



# 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 Cisco Application Control Engine (ACE) module. 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](#)

### 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 Real Server Connection Limits](#)
- [Configuring a Real Server Relocation String](#)
- [Configuring a Real Server Weight](#)
- [Placing a Real Server in Service](#)
- [Gracefully Shutting Down a Server](#)

## 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 1, Configuring Real Servers and Server Farms](#).

## 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](#).

You can instruct the ACE to purge Layer 3 and Layer 4 connections if a real server fails by using the **failaction purge** command. The default behavior of the ACE is to do nothing with existing connections if a real server fails. For details about the **failaction** command, see the [“Configuring the ACE Action on Server Failure”](#) 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 [“Enabling HTTP Persistence Rebalance”](#) section in [Chapter 1, 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 before taking the server out of service
- 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

## 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 Application Control Engine Module 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)#
```

4. (Recommended) Enter a description of the real server.

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

5. Configure an IP address for the real server in dotted-decimal notation.

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

6. Assign one or more existing probes to the real server for health monitoring.

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

---

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

Task and Command Example
<p>7. To prevent the real server from becoming overloaded, configure connection limits.</p> <pre>host1/Admin(config-rserver-host)# <b>conn-limit max 2000000 min 1500000</b></pre>
<p>8. If you plan to use the weighted round-robin or least connections predictor method, configure a weight for the real server.</p> <pre>host1/Admin(config-rserver-host)# <b>weight 50</b></pre>
<p>9. Place the real server in service.</p> <pre>host1/Admin(config-rserver-host)# <b>inservice</b> host1/Admin(config-rserver-host)# <b>Ctrl-Z</b></pre>
<p>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.</p> <pre>host1/Admin# <b>show running-config rserver</b></pre>
<p>11. (Optional) Save your configuration changes to flash memory.</p> <pre>host1/Admin# <b>copy running-config startup-config</b></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 16,384 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-redirection** 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**


---

All servers in a server farm should be of the same type: **host** or **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-redirect)#
```

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

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

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 unquoted alphanumeric text string with no spaces and a maximum of 240 characters.

For example, enter:

```
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

Configure an IP address so that the ACE can access a real server of type **host**. See the “[Configuring Real Servers](#)” section.

You can configure an IP address by using the **ip address** command in either real server host or real server redirect configuration mode. The syntax of this command is as follows:

```
ip address ip_address
```

For the *ip\_address* argument, enter a unique IPv4 address in dotted-decimal notation (for example, 192.168.12.15). The IP address must not be the address of an existing virtual IP address (VIP).

For example, to specify an address enter:

```
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-host)# probe probe1
```

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

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

## Configuring Real Server Connection Limits

To prevent a real server from being overburdened, 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 OUTOFSERVICE until the number of connections falls below the configured *minconns* value. Enter an integer from 2 to 4294967295. The default is 4294967295.
- **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 2 to 429496729. The default is 4294967295. 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 between the two NPs. If you configure an odd value for *maxconns*, one of the NPs will have a *maxconns* value that is one greater than the value on the other NP. With very small values for *maxconns*, it is possible for a connection to 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 for a real server. 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 a connection is closed on the NP with the *maxconns* value of 2. If the next connection request to the ACE goes to 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-host) # conn-limit max 5000 min 4000
```

To reset the real-server maximum connection limit and minimum connection limit to the default values of 4294967295, enter:

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

## 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-redirection** command in real server redirect configuration mode. The syntax of this command is as follows:

**webhost-redirection** *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-redir)# webhost-redirection  
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  
http://%h/redirectbusy.cgi?path=%p 302
```

## Configuring a Real Server Weight

You can configure the connection capacity of a real server of type **host** 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

---

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 leastconns predictor method behaves like the roundrobin method.

---

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
```

## 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 tear down 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 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
```

## 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 on Server Failure](#)
- [Associating Multiple Health Probes with a Server Farm](#)
- [Configuring the Server Farm Predictor Method](#)
- [Configuring Server Farm HTTP Return Code Checking](#)
- [Associating a Real Server with a Server Farm](#)
- [Configuring a Backup Server Farm](#)
- [Specifying No NAT](#)

## 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 Application Control Engine Module 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 SFARM1
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 the following command:

```
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 here, 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 16,384 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 or port number in the real server configuration and accesses server farm redirect configuration mode. See the “[Configuring a Real Server Relocation String](#)” section.
- *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 SFARM1, enter:

```
host1/Admin(config)# serverfarm SFARM1  
host1/Admin(config-sfarm-host)#
```

For example, to create a server farm of type **redirect** called SFARM2, enter:

```
host1/Admin(config)# serverfarm redirect SFARM2  
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 the ACE Action on Server Failure](#)
- [Associating Multiple Health Probes with a Server Farm](#)
- [Configuring the Server Farm Predictor Method](#)
- [Configuring Server Farm HTTP Return Code Checking](#)

- [Associating a Real Server with a Server Farm](#)
- [Specifying No NAT](#)

## 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-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 on Server Failure

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
```

The **purge** keyword specifies that the ACE remove the connections to a real server if that real server in the server farm fails after you enter the command. The ACE sends a reset (RST) to both the client and the server that failed.



### Note

If you do not configure this command, the ACE takes no action when a server fails. To clear connections to servers that have failed prior to entering 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 from a failed server in the server farm, enter:

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

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
- LDAP
- POP/POP3
- RADIUS
- Scripted
- SMTP
- TCP
- Telnet
- UDP

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 configuring probes, see [Chapter 4, Configuring Health Monitoring](#).

For example, enter:

```
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 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 {roundrobin | leastconns [slowstart time] | hash {address  
[destination | source] [netmask] | cookie name1 | header name2 | url  
[begin-pattern text] [end-pattern text]}}
```



### Note

You can associate a maximum of 1024 instances of the same type of regex with each Layer 3 and 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:

- 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

**Note**

---

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

---

The keywords and arguments are as follows:

- **roundrobin**—(Default) Selects the next server in the list of real servers based on server weight (weighted round-robin). For information about setting server weight, see the “[Configuring a Real Server Weight](#)” section and the “[Configuring the Weight of a Real Server in a Server Farm](#)” section.
- **leastconns**—Selects the server with the fewest number of connections based on server weight. Use this predictor for processing light user requests (for example, browsing simple static web pages). For information about setting 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 leastconns predictor method behaves like the roundrobin method.

---

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 timestamp 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 timestamp. If the timestamp 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 timestamp 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 timestamp 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 “[Configuring the ACE Action on Server Failure](#)” section.

- **hash address**—Selects the server using a hash value based on the source and destination IP addresses. Use the **hash address source** and **hash address destination** methods for firewall load balancing (FWLB). For more information about FWLB, see [Chapter 6, Configuring Firewall Load Balancing](#).
  - **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 is 255.255.255.255.
- **hash cookie name1**—Selects the server using a hash value based on the cookie name. Enter a cookie name as an unquoted text string with no spaces and a maximum of 64 alphanumeric characters.
- **hash header name2**—Selects the server using a hash value based on the header name. 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 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
- **hash url**—Selects the server using a hash value based on the requested URL. 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 Application Control Engine Module Administration Guide*.
    - **begin-pattern** *text*—Specifies the beginning pattern of the URL and the pattern string to parse. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 1, 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 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** *text*—Specifies the ending pattern of the URL and the pattern string to parse. You cannot configure different beginning and ending patterns for different server farms that are part of the same traffic classification. (See [Chapter 1, 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 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 select the server with the lowest number of connections in the server farm, enter:

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

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
```

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.

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-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 check count
```

## 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 16,384 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, the default behavior by the ACE is to automatically assign 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 will automatically use 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 would 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-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
```

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 Server Farm](#)
- [Configuring Connection Limits 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 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 leastconns predictor method behaves like the roundrobin 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 0 to 100. The default is 8.

For example, enter:

```
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 keywords and 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.

For example, enter:

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

To remove the backup server from the configuration, enter:

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

## Configuring Health Monitoring for 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) for the server farm. You can associate multiple probes of the same or different protocols with 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 out of service, depending on the configured number of retries.

You can associate a health probe with 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 example, enter:

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

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

```
host1/Admin(config-sfarm-host-rs)# no probe PROBE1_TCP
```

## Configuring Connection Limits for a Real Server in a Server Farm

You can prevent a real server 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 4294967295. The default is 4294967295.
- **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 4294967295.

Because the ACE has two network processors (NPs), the *maxconns* value for a real server is divided between the two NPs. If you configure an odd value for *maxconns*, one of the NPs will have a *maxconns* value that is one greater than the value on the other NP. With very small values for *maxconns*, it is possible for a connection to 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 for a real server. 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 a connection is closed on the NP with the maxconns value of 2. If the next connection request to the ACE goes to 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.

For example, enter:

```
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
```

## 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 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, enter:

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

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

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

## 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 a 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 [sticky] [aggregate-state]]** command. For details about configuring a backup server farm, see the [“Enabling Load Balancing to a Server Farm \(Configuring a Backup Server\)”](#) section in [Chapter 1, Configuring Traffic Policies for Server Load Balancing](#).

## 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
```

## 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
<b>Summary Real Server Information</b>	
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 OUTFSERVICE.
<b>Real</b>	
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 OUTFSERVICE.
Current Connections	Number of active connections to the real server.

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

<b>Field</b>	<b>Description</b>
Total Connections	Total number of connections to the real server.
<b>Detailed Real Server Information</b>	
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 OUTFSERVICE.
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.
Weight	Configured weight of the real server.
<b>Real</b>	
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.
Max-conns	Configured maximum allowable number of active connections to a real server.

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

Field	Description
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.
Total Conn-failures	Total number of connection attempts that failed to establish a connection to the real server.

## 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] [name2]
```

The arguments of this command 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.
- *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.
Dir	Direction of the traffic on the connection: in or out.
Prot	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.
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"> <li>• INIT—Connection is closed. This is the initial state of a connection.</li> <li>• SYNSEEN—ACE received a SYN packet from a client</li> <li>• SYNACK—ACE sent a SYNACK packet to a client</li> <li>• ESTAB—Three-way handshake completed and the connection is established</li> <li>• CLSFIN—ACE closed the connection with a FIN packet</li> <li>• CLSRST—ACE closed the connection by resetting it</li> <li>• CLSTIMEOUT—ACE closed the connection because the connection timed out</li> <li>• CLOSED—Connection is half closed</li> </ul>

# Clearing Real Server Statistics and Connections

This section describes the commands 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, then you need to explicitly clear real-server statistics on both the active and the standby ACEs. Clearing statistics on the active ACE only will leave the standby ACE 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 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 server farm configurational and statistical information. 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 [detail | retcode]]
```

The argument and option 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.
- **detail**—(Optional) Displays detailed statistics for the specified server farm.
- **retcode**—(Optional) Displays the HTTP return codes associated with the server farm.

For example, enter:

```
host1/Admin# show serverfarm name 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 a real server.
Description	User-entered text description of the server farm with a maximum of 240 alphanumeric characters.

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

Field	Description
Predictor	Configured load-balancing method. Possible values are as follows: <ul style="list-style-type: none"> <li>• ROUNDROBIN</li> <li>• LEASTCONNS</li> <li>• HASH ADDRESS</li> <li>• HASH COOKIE</li> <li>• HASH HEADER</li> <li>• HASH URL</li> </ul>
Failaction	Action that the ACE takes for connections if a real server fails in a server farm. Possible actions are purge or none.
Total Conn-dropcount	Total number of connections that the ACE discarded because the number of connections exceeded the configured <b>conn-limit max</b> value. See the <a href="#">“Configuring Real Server Connection Limits”</a> 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.
Maxconns	Configured maximum allowable number of active connections to a real server.

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

Field	Description
Minconns	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.
Total Conn-failures	Total number of connection attempts that failed to establish a connection to the real server.
NP	Number of the network processor (1 or 2) associated with the return codes.
Return Code	Number of the return code.
Action	Configured action for the return code, Count only.
Total Count	Total number of times that the ACE encountered the return code in network traffic and counted the return code.

## 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.
Prot	TCP/IP protocol used for this 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.
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"> <li>• INIT—Connection is closed. This is the initial state of a connection.</li> <li>• SYNSEEN—ACE received a SYN packet from a client</li> <li>• SYNACK—ACE sent a SYNACK packet to a client</li> <li>• ESTAB—Three-way handshake completed and the connection is established</li> <li>• CLSFIN—ACE closed the connection with a FIN packet</li> <li>• CLSRST—ACE closed the connection by resetting it</li> <li>• CLSTIMEOUT—ACE closed the connection because the connection timed out</li> <li>• CLOSED—Connection is half closed</li> </ul>

# 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
```

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

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

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

**Note**

---

If you have redundancy configured, then you need to explicitly clear server-farm statistics on both the active and the standby ACEs. Clearing statistics on the active ACE only will leave the standby ACE 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 1, Configuring Traffic Policies for Server Load Balancing](#).