and applied only to the new connections destined to a real server. The bandwidth rate is the number of bytes per second applied to the network traffic exchanged between the ACE and the real server in both directions.

For a real server that is associated with more than one server farm, the ACE uses the aggregated connection rate or bandwidth rate to determine if the real server has exceeded its rate limits. If the connection rate or the bandwidth rate of incoming traffic destined for a particular server exceeds the configured rate limits of the server, the ACE blocks any further traffic destined to that real server until the connection rate or bandwidth rate drops below the configured limit. Also, the ACE removes the blocked real server from future load-balancing decisions and considers only those real servers in the server farm that have a current connection rate or bandwidth rate less than or equal to the configured limit. By default, the ACE does not limit the connection rate or the bandwidth rate of real servers.

You can also limit the connection rate and the bandwidth rate of the following:

- Real server in a server farm. For details, see the [“Configuring Connection Limits for a Real Server in a Server Farm”](#) section.

- Virtual server in a connection parameter map. For details, see the *Cisco 4700 Series Application Control Engine Appliance Security Configuration Guide*.

To limit the connection rate or the bandwidth rate of a real server, use the **rate-limit** command in real server host configuration mode. The syntax of this command is as follows:

```
rate-limit { connection number1 | bandwidth number2 }
```

The keywords and arguments are as follows:

- **connection** *number1*—Specifies the real server connection-rate limit in connections per second. Enter an integer from 2 to 350000. There is no default value.
- **bandwidth** *number2*—Specifies the real server bandwidth-rate limit in bytes per second. Enter an integer from 1 to 300000000. There is no default value.



Note

The ACE applies rate limiting values across both network processors (NPs). For example, if you configure a rate-limiting value of 500, the ACE applies a rate-limit value of 250 to each NP.

For example, to limit the connection rate of a real server at the aggregate level to 100,000 connections per second, enter:

```
host1/Admin(config)# rserver host SERVER1
host1/Admin(config-rserver-host)# rate-limit connection 100000
```

To return the behavior of the ACE to the default of not limiting the real-server connection rate, enter:

```
host1/Admin(config-rserver-host)# no rate-limit connection 100000
```

For example, to limit the real-server bandwidth rate to 5000000 bytes per second, enter:

```
host1/Admin(config)# rserver host SERVER1
host1/Admin(config-rserver-host)# rate-limit bandwidth 5000000
```

To return the behavior of the ACE to the default of not limiting the real-server bandwidth, enter:

```
host1/Admin(config-rserver-host)# no rate-limit bandwidth 5000000
```

Configuring a Real Server Relocation String

You can configure a real server to redirect client requests to a location specified by a relocation string or a port number. You can configure a relocation string for a real server that you configured as a redirection server (type **redirect**) by using the **webhost-redirect** command in real server redirect configuration mode. The syntax of this command is as follows:

```
webhost-redirect relocation_string [301 | 302]
```

The keywords and arguments are as follows:

- *relocation_string*—URL string used to redirect requests to another server. Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters. The relocation string supports the following special characters:
 - **%h**—Inserts the hostname from the request Host header
 - **%p**—Inserts the URL path string from the request



Note To insert a question mark (?) in the relocation string, press **Ctrl-v** before you type the question mark.

- [301 | 302]—Specifies the redirection status code returned to a client. The codes indicate the following:
 - **301**—The requested resource has been moved permanently. For future references to this resource, the client should use one of the returned URLs.
 - **302**—(Default) The requested resource has been found but has been moved temporarily to another location. For future references to this resource, the client should continue to use the request URI because the resource may be moved to other locations occasionally.

For more information about redirection status codes, see RFC 2616.

For example, enter:

```
host1/Admin(config)# rserver redirect SERVER1
host1/Admin(config-rserver-redir)# webhost-redirect
http://%h/redirectbusy.cgi?path=%p 302
```

Assume the following client GET request:

```
GET /helloworld.html HTTP/1.1
Host: www.cisco.com
```

The redirect response would appear as follows:

```
HTTP/1.1 302 Found
Location: http://www.cisco.com/redirectbusy.cgi?path=/helloworld.html
```

To remove the relocation string from the configuration, enter:

```
host1/Admin(config-rserver-redir) # no webhost-redirection
```

Configuring a Real Server Weight

You can configure the connection capacity of a real server of type **host** or **redirect** in relation to the other servers in a server farm by using the **weight** command in real server host configuration mode. The ACE uses the weight value that you specify for a server in the weighted round-robin and least-connections load-balancing predictors. Servers with a higher configured weight value have a higher priority with respect to connections than servers with a lower weight. For example, a server with a weight of 5 would receive five connections for every one connection for a server with a weight of 1.



Note

For the least-connections predictor, server weights take effect only when there are open connections to the real servers. When there are no sustained connections to any of the real servers, the least-connections predictor behaves like the round-robin predictor.

To specify different weight values for a real server of type **host** in a server farm, you can assign multiple IP addresses to the server. You can also use the same IP address of a real server with different port numbers.

The syntax of this command is as follows:

weight *number*

The *number* argument is the weight value assigned to a real server in a server farm. Enter an integer from 1 to 100. The default is 8.

For example, enter:

```
host1/Admin(config-rserver-host)# weight 50
```

To reset the configured weight of a real server to the default value of 8, enter:

```
host1/Admin(config-rserver-host)# no weight
```

Placing a Real Server in Service

Before you can start sending connections to a real server, you must place the server in service. You can place a real server in service by using the **inservice** command in either real server host or real server redirect configuration mode. The syntax of this command is as follows:

inservice



Note

Before you can place a real server that you have created in service, you must configure a minimum of an IP address for a server of type **host** or a webhost relocation string for a server of type **redirect**.

For example, to place a real server in service, enter:

```
host1/Admin(config-rserver-host)# inservice
```

Managing Real Servers

If a primary real server fails, the ACE takes that server out of service and no longer includes it in load-balancing decisions. If you configured a backup server for the real server that failed, the ACE redirects the primary real server connections to the backup server. For information about configuring a backup server, see the [“Configuring a Backup Server for a Real Server”](#) section.

The ACE can take a real server out of service for the following reasons:

- Probe failure
- ARP timeout
- Entering the **no inservice** command (see the [“Gracefully Shutting Down a Server”](#) section)

- Entering the **inservice standby** command (see the “[Gracefully Shutting Down a Server with Sticky Connections](#)” section)

For servers with sticky connections, the ACE removes those sticky entries from the sticky database when it takes the server out of service. For more information about stickiness, see [Chapter 5, Configuring Stickiness](#).

Use the **failaction purge** command to instruct the ACE to purge Layer 3 and Layer 4 connections if a real server fails. The default behavior of the ACE is to do nothing with existing connections if a real server fails. You can also enter the **failaction reassign** command, which instructs the ACE to send existing connections to a backup server (if configured) in case the primary server fails. For details about the **failaction** command, see the “[Configuring the ACE Action when a Server Fails](#)” section. For Layer 7 connections, you can instruct the ACE to rebalance each HTTP request by entering the **persistence-rebalance** command. For details about the **persistence-rebalance** command, see the “[Configuring HTTP Persistence Rebalance](#)” section in [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).

You can also manually take a primary real server out of service gracefully using either the **no inservice** command or the **inservice standby** command. Both commands provide graceful shutdown of a server. Use the **no inservice** command when you do not have stickiness configured and you need to take a server out of service for maintenance or a software upgrade. The **no inservice** command instructs the ACE to do the following:

- Tear down existing non-TCP connections to the server
- Allow existing TCP connections to complete
- Disallow any new connections to the server

With sticky configured, use the **inservice standby** command to gracefully take a primary real server out of service. The **inservice standby** command instructs the ACE to do the following:

- Tear down existing non-TCP connections to the server
- Allow current TCP connections to complete
- Allow new sticky connections for existing server connections that match entries in the sticky database
- Load balance all new connections (other than the matching sticky connections mentioned above) to the other servers in the server farm
- Eventually take the server out of service

Gracefully Shutting Down a Server

You can shut down a real server gracefully by using the **no inservice** command in either real server host or real server redirect configuration mode. This command causes the ACE to reset all non-TCP connections to the server. For TCP connections, existing flows are allowed to complete before the ACE takes the real server out of service. No new connections are allowed.



Note

The ACE resets all Secure Sockets Layer (SSL) connections to a particular real server when you enter the **no inservice** command for that server.

The syntax of this command is as follows:

```
no inservice
```

For example, to gracefully shut down a real server (for example, for maintenance or software upgrades), enter:

```
host1/Admin(config-rserver-host)# no inservice
```

Examples of Real Server Configurations

The following examples illustrate a running configuration that creates multiple real servers. These configurations show how to specify a host as the real server and how to specify a real server to redirect traffic to a different location. The real server configurations appear in bold in the following two examples:

- [Real Server that Hosts Content](#)
- [Real Server that Redirects Client Requests](#)

Real Server that Hosts Content

The following example creates three real servers, places them in service, and adds each real server to a server farm.

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

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 HTTP_PROBE
  predictor roundrobin
  rserver SERVER1
    weight 10
    inservice
  rserver SERVER2
    weight 20
    inservice
  rserver SERVER3
    weight 30
    inservice
```

Real Server that Redirects Client Requests

The following example specifies a series of real servers that redirect all incoming connections that match URLs ending in /redirect-100k.html, /redirect-10k.html, and /redirect-1k.html to the appropriate server farm.

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

```
rserver redirect SERVER1
  webhost-redirect http://192.168.120.132/redirect-100k.html 301
  inservice
rserver redirect SERVER2
  webhost-redirect http://192.168.120.133/redirect-10k.html 301
  inservice
rserver redirect SERVER3
  webhost-redirect http://192.168.120.134/redirect-1k.html 301
  inservice
serverfarm redirect SFARM1
  rserver SERVER1
  inservice
serverfarm redirect SFARM2
  rserver SERVER2
  inservice
serverfarm redirect SFARM3
  rserver SERVER3
  inservice
```

Configuring a Server Farm

This section describes server farms and how to configure them. It contains the following topics:

- [Server Farm Overview](#)
- [Server Farm Configuration Quick Start](#)
- [Creating a Server Farm](#)
- [Configuring a Description of a Server Farm](#)
- [Configuring the ACE Action when a Server Fails](#)
- [Associating Multiple Health Probes with a Server Farm](#)
- [Configuring AND Logic for Server Farm Probes](#)
- [Configuring the Server Farm Predictor Method](#)
- [Configuring Server Farm HTTP Return Code Checking](#)
- [Configuring a Partial Server Farm Failover](#)
- [Associating a Real Server with a Server Farm](#)
- [Configuring Additional Real Server Attributes in a Server Farm](#)
- [Configuring a Backup Server Farm](#)
- [Specifying No NAT](#)
- [Configuring Asymmetric Server Normalization](#)
- [Example of a Server Farm Configuration](#)

Server Farm Overview

Server farms are groups of networked real servers that contain the same content and that typically reside in the same physical location in a data center. Web sites often comprise groups of servers configured in a server farm. Load-balancing software distributes client requests for content or services among the real servers based on the configured policy and traffic classification, server availability and load, and other factors. If one server goes down, another server can take its place and continue to provide the same content to the clients who requested it.

Server Farm Configuration Quick Start

Table 2-2 provides a quick overview of the steps required to configure server farms. Each step includes the CLI command or a reference to the procedure required to complete the task. For a complete description of each feature and all the options associated with the CLI commands, see the sections following Table 2-2.

Table 2-2 Server Farm Configuration Quick Start

Task and Command Example

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

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

The rest of the examples in this table use the Admin context, unless otherwise specified. For details on creating contexts, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

2. Enter configuration mode.

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

3. Configure a server farm.

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)#
```

4. Associate one or more health probes of the same or different protocols with the server farm. Reenter the command as necessary to associate additional probes with the server farm.

```
host1/Admin(config-sfarm-host)# probe PROBE1
```

5. Configure the server-farm predictor (load-balancing) method.

```
host1/Admin(config-sfarm-host)# predictor leastconns
```

6. Configure HTTP return code checking for a server farm.

```
host1/Admin(config-sfarm-host)# retcode 200 500 check count
```

Table 2-2 Server Farm Configuration Quick Start (continued)**Task and Command Example**

7. (Optional) If you do not want the ACE to use NAT to translate the VIP to the server IP address, enter:

```
host1/Admin(config-sfarm-host)# transparent
```

8. Associate an existing real server with the server farm and enter server farm host real server configuration mode.

```
host1/Admin(config-sfarm-host)# rserver SERVER1 3000  
host1/Admin(config-sfarm-host-rs)#
```

9. (Optional) Configure a weight for the real server associated with the server farm. The ACE uses this weight value in the weighted round-robin and least-connections predictor methods. If you do not configure a weight, the ACE uses the global weight configured for the real server in real server configuration mode.

```
host1/Admin(config-sfarm-host-rs)# weight 50
```

10. Configure a backup server in case the real server becomes unavailable. The backup server can function as a sorry server.

```
host1/Admin(config-sfarm-host-rs)# backup rserver SERVER2 4000
```

11. Configure one or more probes of the same or different protocols to monitor the health of the real server in the server farm.

```
host1/Admin(config-sfarm-host-rs)# probe PROBE1_TCP
```

12. Configure connection limits for the real server in a server farm.

```
host1/Admin(config-sfarm-host-rs)# conn-limit max 5000 min 4000
```

13. Place the real server in service.

```
host1/Admin(config-sfarm-host-rs)# inservice  
host1/Admin(config-sfarm-host-rs)# Ctrl-z
```

14. (Recommended) Use the following command to display the server farm configuration. Make any modifications to your configuration as necessary, then reenter the command to verify your configuration changes.

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

15. (Optional) Save your configuration changes to flash memory.

```
host1/Admin# copy running-config startup-config
```

Creating a Server Farm

You can create a new server farm or modify an existing server farm and enter the server-farm configuration mode by using the **serverfarm** command in configuration mode. You can configure a maximum of 1023 server farms on each ACE.

The syntax of this command is as follows:

```
serverfarm [host | redirect] name
```

The keywords, arguments, and options are as follows:

- **host**—(Optional, default) Specifies a typical server farm that consists of real servers that provide content and services to clients and accesses server farm host configuration mode.
- **redirect**—(Optional) Specifies that the server farm consists only of real servers that redirect client requests to alternate locations specified by the relocation string in the real server configuration and accesses server farm redirect configuration mode. See the “[Configuring a Real Server Relocation String](#)” section. You can use a redirect server farm as a sorry server farm by specifying it as a backup server farm of type **redirect**. For details, see the “[Configuring a Sorry Server Farm](#)” section in [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).
- *name*—Unique identifier of the server farm. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to create a server farm of type **host** called SF1, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)#
```

For example, to create a server farm of type **redirect** called SF2, enter:

```
host1/Admin(config)# serverfarm redirect SF2  
host1/Admin(config-sfarm-redirect)#
```

After you create a server farm, you can configure the other server farm attributes and add real servers to it as described in the following sections:

- [Configuring a Description of a Server Farm](#)
- [Configuring the ACE Action when a Server Fails](#)
- [Associating Multiple Health Probes with a Server Farm](#)

- [Configuring AND Logic for Server Farm Probes](#)
- [Configuring the Server Farm Predictor Method](#)
- [Configuring Server Farm HTTP Return Code Checking](#)
- [Configuring a Partial Server Farm Failover](#)
- [Associating a Real Server with a Server Farm](#)
- [Specifying No NAT](#)
- [Configuring Asymmetric Server Normalization](#)

Configuring a Description of a Server Farm

You can provide a text description of a server farm by using the **description** command in either server farm host or server farm redirect configuration mode. The syntax of this command is as follows:

```
description text
```

For the *text* argument, enter a server farm description with a maximum of 240 alphanumeric characters.

For example, to enter a description of a server farm, enter:

```
host1/Admin(config)# serverfarm host SF1  
host1/Admin(config-sfarm-host)# description CURRENT EVENTS ARCHIVE
```

To remove the description of a server farm, enter:

```
host1/Admin(config-sfarm-host)# no description
```

Configuring the ACE Action when a Server Fails

You can set the action that the ACE takes with respect to connections if any real server fails in a server farm. You can configure the failure action by using the **failaction** command in either server farm host or server farm redirect configuration mode.

The syntax of this command is as follows:

```
failaction { purge | reassign }
```

The keywords and options are as follows:

- **purge**—Instructs the ACE to remove all connections to a real server if that real server in the server farm fails after you enter this command. The appliance sends a reset (RST) to both the client and the server that failed.
- **reassign**—Instructs the ACE to reassign the existing server connections to the backup real server (if configured) on the same VLAN interface if the real server fails after you enter this command. If a backup real server has not been configured for the failing server, this keyword has no effect and leaves the existing connections untouched in the failing real server.

Follow these configuration requirements and restrictions when you use the **reassign** keyword:

- Enable the **transparent** command (see the “[Specifying No NAT](#)” section) to instruct the ACE not to use NAT to translate the ACE VIP address to the server IP address. The **failaction reassign** command is intended for use in stateful firewall load balancing (FWLB) on your ACE, where the destination IP address for the connection coming in to the ACE is for the end-point real server, and the ACE reassigns the connection so that it is transmitted through a different next hop. See [Chapter 6, Configuring Firewall Load Balancing](#) for details.
- Configure the **mac-sticky** command on all server-side interfaces to ensure that packets that are going to and coming from the same server in a flow will traverse the same firewalls or stateful devices.
- Configure the **predictor hash address** command under firewall server farms.

In the absence of the **failaction** command, the default behavior of the ACE is to take a failed real server out of load-balancing rotation for new connections and to allow existing connections to complete. In this case, the ACE does not send the connections to a backup real server in the server farm if one is configured. You must configure the **failaction reassign** command to cause that behavior.

The backup server farm behavior is not affected by either the presence or the absence of the **failaction** command. If all real servers in the server farm fail, then the backup server farm (if configured) automatically takes over the failed server farm connections.

**Note**

To clear connections to real servers that have failed before you entered the **failaction** command, use the **clear conn** command. For details about the **clear conn** command, see the [“Clearing Real Server Connections”](#) section.

For example, to remove connections to a real server in a server farm that fail after you enter this command, enter:

```
host1/Admin(config)# serverfarm host SF1  
host1/Admin(config-sfarm-host)# failaction purge
```

For example, to specify that the ACE reassign the existing real server connections to the backup real server on the same VLAN interface if the real server fails, enter:

```
host1/Admin(config)# serverfarm host SF1  
host1/Admin(config-sfarm-host)# failaction reassign  
host1/Admin(config-sfarm-host)# transparent
```

To reset the behavior of the ACE to the default of taking no action if a real server fails, enter:

```
host1/Admin(config-sfarm-host)# no failaction
```

Associating Multiple Health Probes with a Server Farm

You can associate multiple health probes of the same or different protocols with each server farm. Each server farm supports the following probe types:

- DNS
- ECHO (TCP and UDP)
- Finger
- FTP
- HTTP
- HTTPS
- ICMP
- IMAP
- POP/POP3
- RADIUS
- RTSP
- Scripted
- SIP
- SMTP
- SNMP
- TCP
- Telnet
- UDP

By default, the real servers that you configure in a server farm inherit the probes that you configure directly on that server farm. Multiple probes that you configure on a server farm have an OR logic associated with them. If one of the probes configured on the server farm fails, all real servers configured in the server farm fail and enter the PROBE-FAILED state. You can also configure AND logic for server farm probes. For details, see the [“Configuring AND Logic for Server Farm Probes”](#) section.

You can associate a probe with a server farm by using the **probe** command in server farm host configuration mode only. The syntax of this command is as follows:

probe *name*

For the *name* argument, enter the name of an existing probe as a text string with no spaces and a maximum of 64 alphanumeric characters. For information about creating and configuring probes, see [Chapter 4, Configuring Health Monitoring](#).

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# probe PROBE1
```

To remove the probe from the server farm configuration, enter:

```
host1/Admin(config-sfarm-host)# no probe PROBE1
```

Configuring AND Logic for Server Farm Probes

By default, real servers that you configure in a server farm inherit the probes that you configure directly on that server farm. When you configure multiple probes on a server farm, the real servers in the server farm use an OR logic with respect to the probes. This means that if one of the probes configured on the server farm fails, all the real servers in that server farm fail and enter the PROBE-FAILED state. For information about creating and configuring probes, see [Chapter 4, Configuring Health Monitoring](#).

To configure the real servers in a server farm to use AND logic with respect to multiple server farm probes, use the **fail-on-all** command in server farm host configuration mode. With AND logic, if one server farm probe fails, the real servers in the server farm remain in the OPERATIONAL state. If all the probes associated with the server farm fail, then all the real servers in that server farm fail and enter the PROBE-FAILED state. This command applies to all probe types. You can also configure AND logic for probes that you configure directly on real servers in a server farm. For more information, see the “[Configuring AND Logic for Real Server Probes in a Server Farm](#)” section.

The syntax of this command is as follows:

fail-on-all

For example, to configure the SERVER1 and SERVER2 real servers in the SFARM1 server farm to remain in the OPERATIONAL state unless all server farm probes fail, enter:

```
host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# probe HTTP_PROBE
host1/Admin(config-sfarm-host)# probe ICMP_PROBE
host1/Admin(config-sfarm-host)# fail-on-all
host1/Admin(config-sfarm-host)# rserver server1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# exit
host1/Admin(config-sfarm-host)# rserver SERVER2
host1/Admin(config-sfarm-host-rs)# inservice
```

If either HTTP_PROBE or ICMP_PROBE fails, SERVER1 and SERVER2 remain in the OPERATIONAL state. If both probes fail, both real servers fail and enter the PROBE-FAILED state.

To remove the AND probe logic from the server farm, enter:

```
host1/Admin(config-sfarm-host)# no fail-on-all
```

Configuring the Server Farm Predictor Method

The load-balancing (predictor) method that you configure determines how the ACE selects a real server in a server farm to service a client request. You can configure the load-balancing method by using the **predictor** command in either server farm host or server farm redirect configuration mode. The syntax of this command is as follows:

```
predictor {hash {address [destination | source] [netmask]} | {content
  [offset number1] [length number2] [begin-pattern expression1]
  [end-pattern expression2]} | {cookie name1} | {header name2} |
  {layer4-payload [begin-pattern expression3] [end-pattern
  expression4] [length number3] [offset number4]} | {url [begin-pattern
  expression5] [end-pattern expression6]} | {least-bandwidth
  [assess-time seconds] [samples number5]} | {leastconns [slowstart
  time]} | {least-loaded probe name3} | {response {app-req-to-resp |
  syn-to-close | syn-to-synack}[samples number7]} | {roundrobin}
```

**Note**

The hash predictor methods do not recognize the weight value that you configure for real servers. The ACE uses the weight that you assign to real servers only in the least-connections, application-response, and round-robin predictor methods.

**Note**

You can associate a maximum of 1024 instances of the same type of regex with a Layer 4 policy map. This limit applies to all Layer 7 policy-map types, including generic, HTTP, RADIUS, RDP, RTSP, and SIP. You configure regexes in the following:

- Match statements in Layer 7 class maps
 - Inline match statements in Layer 7 policy maps
 - Layer 7 hash predictors for server farms
 - Layer 7 sticky expressions in sticky groups
 - Header insertion and rewrite (including SSL URL rewrite) expressions in Layer 7 action lists
-

This section contains the following topics:

- [Configuring the Hash Address Predictor](#)
- [Configuring the Hash Content Predictor](#)
- [Configuring the Hash Cookie Predictor](#)
- [Configuring the Hash Header Predictor](#)
- [Configuring the Hash Layer 4 Payload Predictor](#)
- [Configuring the Hash URL Predictor](#)
- [Configuring the Least-Bandwidth Predictor Method](#)
- [Configuring the Least-Connections Predictor](#)
- [Configuring the Least-Loaded Predictor](#)
- [Configuring the Application Response Predictor](#)
- [Configuring the Round-Robin Predictor](#)

Configuring the Hash Address Predictor

To instruct the ACE to select a real server using a hash value based on the source or destination IP address, use the **predictor hash address** command in server farm host or redirect configuration mode. Use this command when you configure firewall load balancing (FWLB). For more information about FWLB, see [Chapter 6, Configuring Firewall Load Balancing](#). The syntax of this command is as follows:

```
predictor hash address [destination | source] [netmask]
```

The keywords and options are as follows:

- **source**—(Optional) Selects the server using a hash value based on the source IP address.
- **destination**—(Optional) Selects the server using a hash value based on the destination IP address.
- **netmask**—(Optional) Bits in the IP address to use for the hash. If not specified, the default mask is 255.255.255.255.

For example, to configure the ACE to load balance client requests based on a hash value of the source address, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)# predictor hash address source  
255.255.255.0
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Content Predictor

To instruct the ACE to select a real server using a hash value based on a content string of the HTTP packet body, use the **predictor hash content** command in server farm host or redirect configuration mode. The syntax of this command is as follows:

```
predictor hash content [offset number1] [length number2] [begin-pattern  
expression1] [end-pattern expression2]
```

The keywords and options are as follows:

- **offset** *number1*—(Optional) Specifies the portion of the content that the ACE uses to stick the client on a particular server by indicating the bytes to ignore starting with the first byte of the payload. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the content.
- **length** *number2*—(Optional) Specifies the length of the portion of the content (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is infinity.



Note You cannot specify both the **length** and the **end-pattern** options in the same **content** command.

The offset and length can vary from 0 to 1000 bytes. If the payload is longer than the offset but shorter than the offset plus the length of the payload, the ACE sticks the connection based on that portion of the payload starting with the byte after the offset value and ending with the byte specified by the offset plus the length. The total of the offset and the length cannot exceed 1000.

- **begin-pattern** *expression1*—(Optional) Specifies the beginning pattern of the content string and the pattern string to match before hashing. If you do not specify a beginning pattern, the ACE starts parsing the HTTP body immediately following the offset byte. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).)

Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“”). The ACE supports the use of regular expressions for matching string expressions. See [Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing](#), for a list of the supported characters that you can use for matching string expressions.



Note When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use the “brackets ([])” character classes to match these symbols (for example, enter `www[.]xyz[.]com` instead of `www.xyz.com`). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

- **end-pattern** *expression2*—(Optional) Specifies the pattern that marks the end of hashing. If you do not specify either a length or an end pattern, the ACE continues to parse the data until it reaches the end of the field or the end of the packet, or reaches the maximum body parse length. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“”). The ACE supports the use of regular expressions for matching string expressions. See [Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing](#), for a list of the supported characters that you can use for matching string expressions.



Note You cannot specify both the **length** and the **end-pattern** options in the same **predictor hash content** command.

For example, to configure the ACE to load balance client requests based on a hash value of the content string of the HTTP packet body, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash content offset 25
length 32 begin-pattern abc123*
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Cookie Predictor

To instruct the ACE to select a real server using the hash value of a cookie name, use the **predictor hash cookie** command in server farm host or server farm redirect configuration mode. The syntax of this command is as follows:

```
predictor hash cookie name
```

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

For example, to configure the ACE to load balance client requests based on a hash value of the cookie name *corvette*, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)# predictor hash cookie corvette
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Header Predictor

To instruct the ACE to select a real server using the hash of an HTTP header value, use the **predictor hash header** command in server farm host or redirect configuration mode. The syntax of this command is as follows:

```
predictor hash header name
```

For the *name* argument, enter a header name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters, or enter one of the following standard HTTP headers:

- Accept
- Accept-Charset
- Accept-Encoding
- Accept-Language
- Authorization
- Cache-Control
- Connection

- Content-MD5
- Expect
- From
- Host
- If-Match
- Pragma
- Referrer
- Transfer-Encoding
- User-Agent
- Via

For example, to configure the ACE to load balance client requests based on the hash of the Host header value, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)# predictor hash header Host
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash Layer 4 Payload Predictor

To instruct the ACE to select a real server based on a hash value of a Layer 4 generic protocol payload, use the **predictor hash layer4-payload** command in server farm host or redirect configuration mode. Use this predictor to load balance packets from protocols that are not explicitly supported by the ACE. The syntax is as follows:

```
predictor hash layer4-payload [begin-pattern expression3] [end-pattern  
expression4] [length number3] [offset number4]
```

The keywords, options, and arguments are as follows:

- **begin-pattern** *expression3*—(Optional) Specifies the beginning pattern of the Layer 4 payload and the pattern string to match before hashing. If you do not specify a beginning pattern, the ACE starts parsing the HTTP body immediately following the offset byte. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“”). The ACE supports the use of regular expressions for matching string expressions. See [Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing](#) for a list of the supported characters that you can use for matching string expressions.



Note

When matching data strings, note that the period (.) and question mark (?) characters do not have a literal meaning in regular expressions. Use the “[]” character classes to match these symbols (for example, enter “www[.]xyz[.]com” instead of “www.xyz.com”). You can also use a backslash (\) to escape a dot (.) or a question mark (?).

- **end-pattern** *expression4*—(Optional) Specifies the pattern that marks the end of hashing. If you do not specify either a length or an end pattern, the ACE continues to parse the data until it reaches the end of the field or the end of the packet, or until it reaches the maximum body parse length. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).)

Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. Alternatively, you can enter a text string with spaces if you enclose the entire string in quotation marks (“”). The ACE supports the use of regular expressions for matching string expressions. See [Table 3-3 in Chapter 3, Configuring Traffic Policies for Server Load Balancing](#) for a list of the supported characters that you can use for matching string expressions.



Note You cannot specify both the **length** and the **end-pattern** options in the same **hash layer4-payload** command.

- **length** *number3*—(Optional) Specifies the length of the portion of the payload (starting with the byte after the offset value) that the ACE uses for sticking the client to the server. Enter an integer from 1 to 1000. The default is the entire payload.



Note You cannot specify both the **length** and the **end-pattern** options in the same **hash layer4-payload** command.

The offset and length can vary from 0 to 1000 bytes. If the payload is longer than the offset but shorter than the offset plus the length of the payload, the ACE sticks the connection based on that portion of the payload starting with the byte after the offset value and ending with the byte specified by the offset plus the length. The total of the offset and the length cannot exceed 1000.

- **offset** *number4*—(Optional) Specifies the portion of the payload that the ACE uses to load balance the client request to a particular server by indicating the bytes to ignore starting with the first byte of the payload. Enter an integer from 0 to 999. The default is 0, which indicates that the ACE does not exclude any portion of the payload.

For example, to configure the ACE to load balance client requests based on a hash value of a Layer 4 frame payload, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)# predictor hash layer4-payload  
begin-pattern abc123* length 250 offset 25
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Hash URL Predictor

To instruct the ACE to select a real server using a hash value based on the requested URL, use the **predictor hash url** command in server farm host or redirect configuration mode. Use this predictor method to load balance cache servers. Cache servers perform better with the URL hash method because you can divide the contents of the caches evenly if the traffic is random enough. In a redundant configuration, the cache servers continue to work even if the active ACE switches over to the standby ACE. For information about configuring redundancy, see the *Cisco 4700 Series Application Control Engine Appliance Administration Guide*.

The syntax of this command is as follows:

```
predictor hash url [begin-pattern expression1] [end-pattern expression2]
```

The keywords, options, and arguments are as follows:

- **begin-pattern** *expression1*—(Optional) Specifies the beginning pattern of the URL and the pattern string to match before hashing. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. If you want to match a URL that contains spaces, you must use “\x20” (without the quotation marks) for each space character.
- **end-pattern** *expression2*—(Optional) Specifies the pattern that marks the end of hashing. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).) Enter an unquoted text string with no spaces and a maximum of 255 alphanumeric characters for each pattern that you configure. If you want to match a URL that contains spaces, you must use “\x20” (without the quotation marks) for each space character.

For example, to configure the ACE to load balance client requests based on a hash value of a URL in the client HTTP GET request, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor hash url end-pattern
.cisco.com
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host) # no predictor
```

Configuring the Least-Bandwidth Predictor Method

To instruct the ACE to select the real server that processed the least amount of network traffic based on the average bandwidth that the server used over a specified number of samples, use the **predictor least-bandwidth** command in server farm host or redirect configuration mode. Use this predictor for heavy traffic use, such as downloading a video clip. This predictor is considered adaptive because the ACE continuously provides feedback to the load-balancing algorithm based on the behavior of the real server.

The ACE measures traffic statistics between itself and the real servers in a server farm in both directions, calculates the bandwidth for each server, and updates the bandwidth for each server every second. It then averages the bandwidth for each server over the number of configured samples. From the averaged bandwidth results, the ACE creates an ordered list of real servers and selects the server that used the least amount of bandwidth. The ACE continues to use the ordered list until the next bandwidth assessment period begins. Then the ACE updates the ordered list with the new values obtained from the most recent assess-time interval and the process starts over again.

The syntax of this command is as follows:

```
predictor least-bandwidth [assess-time seconds] [samples number5]
```

The keywords and arguments are as follows:

- **assess-time *seconds***—(Optional) Specifies the frequency with which the ACE reevaluates the server load based on the traffic bandwidth generated by the server and updates the ordered list of servers. Enter an integer from 1 to 10 seconds. The default is 2 seconds.
- **samples *number5***—(Optional) Specifies the maximum number of samples over which the ACE averages the bandwidth measurements and calculates the final load values for each server. Enter an integer from 1 to 16. Each value must be a power of 2, so the valid values are as follows: 1, 2, 4, 8, and 16. The default is 8.

**Note**

Because the ACE uses the same ordered list of servers during any one assess-time period, it load balances all new connections during that time period to only one server (the one with the lowest average bandwidth use). We recommend that you use this predictor for long-lived and steady traffic, such as downloading video. Applying this predictor to random or bursty short-lived traffic may cause unpredictable load-balancing results.

For example, to configure the ACE to select a real server based on the amount of bandwidth that the server used over a specified number of samples, enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor least-bandwidth samples 4
assess-time 6
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Least-Connections Predictor

To instruct the ACE to select the server with the fewest number of connections based on the server weight, use the **predictor leastconns** command in server farm host or redirect configuration mode. Use this predictor for processing light user requests (for example, browsing simple static web pages). For information about setting the server weight, see the [“Configuring a Real Server Weight”](#) section and the [“Configuring the Weight of a Real Server in a Server Farm”](#) section.

**Note**

Server weights take effect only when there are open connections to the servers. When there are no sustained connections to any of the servers, the least-connections predictor method behaves the same way as the round-robin method.

The syntax of this command is as follows:

```
predictor leastconns [slowstart time]
```

The optional **slowstart** *time* keyword and argument specify that the connections to the real server be in a slow-start mode. The ACE calculates a slow-start time stamp from the current time plus the *time* value that you specify. The ACE uses the time stamp as a fail-safe mechanism in case a real server stays in slow-start mode for a prolonged period of time (for example, with long-lived flows). Enter an integer from 1 to 65535 seconds. By default, **slowstart** is disabled.

Use the slow-start mechanism to avoid sending a high rate of new connections to servers that you have recently put into service. When a new real server enters slow-start mode, the ACE calculates and assigns an artificially high metric weight value to the new server and sends a small number of connections to the new server initially. The remaining connections go to the existing servers based on their weight and current connections. The real server (including the new server) with the lowest calculation (weight \times current connections) receives the next connection request for the server farm.

Each time that the ACE assigns a new connection to the new real server, the ACE checks the validity of the time stamp. If the time stamp has expired, the ACE takes the server out of slow-start mode. As the new server connections are finished (closed), the ACE decrements both the new server connection count and the metric weight value by the number of closed connections.

A real server exits slow-start mode when one of the following events occurs:

- Slow-start time stamp expires
- New real server metric weight reaches a value of 0

**Note**

It is possible for a new real server to remain in slow-start mode for a period that is longer than the *time* value you configure. This may happen because the ACE checks the time stamp only when it assigns a new connection to the real server.

The ACE places real servers in slow-start mode only if they have no existing connections to them. For example, if you have two real servers (RSERVER1 and RSERVER2) in a server farm that have the same number of active connections to them, slow start is configured for 180 seconds, and you add a new real server (RSERVER3) to the server farm, the ACE places the new real server in slow-start mode and sends a small number of new connections to it. The ACE equally divides the remainder of the new connections between RSERVER1 and RSERVER2. At this point, if RSERVER1 goes into any failed state (for example, PROBE-FAILED or ARP-FAILED) and then later it is placed back in service within 180 seconds of RSERVER3's addition to the server farm, the ACE does not put RSERVER1 in slow-start mode because of the connections to RSERVER1 that existed before the out-of-service event.

Instead, the ACE sends most or all of the new connections to RSERVER1 in an effort to make the connection count the same as RSERVER2 and ignores RSERVER3, which is still in slow-start mode. When the connection count for RSERVER1 and RSERVER2 are similar, then the ACE resumes sending a small number of connections to RSERVER3 for the duration of the slow-start timer.

If RSERVER1 returns to the OPERATIONAL state more than 180 seconds after you added RSERVER3 to the server farm, then RSERVER3 will be out of slow-start mode. In this case, the ACE sends most or all of the new connections to RSERVER1 in an effort to make the connection count the same as RSERVER2 or RSERVER3, whichever has the least number of connections.

To ensure that a failed server with existing connections enters slow-start mode when it is placed back in service, configure **failaction purge** for the real server in a server farm. This command removes all existing connections to the real server if it fails. For details about the **failaction purge** command, see the section [“Configuring the ACE Action when a Server Fails”](#).

For example, to configure the ACE to select the real server using the **leastconns** predictor and the slow-start mechanism, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)# predictor leastconns slowstart 3600
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring the Least-Loaded Predictor

To instruct the ACE to select the server with the lowest load, use the **predictor least-loaded** command in server farm host or redirect configuration mode. With this predictor, the ACE uses SNMP probes to query the real servers for load parameter values (for example, CPU utilization or memory utilization). This predictor is considered adaptive because the ACE continuously provides feedback to the load-balancing algorithm based on the behavior of the real server.

To use this predictor, you must associate an SNMP probe with it. The ACE queries user-specified OIDs periodically based on a configurable time interval. The ACE uses the retrieved SNMP load value to determine the server with the lowest load. For details about configuring SNMP probes, see [Chapter 4, Configuring Health Monitoring](#).

The syntax of this predictor command is as follows:

```
predictor least-loaded probe name
```

The *name* argument specifies the identifier of the existing SNMP probe that you want the ACE to use to query the server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to configure the ACE to select the real server with the lowest load based on feedback from an SNMP probe called PROBE_SNMP, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)# predictor least-loaded probe  
PROBE_SNMP  
host1/Admin(config-sfarm-host-predictor)#
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

You can influence the per-server load calculations that the ACE performs by using one or more of the following options:

- Autoadjust (see the [“Using Autoadjust”](#), [“Configuring Autoadjust for Average Load”](#), and [“Turning Off Autoadjust”](#) sections)
- Static real server weight (see the [“Configuring a Real Server Weight”](#) and the [“Configuring the Weight of a Real Server in a Server Farm”](#) sections)
- Current connection count (see the [“Configuring the Current Connection Count”](#) section)

- SNMP probe OID parameters (for details about configuring SNMP probes, see [Chapter 4, Configuring Health Monitoring](#))

Using Autoadjust

Whenever a server's load reaches zero, by default, the ACE uses the autoadjust feature to assign a maximum load value of 16000 to that server to prevent it from being flooded with new incoming connections. The ACE periodically adjusts this load value based on feedback from the server's SNMP probe and other configured options.

Using the least-loaded predictor with the configured server weight and the current connection count option enabled, the ACE calculates the final load of a real server as follows:

$$\text{final load} = \text{weighted load} \times \text{static weight} \times \text{current connection count}$$

where:

- *weighted load* is the load reported by the SNMP probe
- *static weight* is the configured weight of the real server
- *current connection count* is the total number of active connections to the real server

The ACE recalculates the final load whenever the connection count changes, provided that the **weight connection** command is configured. If the **weight connection** command is not configured, the ACE updates the final load when the next load update arrives from the SNMP probe. For details about the **weight connection** command, see the [“Configuring the Current Connection Count”](#) section.

Configuring Autoadjust for Average Load

To instruct the ACE to apply the average load of the server farm to a real server whose load reaches zero, use the **autoadjust average** command in server farm host or redirect predictor configuration mode. The average load is the running average of the load values across all real servers in the server farm. The syntax of this command is as follows:

```
autoadjust average
```

For example, to instruct the ACE to apply the average load of the server farm to a real server whose load value reaches zero, enter:

```
host1/Admin(config-sfarm-host-predictor)# autoadjust average
```

To reset the behavior of the ACE to the default of applying the maximum load value of 16000 to a real server whose load is zero, enter:

```
host1/Admin(config-sfarm-host-predictor)# no autoadjust average
```

Turning Off Autoadjust

There may be times when you want the ACE to send all new connections to a real server whose load is zero. In this case, use the **autoadjust off** command in server farm host predictor configuration mode. The syntax of this command is as follows:

autoadjust off

This command overrides the default behavior of the ACE of setting the load value for a server with a load of zero to 16000. When you configure this command, the ACE sends all new connections to the server that has a load of zero until the next load update arrives from the SNMP probe for this server.

If two servers have the same lowest load (either zero or nonzero), the ACE load balances the connections between the two servers in a round-robin manner.

For example, to turn off the autoadjust feature for all servers in a server farm so that servers with a load of zero receive all new connections, enter:

```
host1/Admin(config-sfarm-host-predictor)# autoadjust off
```

To return the behavior of the ACE to the default of assigning the maximum load value of 16000 to zero-load servers, enter:

```
host1/Admin(config-sfarm-host-predictor)# no autoadjust off
```

Configuring the Current Connection Count

To instruct the ACE to use the current connection count in the final load calculation for a real server, use the **weight connection** command in server farm host or redirect predictor configuration mode. The syntax of this command is:

weight connection

When you configure this option, the ACE includes the current connection count in the total load calculation for each real server in a server farm.

For example, to include the current connection count in the final load calculation for a server, enter:

```
host1/Admin(config-sfarm-host) # predictor least-loaded probe
PROBE_SNMP
host1/Admin(config-sfarm-host-predictor) # weight connection
```

To reset the behavior of the ACE to the default of excluding the current connection count from the load calculation, enter the following command:

```
host1/Admin(config-sfarm-host-predictor) # no weight connection
```

Configuring the Application Response Predictor

To instruct the ACE to select the server with the lowest average response time for the specified response-time measurement based on the current connection count and server weight (if configured), use the **predictor response** command in server farm host or redirect configuration mode. This predictor is considered adaptive because the ACE continuously provides feedback to the load-balancing algorithm based on the behavior of the real server.

To select the appropriate server, the ACE measures the absolute response time for each server in the server farm and averages the result over a specified number of samples (if configured). With the default **weight connection** option configured, the ACE also takes into account the server's average response time and current connection count. This calculation results in a connection distribution that is proportional to the average response time of the server.

The syntax of this command is as follows:

```
predictor response {app-req-to-resp | syn-to-close |
syn-to-synack} [samples number]
```

The keywords and arguments are as follows:

- **app-request-to-resp**—Measures the response time from when the ACE sends an HTTP request to a server to the time that the ACE receives a response from the server for that request.
- **syn-to-close**—Measures the response time from when the ACE sends a TCP SYN to a server to the time that the ACE receives a CLOSE from the server.

- **syn-to-synack**—Measures the response time from when the ACE sends a TCP SYN to a server to the time that the ACE receives the SYN-ACK from the server.
- **samples number**—(Optional) Specifies the number of samples over which you want to average the results of the response time measurement. Enter an integer from 1 to 16 in powers of 2. Valid values are 1, 2, 4, 8, and 16. The default is 8.

For example, to configure the response predictor to load balance a request based on the response time from when the ACE sends an HTTP request to a server to when the ACE receives a response back from the server and average the results over four samples, enter:

```
host1/Admin(config)# serverfarm SFARM1  
host1/Admin(config-sfarm-host)# predictor response app-req-to-resp  
samples 4
```

To reset the predictor method to the default of round-robin, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

To configure an additional parameter to take into account the current connection count of the servers in a server farm, use the **weight connection** command in server farm host predictor configuration mode. By default, this command is enabled. The syntax of this command is as follows:

weight connection

For example, enter:

```
host1/Admin(config)# serverfarm SF1  
host1/Admin(config-sfarm-host)# predictor response app-request-to-resp  
samples 4  
host1/Admin(config-sfarm-host-predictor)# weight connection
```

To remove the current connection count from the calculation of the average server response time, enter:

```
host1/Admin(config-sfarm-host-predictor)# no weight connection
```

Configuring the Round-Robin Predictor

To instruct the ACE to select the next real server in the list of real servers based on the server weight (if configured), use the **roundrobin** command in server farm host or redirect configuration mode. If you did not configure a server weight, the ACE selects the next server in the list regardless of the server weight (regular round-robin). If you configured a weight for the server, the ACE selects the next server in the list based on the configured weight of the servers in the server farm (weighted round-robin). For information about setting the server weight, see the “[Configuring a Real Server Weight](#)” section and the “[Configuring the Weight of a Real Server in a Server Farm](#)” section.

The syntax of this command is as follows:

roundrobin

For example, to select the next server in the list of servers based on the configured weight (if configured), enter:

```
host1/Admin(config)# serverfarm SF1
host1/Admin(config-sfarm-host)# predictor roundrobin
```

To reset the predictor algorithm to the default of **roundrobin**, enter:

```
host1/Admin(config-sfarm-host)# no predictor
```

Configuring Server Farm HTTP Return Code Checking

You can configure HTTP return-code checking (retcode map) for a server farm by using the **retcode** command in server farm host configuration mode only. You can specify a single return code number or a range of return code numbers. For example, you can instruct the ACE to check for and count the number of occurrences of such return codes as HTTP/1.1 200 OK, HTTP/1.1 100 Continue, or HTTP/1.1 404 Not Found.

The syntax of this command is as follows:

```
retcode number1 number2 check {count
| {log threshold_number reset seconds1}
| {remove threshold_number reset seconds1 [resume-service
seconds2]}}
```

The keywords, arguments, and options are as follows:

- *number1*—Minimum value for an HTTP return code. Enter an integer from 100 to 599. The minimum value must be less than or equal to the maximum value.
- *number2*—Maximum value for an HTTP return code. Enter an integer from 100 to 599. The maximum value must be greater than or equal to the minimum value.
- **check**—Checks for HTTP return codes associated with the server farm.
- **count**—Tracks the total number of return codes received for each return code number that you specify.
- **log**—Specifies a syslog error message when the number of events reaches the threshold specified by the *threshold_number* argument.
- *threshold_number*—Threshold for the number of events that the ACE receives before it performs the **log** or **remove** action.
- **reset seconds1**—Specifies the time interval in seconds over which the ACE checks for the return code for the **log** or **remove** action. Enter an integer from 1 to 4294967295.
- **remove**—Specifies a syslog error message when the number of events reaches the threshold specified by the *threshold_number* argument and the ACE removes the server from service.
- **resume-service seconds2**—(Optional) Specifies the number of seconds that the ACE waits before it resumes service for the real server automatically after taking the real server out of service because the **remove** option is configured. Enter an integer from 30 to 3600 seconds. The default setting is 300.

The ACE performs the log or remove actions only if the *threshold_number* value for a particular retcode is reached within a specified period of time. The time period is defined from the receipt of a retcode until the next reset time.

For example, if you enter **retcode 404 404 check 100 remove reset 300**, the remove action will not take place unless the ACE counts more than 100 occurrences of HTTP/1.x 404 within 300 seconds since the last time the counter was reset.

You can configure multiple retcode maps on each server farm. You can view hit counts for retcode checking by using the **show serverfarm** command. For details, see the [“Displaying Server Farm Statistics”](#) section.

For example, to check for and count the number of return code hits for all return codes from 200 to 500 inclusive, enter:

```
host1/Admin(config)# serverfarm host SF1  
host1/Admin(config-sfarm-host)# retcode 200 500 check count
```

To remove the HTTP return-code map from the configuration, enter:

```
host1/Admin(config-sfarm-host)# no retcode 200 500
```

Configuring a Partial Server Farm Failover

By default, if you have configured a backup server farm and all real servers in the primary server farm go down, the primary server farm fails over to the backup server farm. For details about configuring a backup server farm, see the [“Enabling Load Balancing to a Server Farm”](#) section in [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#) and [Chapter 5, Configuring Stickiness](#).

A partial server farm failover allows you to specify a failover threshold using the **partial-threshold** command. The first value that you specify for this command is a percentage of the real servers in a server farm that must remain active for the server farm to stay up. Each time that a server is taken out of service (for example, using the CLI, a probe failure, or the retcode threshold is exceeded), the ACE is updated. If the percentage of active real servers in a server farm falls below the specified threshold, the primary server farm fails over to the backup server farm (if configured).

With partial server farm failover configured, the ACE allows current connections on the remaining active servers in the failed primary server farm to complete and redirects any new connection requests to the backup server farm.

To bring the primary server farm back into service, you specify another threshold value for the **back-inservice** keyword. When the number of active servers is greater than the configured value for this keyword, the ACE places the primary server farm back in service.

You can enable partial server farm failover by using the **partial-threshold** command in server farm host configuration mode. The syntax of this command is as follows:

```
partial-threshold number1 back-inservice number2
```

The keywords and arguments are as follows:

- *number1*—Minimum percentage of real servers in the primary server farm that must remain active for the server farm to stay up. If the percentage of active real servers falls below this threshold, the ACE takes the server farm out of service. Enter an integer from 0 to 99.
- **back-inservice** *number2*—Specifies the percentage of real servers in the primary server farm that must be active again for the ACE to place the server farm back in service. Enter an integer from 0 to 99. The percentage value that you configure for the **back-inservice** keyword must be greater than or equal to the **partial-threshold** *number1* value.

For example, to configure a partial server farm failover, enter:

```
host1/Admin(config)# serverfarm host SF1  
host1/Admin(config-sfarm-host)# partial-threshold 40 back-inservice 60
```

To disable a partial server farm failover and return the ACE behavior to the default of failing over to the backup server (if configured) if all real servers in the server farm go down, enter:

```
host1/Admin(config-sfarm-host)# no partial-threshold
```

Associating a Real Server with a Server Farm

You can associate one or more real servers with a server farm and enter real-server server-farm configuration mode by using the **rserver** command in either server farm host or server farm redirect configuration mode. The real server must already exist. For information about configuring a real server, see the “[Configuring Real Servers](#)” section. You can configure a maximum of 4095 real servers in a server farm.

The syntax of this command is as follows:

```
rserver name [port]
```

The arguments are as follows:

- *name*—Unique identifier of an existing real server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port*—(Optional) Port number used for the real server port address translation (PAT). Enter an integer from 1 to 65535.

If you choose not to assign a port number for the real server association with the server farm, by default, the ACE automatically assigns the same destination port that was used by the inbound connection to the outbound server connection. For example, if the incoming connection to the ACE is a secure client HTTPS connection, the connection is typically made on port 443. If you do not assign a port number to the real server, the ACE automatically uses port 443 to connect to the server, which results in the ACE making a clear-text HTTP connection over port 443. In this case, you typically define an outbound destination port of 80, 81, or 8080 for the back-end server connection.

For example, to identify real server SERVER1 and specify port 80 for the outgoing connection, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 80
host1/Admin(config-sfarm-host-rs)#
```

To remove the real server association with the server farm, enter:

```
host1/Admin(config-sfarm-host)# no rserver server1 80
```

Configuring Additional Real Server Attributes in a Server Farm

After you have associated a real server with a server farm, you can configure other real server attributes as described in the following topics:

- [Configuring the Weight of a Real Server in a Server Farm](#)
- [Configuring a Backup Server for a Real Server](#)
- [Configuring Health Monitoring for a Real Server in a Server Farm](#)
- [Configuring AND Logic for Real Server Probes in a Server Farm](#)
- [Configuring Connection Limits for a Real Server in a Server Farm](#)
- [Configuring Rate Limiting for a Real Server in a Server Farm](#)
- [Placing a Real Server in Service](#)
- [Gracefully Shutting Down a Server with Sticky Connections](#)

Configuring the Weight of a Real Server in a Server Farm

The ACE uses a real server weight value with the weighted round-robin and least connections predictor methods. Servers with higher weight values receive a proportionally higher number of connections than servers with lower weight values. If you do not specify a weight in real configuration mode under a server farm, the ACE uses the weight that you configured for the global real server in real server configuration mode. See the [“Configuring a Real Server Weight”](#) section.

**Note**

Server weights take effect only when there are open connections to the servers. When there are no sustained connections to any of the servers, the least-connections predictor method behaves like the round-robin method.

You can configure the weight of a real server in a server farm by using the **weight** command in server farm host real server configuration mode. The syntax of this command is as follows:

weight *number*

The *number* argument is the weight value assigned to a real server in a server farm. Enter an integer from 1 to 100. The default is 8.

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# weight 50
```

To reset the configured weight of a real server to the default of 8, enter:

```
host1/Admin(config-sfarm-host-rs)# no weight
```

Configuring a Backup Server for a Real Server

You can designate an existing real server as a backup server for another real server in case that server becomes unavailable. A backup server is sometimes referred to as a sorry server. If the real server becomes unavailable, the ACE automatically redirects client requests to the backup server. You can configure a backup server by using the **backup-rserver** command in server farm host real server configuration mode.



Note

If you are configuring a backup server for a real server, be sure to specify the optional **standby** key word when you place the backup server in service. For details, see the [“Placing a Real Server in Service”](#) section.

The syntax of this command is as follows:

```
backup-rserver name [port]
```

The arguments are as follows:

- *name*—Name of an existing real server that you want to configure as a backup server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port*—(Optional) Port number used for the backup real server port address translation (PAT). Enter an integer from 0 to 65535.



Note

You can chain a maximum of three backup real servers.

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# backup-rserver SERVER2 4000
```

To remove the backup server from the configuration, enter:

```
host1/Admin(config-sfarm-host-rs)# no backup-rserver
```

Configuring Health Monitoring for a Real Server in a Server Farm

You can monitor the health of the real servers in a server farm by configuring one or more health probes (sometimes called keepalives) directly on the real servers in the server farm. You can associate multiple probes of the same or different protocols with each real server in a server farm. Once you configure the probes, the ACE periodically sends the probes to the real servers. If the ACE receives the appropriate responses from the servers, the ACE includes the servers in load-balancing decisions. If not, the ACE marks the servers as PROBE-FAILED, depending on the configured number of retries.

When you configure multiple probes on a real server in a server farm, the probes have an OR logic associated with them. This means that if one of the probes fails, the real server associated with that probe fails and enters the PROBE-FAILED state. You can also configure AND logic for multiple probes on a real server in a server farm. For details, see the [“Configuring AND Logic for Real Server Probes in a Server Farm”](#) section.

You can associate a health probe with a real server in a server farm by using the **probe** command in server farm host real server configuration mode. The syntax of this command is as follows:

```
probe probe_name
```

For the *probe_name* argument, enter the name of an existing probe as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters. For information about creating and configuring probes, see [Chapter 4, Configuring Health Monitoring](#).

For example, to configure a probe on a real server in a server farm, enter:

```
host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# rserver SERVER1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# probe TCP_PROBE
```

To remove the real server health probe from the configuration, enter:

```
host1/Admin(config-sfarm-host-rs)# no probe TCP_PROBE
```

Configuring AND Logic for Real Server Probes in a Server Farm

By default, multiple probes that you configure directly on a real server in a server farm have an OR logic associated with them. This means that, if one of the real server probes fails, the real server fails and enters the PROBE-FAILED state. To configure a real server in a server farm to remain in the OPERATIONAL state unless all probes associated with it fail (AND logic), use the **fail-on-all** command in server farm host real server configuration mode. This command is applicable to all probe types. For more information about creating and configuring probes, see [Chapter 4, Configuring Health Monitoring](#). The syntax of this command is as follows:

fail-on-all

You can selectively configure this command on only certain real servers in the server farm. Any real server that you do not configure with the **fail-on-all** command, maintains its default OR logic with respect to probes.

For example, to configure the SERVER1 real server in SFARM1 to remain in the OPERATIONAL state unless all associated probes fail, enter:

```
host1/Admin(config)# serverfarm SFARM1
host1/Admin(config-sfarm-host)# rserver SERVER1
host1/Admin(config-sfarm-host-rs)# inservice
host1/Admin(config-sfarm-host-rs)# probe HTTP_PROBE
host1/Admin(config-sfarm-host-rs)# probe ICMP_PROBE
host1/Admin(config-sfarm-host-rs)# fail-on-all
```

If either HTTP_PROBE or ICMP_PROBE fails, the SERVER1 real server remains in the OPERATIONAL state. If both probes fail, the real server fails and enters the PROBE-FAILED state.

To remove the AND probe logic from the real server in a server farm, enter:

```
host1/Admin(config-sfarm-host-rs)# no fail-on-all
```

Configuring Connection Limits for a Real Server in a Server Farm

You can prevent a real server in a server farm from becoming overloaded by limiting the number of connections allowed to that server. You can limit the number of connections to a real server by using the **conn-limit** command in server farm real server configuration mode. The syntax of this command is as follows:

```
conn-limit max maxconns min minconns
```

The keywords and arguments are as follows:

- **max** *maxconns*—Specifies the maximum number of active connections to a real server. When the number of connections exceeds the *maxconns* threshold value, the ACE stops sending connections to the real server until the number of connections falls below the configured *minconns* value. Enter an integer from 2 to 4000000. The default is 4000000.
- **min** *minconns*—Specifies the minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the *maxconns* threshold. The *minconns* value must be less than or equal to the *maxconns* value. The default is 4000000.

For example, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# conn-limit max 5000 min 4000
```

To remove the connection limit from a real server, enter:

```
host1/Admin(config-sfarm-host-rs)# no conn-limit
```

Configuring Rate Limiting for a Real Server in a Server Farm

In addition to preserving system resources by limiting the total number of active connections to a real server in a server farm (see the “[Configuring Real Server Connection Limits](#)” section), the ACE allows you to limit the connection rate and the bandwidth rate of a real server in a server farm. The connection rate is the number of connections per second that is received by the ACE and destined to a particular real server. The bandwidth rate is the number of bytes per second that is received by the ACE and destined for a particular real server.

Once the connection-rate limit or the bandwidth-rate limit of a real server in a server farm is reached, the ACE blocks any further traffic destined to that real server until the connection rate or bandwidth rate drops below the configured limit. The ACE removes the blocked real server from future load-balancing decisions and considers only those real servers that have a current connection rate or bandwidth less than the configured limit. By default, the ACE does not limit the connection rate or the bandwidth rate of real servers in a server farm.

You can also limit the connection rate and the bandwidth rate of the following:

- Real server at the aggregate level. For details, see the “[Configuring Real Server Rate Limiting](#)” section.
- Virtual server in a connection parameter map. For details, see the *Cisco 4700 Series Application Control Engine Appliance Security Configuration Guide*.



Note

The connection rate or bandwidth rate limit that you configure on a real server associated with a server farm cannot exceed the aggregate connection or bandwidth rate limit that you configure on a real server outside of a server farm.

You can limit the connection rate or the bandwidth rate of a real server in a server farm by using the **rate-limit** command in server farm host real server configuration mode or server farm redirect real server configuration mode. The syntax of this command is as follows:

```
rate-limit { connection number1 | bandwidth number2 }
```

The keywords and arguments are as follows:

- **connection** *number1*—Specifies the real server connection-rate limit in connections per second. Enter an integer from 2 to 4294967295. There is no default value.
- **bandwidth** *number2*—Specifies the real server bandwidth-rate limit in bytes per second. Enter an integer from 1 to 300000000. There is no default value.

For example, to limit the connection rate of a real server to 100000 connections per second, enter:

```
host1/Admin(config)# serverfarm host SF1  
host1/Admin(config-sfarm-host)# rserver SERVER1 4000  
host1/Admin(config-sfarm-host-rs)# rate-limit connection 100000
```

To return the behavior of the ACE to the default of not limiting the real-server connection rate, enter:

```
host1/Admin(config-sfarm-host-rs)# no rate-limit connection 100000
```

For example, to limit the real-server bandwidth rate to 5000000 bytes per second, enter:

```
host1/Admin(config)# serverfarm host SF1  
host1/Admin(config-sfarm-host)# rserver SERVER1 4000  
host1/Admin(config-sfarm-host-rs)# rate-limit bandwidth 5000000
```

To return the behavior of the ACE to the default of not limiting the real-server bandwidth, enter:

```
host1/Admin(config-sfarm-host-rs)# no rate-limit bandwidth 5000000
```

Placing a Real Server in Service

Before you can start sending connections to a real server in a server farm, you must place it in service. You can place a real server in a server farm in service by using the **inservice** command in either server farm host real server or server farm redirect real server configuration mode. The syntax of this command is as follows:

```
inservice [standby]
```

Use the optional **standby** keyword when you are configuring backup real servers. The **standby** keyword specifies that a backup real server remain inactive unless the primary real server fails. If the primary server fails, the backup server becomes active and starts accepting connections.



Note

You can modify the configuration of a real server in a server farm without taking the server out of service.

For example, to place a real server in service and have it remain inactive until the primary real server fails, enter:

```
host1/Admin(config)# serverfarm host SF1
host1/Admin(config-sfarm-host)# rserver SERVER1 4000
host1/Admin(config-sfarm-host-rs)# inservice standby
```

To take a backup real server out of the standby state and out of service for maintenance or software upgrades, enter:

```
host1/Admin(config-sfarm-host-rs)# no inservice
```

To take a backup real server out of the standby state only and put it in service so that it can start accepting connections, enter:

```
host1/Admin(config-sfarm-host-rs)# no inservice standby
```



Note

The **no inservice standby** command has an effect on a backup real server only if **inservice standby** is configured on that backup real server. The **standby** option applies only to a backup real server.

For example, consider the following configuration:

```
serverfarm SFARM1
  rserver SERVER1
    backup-rserver SERVER2
    inservice
  rserver SERVER2
    backup-rserver SERVER3
    inservice standby
  rserver SERVER3
    inservice standby
```

If you enter **no inservice standby** for SERVER2, the ACE takes SERVER2 out of the standby state and places it in service. SERVER2 will start to receive half of the connections; SERVER1 receives the other half. If SERVER1 fails, SERVER2 also will receive the connections of SERVER1.

**Note**

You can chain a maximum of three backup real servers.

Gracefully Shutting Down a Server with Sticky Connections

In addition to putting a backup real server in service standby, another use of the **inservice standby** command is to provide the graceful shutdown of primary real servers. Use this command to gracefully shut down servers with sticky connections. When you enter this command for a primary real server, the ACE does the following:

- Tears down existing non-TCP connections to the server
- Allows current TCP connections to complete
- Allows new sticky connections for existing server connections that match entries in the sticky database
- Load balances all new connections (other than the matching sticky connections mentioned above) to the other servers in the server farm
- Eventually takes the server out of service

For example, to perform a graceful shutdown on a primary real server with sticky connections in a server farm, enter:

```
host1/Admin(config)# serverfarm sf1
host1/Admin(config-sfarm-host)# rserver rs1
host1/Admin(config-sfarm-host-rs)# inservice standby
```

Configuring a Backup Server Farm

Configure a backup server farm to ensure that, if all the real servers in a server farm go down, the ACE continues to service client requests. You configure a backup server farm as an action in a Layer 7 policy map under a Layer 7 class map using the **serverfarm name1 [backup name2]** command. For details about configuring a backup server farm under a policy, see the “[Enabling Load Balancing to a Server Farm](#)” section in [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#). You can also configure a backup server farm under a sticky group using the same command. For details about a backup server farm under a sticky group, see the “[Backup Server Farm Behavior with Stickiness](#)” section in [Chapter 5, Configuring Stickiness](#).

Specifying No NAT

You can instruct the ACE not to use NAT to translate the VIP address to the server IP address by using the **transparent** command in serverfarm host configuration mode. Use this command in firewall load balancing (FWLB) when you configure the insecure and secure sides of the firewall as a server farm. For details about FWLB, see [Chapter 6, Configuring Firewall Load Balancing](#). The syntax of this command is as follows:

transparent

For example, enter:

```
host1/Admin(config-sfarm-host)# transparent
```

Configuring Asymmetric Server Normalization

Asymmetric Server Normalization (ASN) allows the ACE to load balance an initial request from the client to a real server; however, the server directly responds to the client bypassing the ACE. This behavior allows the acceleration of server to client communications and is transparent to the client. When the ACE operates in ASN, it does not perform any network translation when receiving packets destined to the VIP address. Traffic from the client hits the VIP address and the ACE uses the address as the destination address but rewrites the destination MAC address to the address of the real server.

This section contains the following topics:

- [ASN Sample Topology](#)
- [ASN Configuration Considerations](#)
- [Configuring ASN on the ACE](#)

ASN Sample Topology

[Figure 2-1](#) shows an ASN network topology where the VIP address does not belong to the IP subnet of the real servers.

Figure 2-1 *ASN Network Topology Example*



The topology consists of a client (10.20.10.101), a router, the ACE, and two real servers (10.10.15.100 and 10.10.15.101). Both real servers are grouped in a server farm represented by the VIP address 192.168.5.5. The router and the ACE are on subnet 10.10.15.0/24. When the client connects to 192.168.5.5, its packets hit the router. The preferred method for the router to forward traffic to the VIP address is a static route to 192.168.5.5 through 10.10.15.1. The router forwards the traffic destined to 192.168.5.5 to the MAC address of the ACE. There is no reason for the router to ever use Address Resolution Protocol (ARP) for 192.168.5.5.

ASN Configuration Considerations

When configuring ASN, note the following:

- Real servers must be configured with a virtual interface, sometimes referred to as a loopback interface.
- The virtual interface has an IP address of the VIP address.
- The virtual interface must not respond to ARP requests.
- ASN is a Layer 4-only feature. The load balancer does not participate in both legs of bidirectional client and server communications. Therefore, features that require TCP/SSL termination are not applicable to ASN environments. Features that are incompatible with ASN are as follows:
 - HTTP header parsing
 - HTTP header insert
 - Cookie sticky
 - SYN cookies
 - SSL termination or initiation
 - End-to-end SSL
- Because the ACE sees only the client-server leg of the connection, it cannot detect whether the connection has ended. With TCP being full duplex, the presence of a FIN in one direction does not mean that the ACE should tear down the entire connection. Therefore, connections age out after the connection inactivity timeout. The default connection inactivity timeouts are as follows:
 - ICMP—2 seconds
 - TCP—3600 seconds (1 hour)
 - UDP—120 seconds (2 minutes)

You can adjust the timeouts by using the **set timeout inactivity** command in parameter map connection configuration mode. To apply the timeout to a specific class of traffic, create a class map and policy map.

The following example creates a connection parameter map that times out all TCP connections after 20 seconds of inactivity and applies it under a Layer-4 policy map:

```
host1/Admin(config)# parameter-map type connection timeouts
host1/Admin(config-parammap-conn)# set time activity 20
host1/Admin(config-parammap-conn)# exit
host1/Admin(config)# policy-map multi-match LBPOL
host1/Admin(config-pmap)# class vip
host1/Admin(config-pmap-c)# connection advanced-options timeouts
```

The ACE automatically sends a TCP RST to the client and server if you enable TCP normalization. For more information of enabling TCP normalization, see the *Cisco 4700 Series Application Control Engine Appliance Security Configuration Guide*.

- Real servers must be Layer-2 adjacent to the load balancer when running in ASN. The reason is that the load balancer performs a destination MAC address rewrite, and the rewritten MAC always belongs to a real server. In the example in [Figure 2-1](#), real servers must be placed on VLAN 150. You cannot insert a routed hop between the ACE and the real servers.
- You must create a virtual interface on the real servers for each VIP address that exists on the load balancers. Depending on the topology, you may need to configure ARP suppression on the servers.

Configuring ASN on the ACE

To configure ASN on the ACE, do the following:

- Configure the server farm as a transparent server farm by using the **transparent** command in serverfarm host configuration mode. When you configure a transparent server farm, the ACE does not perform NAT of the VIP address to a real server address.
- Disable the TCP normalization on the client-side interface by using the **no normalization** command in VLAN interface configuration mode. Disabling normalization turns off ACE statefulness, allowing the ACE to accept client-originated TCP ACKs and data without having seen a full three-way handshake previously. For more information of disabling TCP normalization, see the *Cisco 4700 Series Application Control Engine Appliance Security Configuration Guide*.

With ASN, the servers respond directly to the client by bypassing the ACE. However, the clients traverse the ACE to reach the servers. The result is traffic asymmetry that is handled by disabling normalization.

The following example is the configuration of the real servers, server farm, and VIP address for the ASN topology in [Figure 2-1](#):

```
access-list inbound line 10 extended permit ip host 10.20.10.101 host
192.168.5.5

rserver host real1
    ip address 10.10.15.100
    inservice
rserver host real2
    ip address 10.10.15.101
    inservice

serverfarm host farm1
    transparent
    rserver real1
        inservice
    rserver real2
        inservice

class-map match-all vip
    2 match virtual-address 192.168.5.5 any

parameter-map type connection timeouts
    set time activity 20

policy-map type loadbalance first-match lbp01
    class class-default
        serverfarm farm1
policy-map multi-match LBPOL
    class vip
        loadbalance vip inservice
        loadbalance policy lbp01
        loadbalance vip icmp-reply active
        connection advanced-options timeouts

interface vlan 150
    description server-side
    ip address 10.10.15.1 255.255.255.0
    no normalization
    access-group input inbound
    service-policy input LBPOL
    no shutdown
```

```
ip route 0.0.0.0 0.0.0.0 10.10.15.2
```

When the client connects to the VIP address, the ACE logs a bidirectional connection creation that you can display by using the **show conn** command in Exec mode.

However, the second leg of the connection from the server to the client remains idle and its byte count does not increment.

To periodically check the health status of the servers, you must probe the virtual address, 192.168.5.5 in the previous configuration example. The following example is a simple ICMP probe configuration:

```
probe icmp ICMP1
  ip address 192.168.5.5
serverfarm host farm1
  probe ICMP1
```

Example of a Server Farm Configuration

The following example shows the running configuration that selects the next server in the list of real servers based on the server weight (weighted round-robin). The server farm configuration appears in bold in the example.

In this configuration, the ACE uses a real server weight value with the weighted round-robin predictor method. Real servers with higher weight values receive a proportionally higher number of connections than real servers with lower weight values.

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

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
rserver host SERVER4
  ip address 192.168.252.248
  inservice
rserver host SERVER5
  ip address 192.168.252.249
  inservice
rserver host SERVER6
```

```

ip address 192.168.252.250
inservice
serverfarm host SFARM1
  probe HTTP_PROBE
  predictor roundrobin
  rserver SERVER1
    weight 10
    inservice
  rserver SERVER2
    weight 20
    inservice
  rserver SERVER3
    weight 30
    inservice

serverfarm host SFARM2
  probe HTTP_PROBE
  predictor roundrobin
  rserver SERVER4
    weight 10
    inservice
  rserver SERVER5
    weight 20
    inservice
  rserver SERVER6
    weight 30
    inservice

class-map match-all L4WEB_CLASS
  2 match virtual-address 192.168.120.112 tcp eq www
policy-map type loadbalance first-match L7WEB_POLICY
  class class-default
    serverfarm SFARM1 backup SFARM2
policy-map multi-match L4WEB_POLICY
  class L4WEB_CLASS
    loadbalance vip inservice
    loadbalance policy L7WEB_POLICY
    loadbalance vip icmp-reply active
    nat dynamic 1 VLAN 120

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 L4WEB_POLICY

```

```
no shutdown
ip route 10.1.0.0 255.255.255.0 192.168.120.254
```

Displaying Real Server Configurations and Statistics

This section describes the commands that you can use to display information about the real-server configuration and statistics. It contains the following topics:

- [Displaying Real Server Configurations](#)
- [Displaying Real Server Statistics](#)
- [Displaying Real Server Connections](#)

Displaying Real Server Configurations

You can display information about the real-server configuration by using the **show running-config rserver** command in Exec mode. The syntax of this command is as follows:

```
show running-config rserver
```

Displaying Real Server Statistics

You can display summary or detailed statistics for a named real server or for all real servers by using the **show rserver** command in Exec mode. The syntax of this command is as follows:

```
show rserver [name] [detail]
```

The argument and option are as follows:

- *name*—(Optional) Identifier of an existing real server. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **detail**—(Optional) Displays detailed statistics for the real server name that you enter or for all real servers.

For example, to display detailed statistics for all configured real servers, enter:

```
host1/Admin# show rserver detail
```

Table 2-3 describes the fields in the **show rserver** command output.

Table 2-3 *Field Descriptions for the show rserver Command Output*

Field	Description
Summary Real Server Information	
Rserver	Name of the real server.
Type	Type of configured real server: HOST or REDIRECT.
State	Current state of the real server: <ul style="list-style-type: none"> • INACTIVE—Real server is not associated with a server farm • OPERATIONAL—Real server is in primary mode and is Up • STANDBY—Real server is in backup mode and is Up • OUTOFSERVICE—Real server is no longer in service (for both the primary and the backup real server)
Real	
Serverfarm	Name of the server farm to which the server belongs.
IP Address	IP address of the real server.
Weight	Weight of the real server in the server farm.
State	State of the server farm: OPERATIONAL or OUTOFSERVICE.
Current Connections	Number of active connections to the real server.
Total Connections	Total number of connections to the real server.
Detailed Real Server Information	
Rserver	Name of the real server.
Type	Type of configured real server: HOST or REDIRECT.
State	Current state of the real server: INACTIVE (not associated with a server farm), OPERATIONAL or OUTOFSERVICE.

Table 2-3 *Field Descriptions for the show rserver Command Output (continued)*

Field	Description
Description	User-entered text description of the real server with a maximum of 240 alphanumeric characters.
Max-conns	Configured maximum allowable number of active connections to a real server.
Min-conns	Configured minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold.
Out-of-rotation-count	Number of times that the real server was not considered for load balancing because the number of connections, connection rate, or bandwidth rate exceeded the configured limits of the server.
Conn-rate-limit	Configured connection rate limit of the real server in connections per second.
Bandwidth-rate-limit	Configured bandwidth rate limit of the real server in bytes per second.
Weight	Configured weight of the real server.
Redirect Str	For redirect servers only, the configured redirect string.
Redirect Code	For redirect servers only, the configured redirect code.
Redirect Port	For redirect servers only, the configured redirect port.
Real	
Serverfarm	Name of the server farm to which the server belongs.
IP Address	IP address of the real server.
Weight	Configured weight of the real server in the server farm.
State	Current state of the real server: OPERATIONAL or OUTFSERVICE.
Current Connections	Number of active connections to the real server.
Total Connections	Total number of connections to the real server.

Table 2-3 *Field Descriptions for the show rserver Command Output (continued)*

Field	Description
Max-Conns	Configured maximum allowable number of active connections to a real server.
Min-Conns	Configured minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold.
Conn-Rate-Limit	Configured connection rate limit in connections per second of the real server in the server farm.
Bandwidth-Rate-Limit	Configured bandwidth rate limit in bytes per second of the real server in the server farm.

Table 2-3 *Field Descriptions for the show rserver Command Output (continued)*

Field	Description
Out-of-Rotation-Count	Number of times that the real server was not considered for load balancing because the number of connections, connection rate, or bandwidth rate exceeded the configured limits of the server.
Total Conn-failures	<p>Total number of connection attempts that failed to establish a connection to the real server.</p> <p>For Layer 4 traffic with normalization on, the count increments if the three-way handshake fails to be established for either of the following reasons:</p> <ul style="list-style-type: none"> • A RST comes from the client or the server after a SYN-ACK. • The server does not reply to a SYN. The connection times out. <p>For Layer 4 traffic with normalization off, the count does not increment.</p> <p>For L7 traffic (normalization is always on), the count increments if the three-way handshake fails to be established for either of the following reasons:</p> <ul style="list-style-type: none"> • A RST comes from the server after the front-end connection is established • The server does not reply to a SYN. The connection times out.

Displaying Real Server Connections

You can display active inbound and outbound real server connections by using the **show conn rserver** command in Exec mode. The syntax of this command is as follows:

```
show conn rserver name1 [port_number][serverfarmname2]
```

The arguments and options are as follows:

- *name1*—Identifier of an existing real server whose connections you want to display. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port_number*—(Optional) Port number of the server. Enter an integer from 1 to 65535.
- **serverfarm***name2*—(Optional) Identifier of an existing server farm with which the real server is associated. Enter an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin# show conn rserver SERVER1 4000 serverfarm SFARM1
```

Table 2-4 describes the fields in the **show conn rserver** command output.

Table 2-4 *Field Descriptions for the show conn rserver Command Output*

Field	Description
Conn-ID	Numerical identifier of the connection.
NP	Network processor that is handling the inbound or outbound real server connection.
Dir	Direction of the traffic on the connection: in or out.
Proto	TCP/IP protocol used in the connection.
VLAN	Numerical identifier of the virtual LANs used in the inbound and the outbound connections.
Source	Source IP addresses and ports of the inbound and outbound connections.

Table 2-4 *Field Descriptions for the show conn rserver Command Output (continued)*

Field	Description
Destination	Destination IP addresses and ports of the inbound and outbound connections.
State	<p>Current state of the connection. Non-TCP connections display as "--". Possible values for TCP connections are as follows:</p> <ul style="list-style-type: none"> • INIT—Connection is closed. This is the initial state of a connection. • SYNSEEN—ACE received a SYN packet from a client. • SYNACK—ACE sent a SYNACK packet to a client. • ESTAB—Three-way handshake completed and the connection is established. • CLSFIN—ACE closed the connection with a FIN packet. • CLSRST—ACE closed the connection by resetting it. • CLSTIMEOUT—ACE closed the connection because the connection timed out. • CLOSED—Connection is half closed.

Clearing Real Server Statistics and Connections

This section describes the commands that you use to clear real server statistics and connection information. It contains the following topics:

- [Clearing Real Server Statistics](#)
- [Clearing Real Server Connections](#)

Clearing Real Server Statistics

You can reset statistics to zero for all instances of a particular real server regardless of the server farms that it is associated with by using the **clear rserver** command in Exec mode. The syntax of this command is as follows:

```
clear rserver name
```

The *name* argument is the identifier of an existing real server whose statistics you want to clear. Enter a real server name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.

For example, to reset the statistics of real server SERVER1, enter:

```
host1/Admin# clear rserver SERVER1
```

**Note**

If you have redundancy configured, you need to explicitly clear real-server statistics on both the active and the standby ACEs. Clearing statistics on the active appliance only leaves the standby appliance's statistics at the old values.

Clearing Real Server Connections

You can clear real server connections by using the **clear conn rserver** command in Exec mode. The syntax of this command is as follows:

```
clear conn rserver name1 [port] serverfarm name2
```

The arguments, options, and keyword are as follows:

- *name1*—Unique identifier of an existing real server whose connections you want to clear. Enter the real server name as unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- *port*—(Optional) Port number of the real server. Enter an integer from 1 to 65535.
- **serverfarm** *name2*—Specifies a unique identifier of the server farm with which the real server is associated. Enter the server farm name as an unquoted text string with no spaces and maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin# clear conn rserver SERVER1 4000 SFARM1
```

Displaying Server Farm Configurations and Statistics

This section describes the commands that you can use to display information about the server farm configuration and statistics. It contains the following topics:

- [Displaying Server Farm Configurations](#)
- [Displaying Server Farm Statistics](#)
- [Displaying Server Farm Connections](#)

Displaying Server Farm Configurations

You can display information about the server farm configuration by using the **show running-config serverfarm** command in Exec mode. The syntax of this command is as follows:

```
show running-config serverfarm
```

Displaying Server Farm Statistics

You can display summary or detailed server-farm statistics by using the **show serverfarm** command in Exec mode. The syntax of this command is as follows:

```
show serverfarm [name [retcode]] [detail]
```

The argument and options are as follows:

- **name**—(Optional) Statistics for the server farm specified in the *name* argument. Enter the name of an existing server farm as an unquoted text string with a maximum of 64 alphanumeric characters.
- **retcode**—(Optional) Displays the HTTP return codes associated with the server farm.
- **detail**—(Optional) Displays detailed statistics for the specified server farm.

For example, enter:

```
host1/Admin# show serverfarm sfarm1 detail
```

[Table 2-5](#) describes the fields in the **show serverfarm detail** command output.

Table 2-5 *Field Descriptions for the show serverfarm detail Command Output*

Field	Description
Serverfarm	Name of the server farm.
Type	Configured server farm type: HOST or REDIRECT.
Total Rservers	Total number of real servers associated with the server farm.
Active Rservers	Number of real servers that are active in the server farm.
Description	User-entered text description of the server farm with a maximum of 240 alphanumeric characters.
State	Current state of the server farm. Possible values are ACTIVE or INACTIVE.

Table 2-5 *Field Descriptions for the show serverfarm detail Command Output (continued)*

Field	Description
Predictor and field values	Configured load-balancing method and values for various fields associated with the predictor. Possible predictor values are as follows: <ul style="list-style-type: none"> • HASH-ADDRSRC • HASH-ADDRDEST • HASH-COOKIE • HASH-HEADER • HASH-HTTP-CONTENT • HASH-LAYER4-PAYLOAD • HASH-URL • LEASTBANDWIDTH • LEASTCONNS • LEASTLOADED • RESPONSE • ROUNDROBIN
Failaction	Action that the ACE takes for connections if a real server fails in a server farm. Possible actions are purge or none.
Back-Inservice	Configured value of the back-inservice keyword of the partial-threshold command. Specifies the minimum percentage of real servers in the primary server farm that must be active again for the ACE to place the server farm back in service.
Partial-Threshold	Configured value of the partial-threshold command. Specifies the minimum percentage of real servers in the primary server farm that must remain active for the server farm to stay up.
Num Times Failover	Number of time that the server farm failed over to the backup server farm.

Table 2-5 *Field Descriptions for the show serverfarm detail Command Output (continued)*

Field	Description
Num Times Back Inservice	Number of times that the ACE placed the server farm back in service after a failover.
Total Conn-Dropcount	Total number of connections that the ACE discarded because the number of connections exceeded the configured conn-limit max value. See the “ Configuring Real Server Connection Limits ” section.
Real	
Rserver	Name of the real server associated with the server farm.
IP Address:Port	IP address and port of the real server.
Weight	Weight assigned to the real server in the server farm.
State	Current state of the real server. Possible states are OPERATIONAL or OUTFSERVICE.
Current Connections	Number of active connections to the real server.
Total Connections	Total number of connections to the specified server farm.
Max-conns	Configured maximum allowable number of active connections to a real server.
Min-conns	Configured minimum number of connections that the number of connections must fall below before sending more connections to a server after it has exceeded the maximum connections threshold.
Out-of-rotation-count	Number of times that the real server was not considered for load balancing because the number of connections, connection rate, or bandwidth rate exceeded the configured limits of the server.
Conn-rate-limit	Configured connection rate limit of the real server in connections per second.
Bandwidth-rate-limit	Configured bandwidth rate limit of the real server in bytes per second.

Table 2-5 *Field Descriptions for the show serverfarm detail Command Output (continued)*

Field	Description
Retcode	Configured HTTP return code.
Average Response Time	For the response predictor, the average response time of the real server in milliseconds.
Connection Failures	Total number of connection attempts that failed to establish a connection to the real server.

Displaying Server Farm Connections

You can display the current active inbound and outbound connections for all real servers in a server farm by using the **show conn serverfarm** command in Exec mode. The syntax of this command is as follows:

```
show conn serverfarm name
```

The *name* argument is the name of the server farm whose real-server connections you want to display. Enter an unquoted alphanumeric text string with no spaces and a maximum of 64 alphanumeric characters.

For example, enter:

```
host1/Admin# show conn serverfarm sfarm1
```

Table 2-6 describes the fields in the **show conn serverfarm** command output.

Table 2-6 *Field Descriptions for the show conn serverfarm Command Output*

Field	Description
Conn-ID	Numerical identifier of the connection.
Dir	Direction of the traffic on the connection: in or out.
Proto	TCP/IP protocol used for this connection.
VLAN	Numerical identifier of the virtual LANs used in the inbound and the outbound connections.

Table 2-6 *Field Descriptions for the show conn serverfarm Command Output (continued)*

Field	Description
Source	Source IP addresses and ports of the inbound and outbound connections.
Destination	Destination IP addresses and ports of the inbound and outbound connections.
State	<p>Current state of the connection. Non-TCP connections display as "--". Possible values for TCP connections are as follows:</p> <ul style="list-style-type: none"> • INIT—Connection is closed. This is the initial state of a connection. • SYNSEEN—ACE received a SYN packet from a client. • SYNACK—ACE sent a SYNACK packet to a client. • ESTAB—Three-way handshake completed and the connection is established. • CLSFIN—ACE closed the connection with a FIN packet. • CLSRST—ACE closed the connection by resetting it. • CLSTIMEOUT—ACE closed the connection because the connection timed out. • CLOSED—Connection is half closed.

Clearing Server Farm Statistics

You can reset statistics to zero for all real servers in a specified server farm by using the **clear serverfarm** command in Exec mode. The syntax of this command is as follows:

```
clear serverfarm name [retcode]
```

The argument and option are as follows:

- *name*—Identifier of an existing server farm with real server statistics that you want to reset. Enter a server farm name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **retcode**—Resets all HTTP return code (retcode) statistics for the specified server farm. For information about configuring a server farm to perform HTTP retcode checking, see the “[Configuring Server Farm HTTP Return Code Checking](#)” section.

For example, to reset the statistics (including retcode statistics) of all real servers in server farm SF1, enter:

```
host1/Admin# clear serverfarm SF1 retcode
```

**Note**

If you have redundancy configured, you need to explicitly clear server-farm statistics on both the active and the standby ACEs. Clearing statistics on the active appliance only leaves the standby appliance’s statistics at the old values.

Where to Go Next

Once you are satisfied with your real server and server-farm configurations, configure an SLB traffic policy to filter interesting traffic and load balance it to the servers in your server farm, as described in [Chapter 3, Configuring Traffic Policies for Server Load Balancing](#).