Chapter 50

Load Balancer

Introduction

This chapter describes the load balancer, and how to configure load balancing for different situations.

☞ *To use the load balancer you need a special feature licence that enables the load balancer and the firewall. To purchase a feature licence contact your authorised distributor or reseller.*

One approach to meeting client requests for data and other services is to deliver all client requests to one *resource*. A resource may be a server, firewall or other routing device. A single resource may be able to process requests, but if it fails or suffers poor performance, requests are processed slowly or not at all.

The load balancer distributes incoming requests between multiple resources. These resources are grouped together in farms, or *resource pools*. The load balancer is made up of one or more *virtual balancers* that each map to one resource pool. Virtual balancers carry out actual load balancing operation by selecting one resource from a resource pool to process individual client requests. The resource then sends a response back to the virtual balancer, which in turn delivers the response to the requesting client.

Load Balancer Overview

The load balancer supports between 1 and 32 virtual balancers. A virtual balancer exposes one IP address and port, and optionally a URL, on the public side, and maps to a single resource pool. At least one virtual balancer must be configured before load balancing can commence. All connections to a virtual balancer's IP address and port or URL initiated from the public side are balanced over the resources in the virtual balancer's resource pool.

☞ *Take care with network settings when you configure the load balancer so that routing operates correctly. The load balancer should only be configured by experienced networking professionals.*

You can configure four different types of load balancer. Each different type of balancing requires a different type of virtual balancer. The types are:

- TCP (server) virtual balancer

- Route-based virtual balancer

- HTTP virtual balancer

- SSL virtual balancer

Each type of virtual balancer is described in detail in the *Load Balancer on the Router* section of this chapter. See Figure 50-1 on page 50-4 for an illustration of basic server load balancing.

Figure 50-1: Basic Server Load Balancing Operation
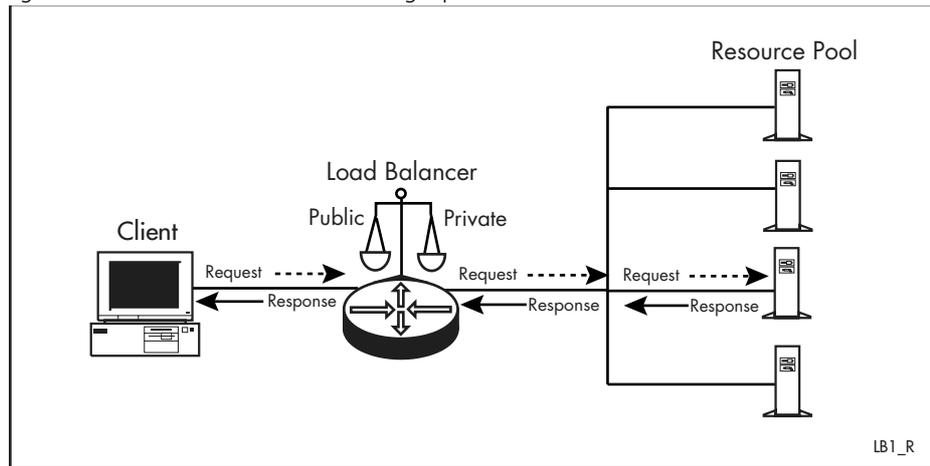
When a TCP or SSL type virtual balancer receives a packet on its public interface, it uses half Network Address Translation (NAT) to change the packet's destination IP address from its own address to that of the selected resource. When a resource responds to a client request, the packet it sends back to the client has a source address of the virtual balancer's public interface, even though the actual response came from the resource's IP address. For details on NAT, see Chapter 41, Firewall.

Virtual balancers optionally create *affinity table* entries that record persistency between a client and a particular resource. One affinity table exists for each type of virtual balancer. When a virtual balancer receives a client request, it checks its affinity table to see if an entry exists for that client. If there is an entry, that request and every new request from that client goes to the same resource. If the resource that the client has affinity with is not available, the virtual balancer selects another resource. Each time a table entry is used, it is refreshed. Affinity table entries time out after a configurable period, and are then deleted. A request from a client whose table entry has timed out is sent to a newly selected resource and an affinity table entry is created for the new connection.

When configuring a resource pool you choose a *selection algorithm*. This determines how the virtual balancer selects a resource from the pool when it receives a request from a client without an affinity table entry. Depending on the algorithm, you can give resources a higher preference if they are faster or more reliable, or each resource can be used in turn, regardless of its performance.

Resources can be maintained or added without disrupting services to clients. Because the load is shared across many resources, you can add a new resource, or carry out diagnostics on an existing resource, while the virtual balancer continues to connect clients with the remaining resources.

Load Balancer on the Router

There are currently no RFCs or other standards for load balancing.

The load balancer supports HTTP versions 1.0 and 1.1 as defined in RFCs 1945 and 2068 respectively.

Load balancing is not enabled until resource pools, resources and at least one virtual balancer are created, and the virtual balancer is enabled.

To enable the load balancer, use the command:

```
enable loadbalancer
```

To create a resource pool use the command:

```
add loadbalancer respool
```

To add resources to the resource pool use the command:

```
add loadbalancer resource
```

Once created, a resource is in the UP state, meaning it is enabled and is ready to accept connections.

To add a virtual balancer that connects to the resource pool, use the command:

```
add loadbalancer virtualbalancer
```

The newly created virtual balancer is disabled by default. To enable the virtual balancer so that load balancing can commence, use the command:

```
enable loadbalancer virtualbalancer
```

To enable debugging on the load balancer use the command:

```
enable loadbalancer debug
```

To view the status of the load balancer, use the command:

```
show loadbalancer
```

It is important to consider the Maximum Transmission Unit (MTU) of the private side interface on the load balancer in order to avoid packet fragmentation. Fragmentation affects load balancing throughput. If Ethernet is used on all connections between the client and resource, the MTU is always the same so it needs no modification. But if the MTU on the resource side interface is lower than that on the client side, the client side MTU should be lowered. For example, if the MTU on the client side is 1460 bytes, and the MTU on the resource side is 536 bytes, the client side MTU should be changed to 536 or less. Use the **set interface mtu** command on page 7-44 of Chapter 7, Interfaces to change this value.

TCP Virtual Balancer

A TCP based virtual balancer operates at layer 4 of the OSI model. This type of virtual balancer examines a packet's IP address to determine where the packet is from and to which resource it will be sent. This type of load balancing is suitable for when clients always have the same IP address. If IP addresses

change between sessions, the virtual balancer's affinity table cannot correctly identify the client. Affinity is between the client's IP address and the resource's IP address.

The destination address of any incoming packet received by the virtual balancer is translated with half NAT to that of the resource selected to process the request. The source IP address of a packet sent back to the client via the virtual balancer is translated to that of the virtual balancer's public interface.

Route-based Virtual Balancer

A route-based virtual balancer operates at layer 3 of the OSI model. This type of virtual balancer is used when packets are routed through a device such as a firewall that is not the packet's final destination, as illustrated in Figure 50-2 on page 50-6. This illustration shows firewalls configured in a 'sandwich' with load balancers on either side. The firewalls are the resources in a resource pool. Each load balancer must be configured identically for this scenario to work, except that on the second load balancer, the public and private interfaces are reversed. The source and destination addresses of packets travelling through a route-based virtual balancer are not changed with NAT, so this mode of operation is not suitable for server load balancing. For route-based load balancing to succeed, each virtual balancer must have multiple routes to the same destination network.

☞ *You should not use dynamic routing protocols such as RIP and OSPF with route-based virtual balancers.*

Figure 50-2: Route-Based Load Balancing with Firewalls
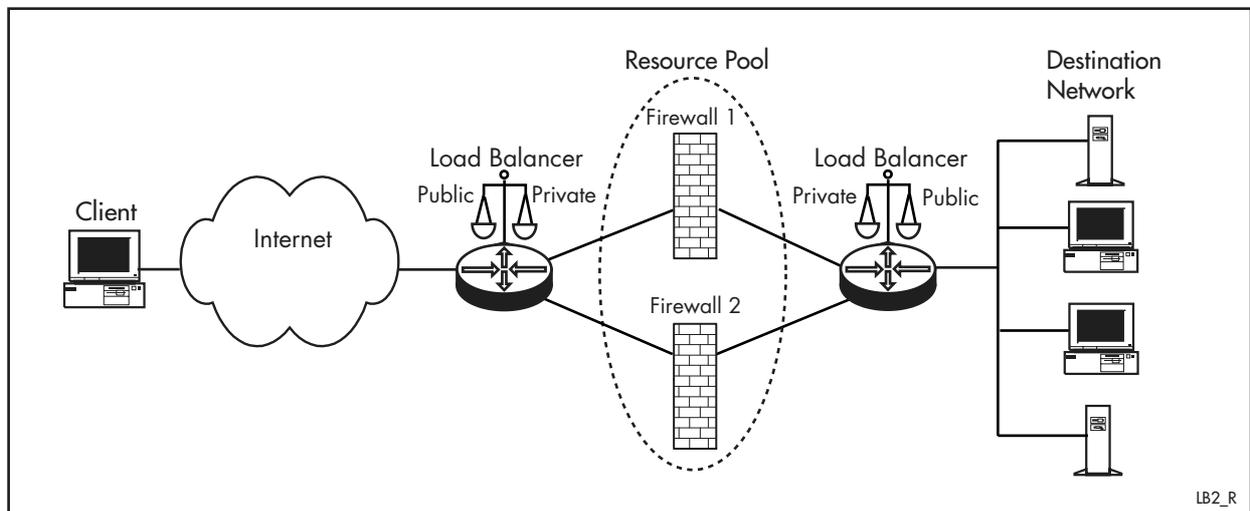
A route-based virtual balancer must be configured with multiple static routes through each resource in its resource pool to the destination network. The next hops from the virtual balancer must be each resource. As with TCP and HTTP load balancing, a route-based virtual balancer uses a selection algorithm to determine which resource is selected as the next hop. The chosen next hop is listed in the virtual balancer's affinity table. Like TCP balancing, affinity is between the client's IP address and the resource's IP address. The virtual balancer selects a resource for the request from its resource pool unless there is affinity between a destination IP address and a previously taken route. Messages returning to the client take the same path as those sent by the client.

☞ *Avoid using the Weighted Lottery selection algorithm when you configure a route-based virtual balancer.*

HTTP Virtual Balancer

An HTTP type virtual balancer operates at layer 7 of the OSI model, and is primarily for load balancing web servers or applications that tunnel over HTTP. An HTTP virtual balancer examines the cookies in HTTP requests and responses. The virtual balancer uses the cookie's *name* and *value* to create affinity table entries and make load balancing decisions. The cookie name is the character string *before* the equals ("=") sign in a cookie contained in an HTTP request, or the Set Cookie header of an HTTP response. The cookie value is the character string immediately *after* the "=" sign.

When the virtual balancer receives an HTTP request from a client that sends a cookie matching the name set with the **cookieheader** parameter, the virtual balancer checks its affinity table. The virtual balancer looks for an affinity table entry with a cookie whose value matches the value set by one of the resources. If this exists, the virtual balancer tries to send the request to that resource. If this resource is not available, another resource is chosen using the configured selection algorithm.

An HTTP virtual balancer processes a client request in four main steps, as described below. The numbers correspond to those in Figure 50-3 on page 50-7.

Figure 50-3: Basic HTTP Virtual Balancing
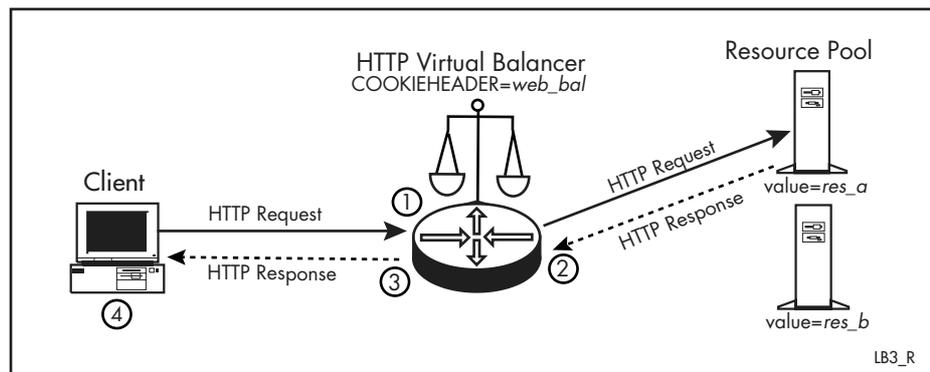
1. If the client's HTTP request does not contain a cookie that has been configured with *name=web_bal*, the virtual balancer uses a selection algorithm to choose a resource.

    The virtual balancer looks for a cookie with a name that matches the name set with the **cookieheader** parameter in the **add loadbalancer virtualbalancer** command. If no match is found, this means the client sending the request has not previously browsed to the web site served by the virtual balancer's resource pool. In this case, the virtual balancer uses its configured selection algorithm to choose a resource to process the request.

2. After the request is processed by the selected resource, an HTTP response is sent to the virtual balancer with the set cookie header: *web_bal=res_a*.

    When the HTTP request is processed by a resource, it adds a set cookie header to the HTTP response. This contains a cookie with a unique value that identifies the resource. The resource automatically adds this value; the cookie value is not set via the command line.

3. The virtual balancer creates an affinity table entry between the cookie's value, (*res_a*) and the resource that set the value. The HTTP response is then forwarded to the client.

4. The next HTTP request from this client contains a cookie with: *web_bal=res_a*. When this value is in the affinity table, the virtual balancer sends the request to *res_a*.

If a client has a different IP address during different sessions, the virtual balancer can still identify the client, assuming that each client has a unique cookie. The virtual balancer does not select a resource until the whole HTTP request arrives. With other types of balancing the resource is selected when the first packet in the transmission is received. For details on cookies including their syntax, see *RFC 2965: HTTP State Management Mechanism*.

The HTTP response is examined for its status code. The virtual balancer can have its **httperrorcode** parameter set to look for server error codes, which start at 500. If an HTTP response has an error code that matches one in this parameter, the resource that sent it is marked as failed, and the request is sent to a new resource chosen with the selection algorithm. A trigger can be configured to perform recovery or notification actions if this happens.

SSL Load Balancer

The load balancer operates with Secure Sockets Layer (SSL) in two different ways. You can either configure an SSL type virtual balancer, or enable SSL for an HTTP type virtual balancer. For details on SSL see Chapter 51, Secure Sockets Layer (SSL).

SSL encrypts data and ensures that only trusted devices can exchange confidential information. Affinity is important for SSL connections because it is necessary to maintain a *sticky*, or persistent, connection. This is when a client maps to the same resource. For example, a sticky connection is required when a client browses to an SSL secured banking web site. If the load balancer switches resources before the client completes the transaction, a new SSL connection is needed for the new resource. The SSL handshake sequence, which negotiates the security options for the connection, can be time-consuming. A sticky connection improves performance because the handshake time is cut down. Both types of SSL load balancing allow for sticky connections. The public port for SSL is usually 443.

If you want to use SSL and the load balancer with the Single DES algorithm, you need:

■ a Mini Accelerator Card (MAC), and

■ a load balancer and firewall feature licence.

If you want to use SSL and the load balancer with the Triple DES algorithm, you need:

■ a Mini Accelerator Card (MAC), and

■ a load balancer and firewall feature licence, and

■ a 3DES feature licence.

In the unlikely event of the router's flash memory becoming corrupted, you may lose current feature licences, including those enabled by default. If this happens, contact your authorised distributor or reseller.

SSL Virtual Balancing

You can configure an SSL type virtual balancer. This type of virtual balancer examines the SSL Client Hello or Server Hello, transmitted as part of the handshake, for the SSL Session ID. SSL virtual balancers create affinity table entries based on this Session ID. All Server Hellos with Session ID numbers not already listed in the affinity table create new entries. This happens if the Client ID is set to zero, or the Client ID is not present in the Server Hello. The Session ID is contained in the Server Hello for new connections, and is in the Client Hello for resumed connections. A selection algorithm or affinity table entry determines the (secured) resource that processes client requests.

HTTP Virtual Balancing with SSL Enabled

You can enable an HTTP type virtual balancer for SSL. In this situation, the SSL connection from the client is terminated at the virtual balancer's public interface. This allows it to decrypt SSL messages and balance requests based on the cookie, as with normal HTTP type balancing. Affinity table entries are based on cookies. Optionally, you can enable SSL for the connection from the private interface of the virtual balancer to the resource pool. This requires a separate SSL connection. See Figure 50-4 on page 50-9.

Figure 50-4: HTTP Load Balancing with SSL connection
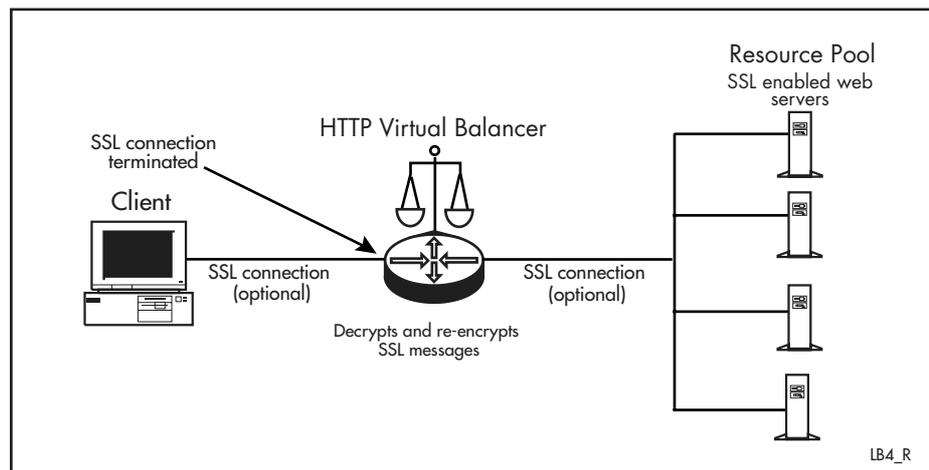
Load Balancer and Firewall

The load balancer uses the firewall's Network Address Translation (NAT) facility, so a firewall must be configured for load balancing to work. For details on configuring a firewall, refer to Chapter 41, Firewall.

When configuring a firewall, it is essential that the firewall's policies allow traffic from clients to travel to and from the public interface and port on each configured virtual balancer. If the firewall blocks this traffic, the load balancer does not operate. If you already have a firewall configured on the routing device that will act as a load balancer, ensure that existing policies allow this traffic flow.

When the load balancer is enabled, the value of the **orphantimeout** parameter in the **set loadbalancer** command overwrites timeout values configured in the firewall policy.

☞ *Ensure there are no proxies configured on a policy where load balancing occurs. Take care if changing the firewall's NAT settings because this could prevent the load balancer from operating correctly.*

Redundancy

As a backup measure, you can configure two load balancers on separate Allied Telesyn routing devices as a *redundant pair*. With redundancy, in the unlikely event of one load balancer failing, the identically configured backup device immediately attempts to take over load balancing operations.

The two balancers must have their management interfaces connected, and their public and private interfaces connected to the same networks. The management interfaces are likely to be the same as the private interface, but should never be the public interface. The load balancers should be manually configured with identical information. Any changes to one load balancer, such as the addition of a resource or virtual balancer, must be manually replicated on the other balancer. In the pair, the *master* balancer carries out the load balancing, and communicates the status of its resources and connections to the *slave* (backup) balancer. The slave immediately attempts to take over balancing duties if the master fails. The master and slave balancers are also known as *fail-over* balancers. See Figure 50-5 on page 50-10.

☞ *To configure redundancy you need two separate Allied Telesyn routing devices with load balancer feature licences. The routing devices do not have to be identical, but they must have the same version of load balancer software.*

Figure 50-5: Load Balancer Redundancy.
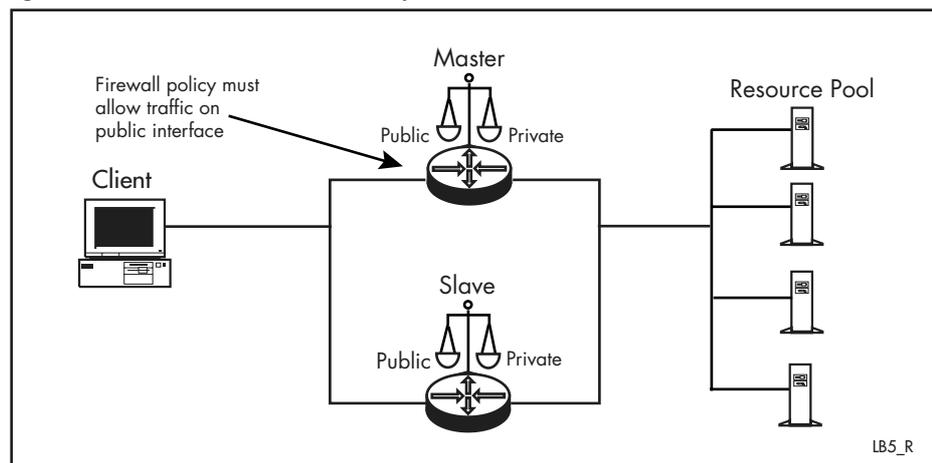
To set up a redundant pair of load balancers, first configure one load balancer with all the information it needs to commence balancing operations. Then configure the other load balancer with identical information, so the two are exact copies. Next, physically connect the two devices and then enter the information they need to communicate with each other using the **set loadbalancer redundancy** command on page 50-38. The master is most likely to be the first load balancer that is enabled. When both devices are powered up

at the same time, they negotiates to determine who is the master. In this case, the master is the device with the highest IP address on the interface it uses to communicate redundancy information. Finally, use the **enable loadbalancer redundancy** command on page 50-34 to start redundancy.

When a load balancer becomes the master in the redundant pair, it creates a new IP address *alias* for the public interface. This redundant IP address has been set with the **set loadbalancer redundancy** command on page 50-38. This alias then becomes a new logical interface on the load balancer, and must be added to the firewall policy so that traffic can be balanced on it.

☞ *Load balancing is performed on the redundant IP address, and this address is backed up by the slave.*

The newly created logical interface cannot be added automatically to the load balancer's firewall policy, so you must use a trigger to notify the firewall. This trigger activates when the new interface is created. The trigger executes a script that adds the new interface to the firewall policy. For details on triggers, see Chapter 30, Trigger Facility and "Triggers" on page 50-13.

Configuring Redundancy

When configuring redundancy, the load balancer is configured as usual, except for the public and private interfaces. On the public interface, do not configure the redundant IP address on either load balancer. This IP address is added automatically when one load balancer becomes the master, and is removed automatically when it is no longer the master. You do need to configure an IP address that is on the same subnet as the intended redundant IP address. For example, if the redundant IP address is 172.168.1.2, then configure an interface on each redundant load balancer with the same subnet as 172.168.1.0. On the private interface, do not configure the private IP address, as this is added by scripts later. But you do need to configure the peer IP addresses for both devices. These addresses should each be on the same subnet, but on a different subnet than the intended private IP address.

Enter the redundancy information on each device with the **set loadbalancer** command on page 50-37. To set the device's peer IP address to 192.168.2.201, the listenport to 5555, the redundant IP address to 172.207.1.2, the public interface to vlan1 and the redundant IP addresses's subnet mask to 255.255.0.0, use the command:

```
set loadbalancer redundancy peerip=192.168.2.201
    listenport=5555 redundanip=172.207.1.2 publicint=vlan1
    redunmask=255.255.0.0
```

A critical part of configuring redundancy is to set up triggers that activate scripts. These scripts update the firewall policy, and add and delete the private IP address. The script that executes when a redundant load balancer becomes the master must add the private IP address to a private interface and add this interface to the firewall policy. It must also add the redundant IP interface to the firewall policy. This script executes when a trigger activates after one device becomes the master (master.scp).

Another script executes when one device becomes the slave (slave.scp). The slave script deletes the master's redundant IP interface, its private interface and deletes the two firewall rules created by the master script that allowed traffic over the master's interfaces.

To set up a trigger that executes the script *master.scp* when one load balancer becomes the master, use the command:

```
ena trigger

create trigger=1 module=lb event=master script=master.scp
```

Next, to set up a trigger that executes the script *slave.scp* when the master load balancer fails, and the slave becomes the master, use the command:

```
create trigger=2 module=lb event=slave script=slave.scp
```

The script *master.scp* should contain the following commands to add the redundant IP interface to the firewall. Load balancing occurs on this interface:

```
add firewall policy=name interface=interface type=public

add firewall policy=name rule=rule-id interface=interface
    action=allow protocol=tcp port=80
```

The script *master.scp* should contain the following commands to add the private IP interface to the firewall:

```
add ip interface=interface ip=ip-add

add firewall policy=name interface=interface type=private

add firewall policy=name rule=rule-id interface=interface
    action=allow protocol=all

enable loadbalancer healthpings
```

The script *slave.scp* should contain the following commands to remove the redundant IP interface from the firewall policy:

```
delete firewall policy=name int=interface

delete firewall policy=name rule=rule-id
```

The script *slave.scp* should contain the following command to remove the redundant IP interface:

```
delete ip interface=interface
```

The script *slave.scp* should contain the following commands to remove the private IP interface from the firewall policy:

```
delete ip interface=interface

delete firewall policy=name interface=interface

delete firewall policy=name rule=rule-id

disable loadbalancer healthpings
```

For details on creating scripts, refer to Chapter 34, Scripting.

To enable the redundancy protocol, use the command:

```
enable loadbalancer redundancy
```

Health Checks

The load balancer regularly conducts health checks to establish the status of:

■ the resources in its resource pools.

■ the fail-over balancer.

■ the load balancing process.

Resource Health Checks

Resource health checks are first performed by pinging each resource's IP address. Resources are pinged one minute after they are first enabled. Once the load balancer has pinged every resource, it waits for one minute before starting the next round of pings. A resource is marked as CLOSING if one ping fails to return a response. If the next ping also fails, the resource is marked as DOWN. A successful ping brings the resource back UP. A change in the state of a resource is replicated on the fail-over balancer if redundancy is enabled. You can disable health check pings with the **disable loadbalancer healthpings** command on page 50-30.

The second form of health checking is the recorded number of RST messages sent to a resource in the last 100 connections. A RST or 'reset' message is generated by TCP if it receives anything unexpected. If the percentage of resource RST messages exceeds the configured limit, that resource is marked as DOWN. If this happens, the administrator must intervene to put the resource back UP using the **enable loadbalancer resource** command on page 50-35.

The third form of health checking applies only to HTTP type virtual balancers. If a resource returns an HTTP response with an error code that matches a code entered with the **add loadbalancer virtualbalancer httperrorcode** command on page 50-25, that resource is marked as DOWN.

Fail-over Load Balancer Health Checks

Fail-over balancer health checks are performed to establish the status of the other balancer in the redundant pair. The master balancer checks the status of the slave by sending management packets through the management LAN or private side interface to the opposite interface on the slave balancer. When the packets are not acknowledged, the slave is deemed to have failed and the master does not send the slave any more redundancy information. To correct this you must disable and then re-enable redundancy on both the active and passive balancers. The slave checks the status of the master by monitoring heartbeat messages sent to it from the master. When the slave fails to receive three consecutive heartbeats, it assumes that the master is down. When this happens, an error is recorded, the slave assumes balancing duties, and an administrator must put the master balancer back online.

Load Balancer Process

The load balancing process is checked for orphaned connections every two minutes. Timed-out connections found in affinity tables are deleted.

Triggers

The Trigger Facility can be used to automatically run specified command scripts when particular triggers are activated. When a trigger is activated by an event, parameters specific to that event are passed to the script that is run. For a full description of the Trigger Facility, see *Chapter 30, Trigger Facility*.

Triggers can be created to perform the following:

■   change the state of a particular resource or virtual balancer, or

■   match an HTTP response with an HTTP error code.

☞   *If you are configuring two load balancers for redundancy, you must create a trigger to add a logical interface to the firewall policy when one load balancer becomes the master.*

Event   RESSTATE

Description   A change in state is activated for a particular resource, one of:

■   UP -> CLOSING

■   CLOSING -> DOWN

■   UP -> DOWN

■   DOWN -> UP

Parameters   The following command parameters can be specified in the **create** and **set trigger** commands:

Parameter	Description
resource=*resourcename*	A change in state is detected for this particular resource.
lbstate=down	The DOWN state is detected.

Script Arguments   The trigger passes the following arguments to the script:

Argument	Description
%1	The resource name.
%2	The state of the resource; one of UP, DOWN or CLOSING.

Example   To create a trigger to activate the script "upres.scp" whenever resource "webserver_one" moves into the "DOWN" state, use the command:

```
create trigger=1 module=lb event=resstate
    resource=webserver_one lbstate=down script=upres.scp
```

Event   VIRTSTATE

Description   A change in state for a particular virtual balancer, one of:

■   UP -> CLOSING

■   CLOSING -> DOWN

■   UP -> DOWN

■   DOWN -> UP

Parameters   The following command parameters can be specified in the **create** and **set trigger** commands:

Parameter	Description
virtualbalancer=webbalancer_one	A change in state is detected for this virtual balancer.
virtstate=up	The state "UP" is detected.

Script Arguments   The trigger passes the following arguments to the script:

Argument	Description
%1	The virtual balancer name.

Argument	Description
%2	The state of the virtual balancer; one of UP, DOWN or CLOSING.

Examples To create a trigger to activate the script "downvirt.scp" whenever virtual balancer "webbalancer_one" moves into the UP state, use the command:

```
create trigger=2 module=lb event=virtstate
    virtualbalancer=webbalancer_one lbstate=up
    script=downvirt.scp
```

Event HTTPERROR

Description An HTTP response returned by a resource matches a configured HTTP error code.

Parameters The following parameter can be specified in the **create** and **set trigger** commands:

Parameter	Description
resource=webserver_one	A match in HTTP error code returned by the HTTP response is detected for this resource.

Script Arguments The trigger passes the following arguments to the script:

Argument	Description
%1	The resource name.

Example To create a trigger to activate the script "match.scp" whenever a match in HTTP error code for resource "webserver_one" is detected, use the command:

```
create trigger=3 module=lb event=httperror
    resource=webserver_one script=match.scp
```

Event LASTFAIL

Description A resource could not be found in the resource pool for completing a connection.

Parameters The following command parameter can be specified in the **create** and **set trigger** command:

Parameter	Description
respool=webfarm_one	A connection resource allocation that failed is detected for this resource pool.

Script Arguments The trigger passes the following arguments to the script:

Argument	Description
%1	The resource pool name.

Example To create a trigger to activate the script "last.scp" whenever resource pool "webfarm_one" failed to allocate a resource for a connection, use the command:

```
create trigger=4 module=lb event=lastfail respool=webfarm_one
    script=last.scp
```

Event MASTER

Description The active/passive peer of a redundant pairing becomes the master.

Parameters No options.

Script Arguments No arguments are passed to the script.

Example To create a trigger to activate the script "addeth.scp" whenever a peer becomes the MASTER of the redundant pairing, use the command:

```
create trigger=5 module=lb event=master script=addeth.scp

#Add the redundant IP interface to the firewall policy
    add firewall policy=lb interface=vlan2-1 type=public
    add firewall policy=lb rule=3 interface=vlan2-1
    action=allow proto=tcp port=80

#Add the private IP interface to the firewall policy
    add ip interface=vlan3-1 ip=192.168.1.200
    add firewall policy=lb interface=vlan3-1 type=private
    add firewall policy=lb rule=4 interface=vlan3-1
    action=allow protocol=all
    enable loadbalancer healthpings
```

Event SLAVE

Description The active/passive peer of a redundant pairing becomes the slave.

Parameters No options

Script Arguments No arguments are passed to the script.

Example To create a trigger to activate the script "slave.scp" whenever a peer becomes the SLAVE of the redundant pairing, use the command:

```
create trigger=6 module=lb event=slave script=slave.scp

#Remove the redundant IP interface from the firewall policy
    delete firewall policy=lb interface=vlan2-1
    delete firewall policy=lb rule=3

#Remove the redundant IP interface
    delete ip interface=vlan2-1

#Remove the private interface from the firewall policy
    delete ip interface=vlan3-1
    delete firewall policy=lb interface=vlan3-1
    delete firewall policy=lb rule=4
    disable loadbalancer healthpings
```

Configuration Example

The following example illustrates the steps required to configure a TCP type virtual balancer, as illustrated in Figure 50-6 on page 50-17.

Figure 50-6: Example configuration for a TCP virtual balancer
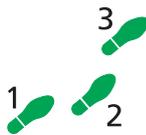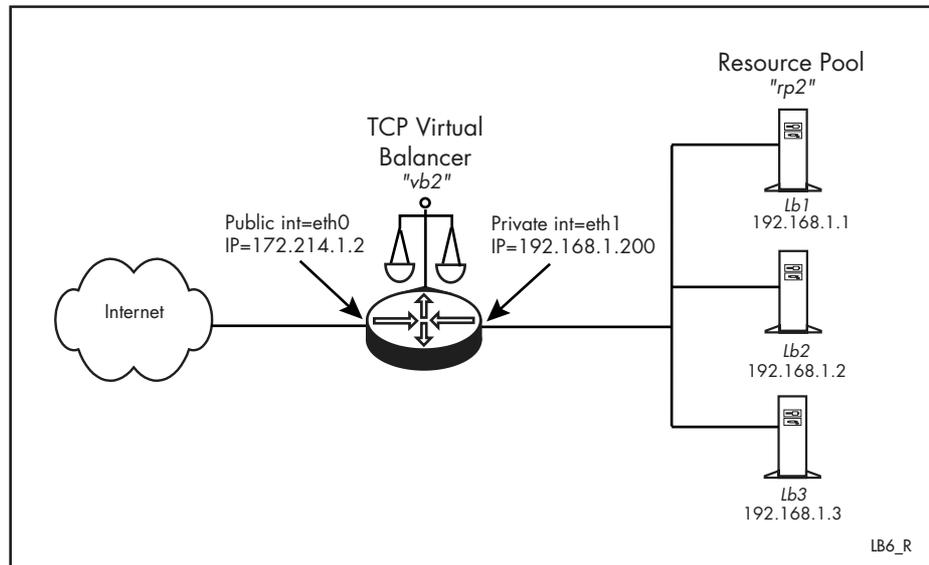
Create interfaces for the load balancer

1. **Enable IP.**

   Before creating interfaces for the load balancer, Internet Protocol must be enabled on the router. To enable IP, use the command:

   ```
   enable ip
   ```

2. **Configure the load balancer's public interface.**

   Create an interface over which the load balancer connects to the Internet, and configure its IP address and physical port. This becomes the load balancer's public interface.

   To add the interface eth0 to IP, with the IP address 172.214.1.2, use the command:

   ```
   add ip interface=eth0 ip=172.214.1.2 mask=255.255.255.0
   ```

3. **Create the load balancer's private interface.**

   Create the load balancer's private interface, and configure its IP address and physical port. This is the interface over which the resources connect to the load balancer.

   To add the interface eth1 to IP, with the IP address 192.168.1.200, use the command:

   ```
   add ip interface=eth1 ip=192.168.1.200
   ```

☞ *Before load balancing can commence, you must separately configure each resource with a default route using the load balancer's private IP address as the next hop or gateway address. You must do this from the resource itself. In this configuration example, each resource's next hop would be set to 192.168.1.200.*
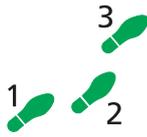
3 Configure the firewall policy for the load balancer's interfaces.

You must create a firewall policy that allows traffic to pass over the load balancer's public and private interfaces. For details on the firewall, and firewall policies, see Chapter 41, Firewall.

1. Enable the firewall.

To enable the firewall, use the command:

```
enable firewall
```

2. Configure a firewall policy for the load balancer.

To create a firewall policy for the load balancer called *LB*, use the command:

```
create firewall policy=lb
```

Set the newly created firewall's session timeouts. Low session timeouts improve the load balancer's performance. To set the firewall's timeouts for TCP, UDP and other sessions to 3 minutes, use the command:

```
set firewall policy=lb tcptimeout=180 udptimeout=180
    othertimeout=180
```

3. Add the load balancer's interfaces to the firewall policy.

To add the private interface eth1 to policy *LB*, use the command:

```
add firewall policy=lb interface=eth1 type=private
```

To add the public interface eth0 to policy *LB*, use the command:

```
add firewall policy=lb interface=eth0 type=public
```

4. Add a rule to the policy allowing traffic on the public interface.

To add a rule that allows traffic on the public interface eth0 for TCP traffic over port 80, use the command:

```
add firewall policy=lb rule=1 action=allow interface=eth0
    prot=tcp port=80
```

You don't have to add a rule for the private side because the firewall allows this traffic to pass by default.
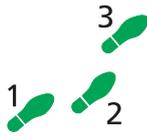
Configure the load balancer

Use the load balancer commands to configure resource pools, resources and virtual balancers. When these commands are executed, load balancing begins.

1. **Enable the load balancer.**

   To enable the load balancer, use the command:

   ```
   enable loadbalancer
   ```

2. **Create a resource pool.**

   This command adds a resource pool and specifies the selection algorithm that determines how resources in the pool are chosen to process client requests. It also specifies whether the last resource in the pool can be marked as failed.

   To add a resource pool called *rp2*, with the round robin selection algorithm, specifying that the last resource in the pool cannot be marked as failed, use the command:

   ```
   add loadbalancer respool=rp2 selectmethod=roundrobin
       fail=no
   ```

3. **Add resources to the resource pool.**

   The **add loadbalancer resource** command must be executed for each individual resource, specifying its IP address, well known port number and the name of the resource pool it belongs to.

   To add the resource named *lb1*, with the IP address 192.168.1.1, on port 80 to the resource pool *rp2*, use the command:

   ```
   add loadbalancer resource=lb1 ip=192.168.1.1 port=80
       respool=rp2
   ```

   To add the resource named *lb2*, with the IP address 192.168.1.2, on port 80 to the resource pool *rp2*, use the command:

   ```
   add loadbalancer resource=lb2 ip=192.168.1.2 port=80
       respool=rp2
   ```

   To add the resource named *lb3*, with the IP address 192.168.1.3, on port 80 to the resource pool *rp2*, use the command:

   ```
   add loadbalancer resource=lb3 ip=192.168.1.3 port=80
       respool=rp2
   ```

4. **Create a virtual balancer.**

   Create a virtual balancer that connects clients with resources in the resource pool. To create a TCP type virtual balancer called *vb2* with the public IP address 172.214.1.2 that maps to the resource pool *rp2* and creates affinity table entries, use the command:

   ```
   add loadbalancer virtualbalancer=vb2 publicip=172.214.1.2
       publicport=80 respool=rp2 type=tcp affinity=yes
   ```

5. **Enable the virtual balancer.**

   The virtual balancer must be enabled before load balancing can commence. Once this command has been executed, the virtual balancer starts distributing client requests between resources in the resource pool. To enable the virtual balancer *vb2*, use the command:

   ```
   enable loadbalancer virtualbalancer=vb2
   ```

Command Reference

This section describes the commands available to configure and monitor load balancing on the router.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page xcv of Preface in the front of this manual for details of the conventions used to describe command syntax. See Appendix A, Messages for a complete list of messages and their meanings.

You can use "lb" in place of "loadbalancer" in any of the load balancer commands. For example, you can enter **add lb resource** instead of **add loadbalancer resource**.

Before load balancing can start, you must configure the router and all interfaces on which virtual balancers listen for IP. For information on configuring IP, see Chapter 14, Internet Protocol (IP).

add loadbalancer resource

Syntax
```
ADD LOADBalancer RESource=resourcename IP=ipadd
    POrt=0..65535 [RESPool=resourcepoolname]
    [WEIght=0..4294967295]
```

where:

- *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

- *ipadd* is an IP address in dotted decimal notation.

- *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

Description   This command adds a new resource to the load balancer, and optionally allocates it to an existing resource pool. A resource can only be utilised by a virtual balancer once it is allocated to that virtual balancer's resource pool. A new resource is UP (enabled) by default. It can accept requests for its services once it has been allocated to a resource pool and at least one virtual balancer maps to that resource pool.

The RESOURCE parameter specifies the name used to identify this resource. Every resource must have a unique name. This parameter must be provided before a new resource can be created. There is no default.

The IP parameter specifies the IP address a virtual balancer uses when sending connections to the resource. This parameter must be provided before a new resource can be created.

The PORT parameter specifies the port that the resource's service runs on. When the resource is a web server, then the PORT parameter is 80; when it is an FTP server, the PORT parameter is 21. If the resource is allocated to a resource pool that is used by a route-based virtual balancer, this parameter is ignored,

and should be set to 0. When the resource is allocated to a resource pool that is associated with a SSL virtual balancer, the resource's port must be the same as the public port of the SSL virtual balancer. This applies when the data within the packets being balanced contains port information to be used by the resource application, such as a web page request on a specific port. This does not apply to HTTP type virtual balancers with SSL enabled. This parameter must be provided before a new resource can be created.

The RESPOOL parameter specifies the name of the resource pool to which this resource is to be added. The specified resource pool must have already been created using the **add loadbalancer respool** command on page 50-21 before resources can be allocated to it.

The WEIGHT parameter specifies the preference applied to this resource when the virtual balancer selects a resource for an incoming connection. The higher the weight of a resource, the more likely it will be selected for an incoming request. This parameter is used when the WLOTTERY (weighted lottery) or WLEASTCONNECT (weighted least connect) resource selection algorithms are used by the resource's resource pool. The default is 10000.

Examples   To create a new resource called "webserver_two", with an IP address of 172.16.100.1, that offers a service on port 80, with a weight of 30000, and that is allocated to the resource pool "web_farm", use the command:

```
add lb res=webserver_two ip=172.16.100.1 po=80 wei=30000
    resp=web_farm
```

Related Commands   add loadbalancer respool
delete loadbalancer resource
delete loadbalancer respool
set loadbalancer resource
set loadbalancer respool
show loadbalancer resource
show loadbalancer respool

add loadbalancer respool

Syntax   ADD LOADBalancer RESPool=*resourcepoolname* [FAIllast={YES|
NO}] [SELectmethod={ROundrobin|WLEastconnect|WLOttery|
FAstestresponse}]

where:

■   *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

Description   This command creates a new resource pool for the load balancer. A resource pool contains one or more resources that a virtual balancer can use to complete client requests. A virtual balancer maps to one resource pool.

The RESPOOL parameter specifies the unique name used to identify this resource pool. This parameter must be specified.

The FAILLAST parameter specifies whether a resource is to be marked as failed if it is the last resource remaining in a resource pool. If NO is specified, then all the resources in a resource pool except for the last resource can be marked as failed by the load balancer. If YES is specified, then all resources in a resource pool can be marked as failed. The default is NO.

The SELECTMETHOD parameter specifies the algorithm the virtual balancer uses when determining which resource to select from this resource pool. If ROUNDROBIN is specified, the virtual balancer accessing this resource pool selects the resource in the pool with the next address after the last resource used. If WLEASTCONNECT is specified, the virtual balancer selects the resource with the highest result achieved after dividing its assigned weight by the number of its current connections. A resource's weight is specified with the ADD LOADBALANCER RESOURCE command. If WLOTTERY is specified, the load balancer randomly selects a resource from its resource pool. Resources with higher weights are more likely to be selected, but if all resources have the same weight WLOTTERY provides a similar result to the ROUNDROBIN algorithm. If FASTESTRESPONSE is specified, the load balancer selects the resource that has the fastest response time based on resource health checks. The default is ROUNDROBIN.

Examples   To create a new resource pool called "web_farm" that uses the WLOTTERY selection algorithm, use the command:

```
add lb resp name=web_farm sel=wlottery
```

Related Commands   add loadbalancer resource
delete loadbalancer resource
delete loadbalancer respool
set loadbalancer respool
show loadbalancer resource
show loadbalancer respool

add loadbalancer virtualbalancer

Syntax   ADD LOADBalancer VIRTualbalancer=*virtualbalancername*
[AFFinity={YES|NO}] [COOkieheader=*cookiedetectorstring*]
[DOMAinname=*domainname*] [POLicy=*policyName*]
[PUBLICIp=*ipadd*] [PUBLICPort=0..65535]
[RESPool=*resourcepoolname*] [SSL={ON|OFf}]
[SSLKey=*key-id*] [SSLToresource={ON|OFf}] [TYpe={ROute|
TCp|SSl|HTtp}]

where:

■   *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

■   *cookiedetectorstring* is a character string 1 to 31 characters long. Valid characters are any printable character.

■   *domainname* is a character string 1 to 128 characters long representing a domain name.

■   *policy* is a character string 1 to 15 characters long. Valid characters are any printable character. *policy* is not case sensitive.

- *ipadd* is an IP address in dotted decimal notation.

- *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

- *key-id* is a decimal number from 0 to 65535.

Description   This command creates a new virtual balancer. A load balancer can have up to 32 virtual balancers. Each virtual balancer maps to resources in one resource pool. When a new virtual balancer is configured, it is always in the DOWN (disabled) state and performs no balancing operations. To move it to the UP state that balances operations, use the **enable loadbalancer virtualbalancer command on page 50-36**. If you want to create an HTTP virtual balancer, this command specifies whether it will be enabled for Secure Sockets Layer (SSL).

The VIRTUALBALANCER parameter specifies the name of the virtual balancer for subsequent operations. Each virtual balancer must have a unique name. There is no default.

The AFFINITY parameter for a route or TCP type virtual balancer specifies whether entries are made in its affinity table for each stateful connection to a resource in its resource pool. For an HTTP type balancer the AFFINITY parameter specifies whether affinity table entries are made for cookies received from clients. If an entry exists for a client IP address or cookie in a virtual balancer's affinity table, the resource associated with that entry is used first to establish the connection. This means that no selection process takes place unless the connection fails. If YES is specified, entries are made in the virtual balancer's affinity table unless it is a ROUTE type balancer with stateful operation disabled. If NO is specified, no entries are made in the affinity table under any circumstances. This means that after a connection is made to a resource, subsequent connections originating from the same IP address or with the same cookie do not necessarily use the same resource. The default is YES.

The COOKIEHEADER parameter specifies the name of the cookie that is searched for in HTTP requests and responses. The cookie name is the string that precedes the equals ("=") sign in a cookie contained in an HTTP request, or the Set Cookie header of an HTTP response. The cookie value is the string immediately after the "=" sign. Once found in an HTTP request, or any packet of an HTTP response, the value of this cookie is hashed and stored in the affinity table associated with the client's IP address. If COOKIEHEADER is not specified, then the value of the first cookie provided in a request or response is hashed and associated with the client IP address in the affinity table. This parameter is for an HTTP virtual balancer only. There is no default.

The DOMAINNAME parameter specifies the domain name that this virtual balancer provides. The virtual balancer, whose DOMAINNAME matches the host field in a received HTTP GET, processes the message when it is received on an IP address and port already occupied by a set of virtual balancers. Two virtual balancers cannot have the same DOMAINNAME, unless they have different PUBLICIP and PUBLICPORT values. DOMAINNAME can be specified only if the PUBLICIP and/or PUBLICPORT parameters are specified. This parameter is valid for HTTP type virtual balancers only. There is no default.

The POLICY parameter specifies the firewall policy that relates to the virtual balancer being created. This parameter is required for HTTP type virtual balancers only.

The PUBLICIP parameter specifies the IP address upon which this virtual balancer is to receive requests for its resource pool's services. Different virtual balancers can use the same public IP address as long as they all have different ports or a different DOMAINNAME, and no more than one is a route-based virtual balancer. Because route-based virtual balancers operate solely on IP address, and ignore the PUBLICPORT parameter, no two route-based virtual balancers can share the same public IP address. If one route-based and a number of TCP or HTTP based virtual balancers share the same public IP address, then packets received for the public IP address are processed by the route-based virtual balancer when they are not intended for one of the non-route-based virtual balancer's ports. There is no default.

The PUBLICPORT parameter specifies on which port this virtual balancer is to listen for requests for its resource pool's services. Only one virtual balancer can listen on a particular PUBLICPORT at a time unless it has a different DOMAINNAME than the other virtual balancers listening on the same PUBLICPORT and PUBLICIP. This parameter cannot be supplied for a route-based virtual balancer. There is no default.

The RESPOOL parameter specifies the name of the resource pool from which the virtual balancer allocates resources. This resource pool must already have been created with the ADD LOADBALANCER RESPOOL command. This parameter must be provided to create a virtual balancer. There is no default.

The SSL parameter enables SSL for this virtual balancer. When set to ON, the virtual balancer accepts SSL secured HTTP connections. When set to OFF, the virtual balancer accepts non-SSL connections. SSL can be activated when the TYPE parameter is set to HTTP. The default is OFF.

The SSLKEY parameter must hold an identification number that refers to a valid private key. SSLKEY is required if the SSL parameter is set to ON. There is no default.

The SSLTORESOURCE parameter informs the virtual balancer whether the resources it connects to require an SSL connection. When this parameter is set to ON, the virtual balancer tries to establish a secure connection to a resource in response to a connection from a client. If a secure connection cannot be made to the resource, the connection to the client is closed. When SSLTORESOURCE is set to OFF, the connection between the virtual balancer and the server is unsecured. The default is OFF.

The TYPE parameter specifies the type of load balancing performed by this virtual balancer. The default is TCP.

■   If ROUTE is specified, each packet received by the virtual balancer will have its next hop changed to the resource selected for the packet before it is forwarded to the resource, leaving its source and destination IP addresses unchanged. By default this type of load balancing is not stateful, so there is no guarantee that packets originating from the same IP address will be sent by the virtual balancer to the same resource even when they are part of the same TCP stream. This causes connections to fail if the load balancing devices are stateful, such as a stateful packet inspection firewall. To overcome this, stateful operation (where all packets of a TCP stream are sent to the same resource) is automatically enabled for ROUTE type virtual balancers. Packets being returned to the client by the resource have their next hop changed from the virtual balancer's resource side address to that of the next hop to the client.

■   If the TCP type is specified, the virtual balancer changes the destination address of any packet it receives to that of the resource selected for the

packet before forwarding it to the resource. Packets returned by the resource to the virtual balancer for the client have their source address changed from the resource's IP address to the virtual balancer's client side IP address before being sent to the client. This type of load balancing is for servers offering TCP services such as web serving, or SMTP. Its behaviour is stateful, so packets belonging to the same TCP stream are always sent to the same destination.

■   If the SSL type is specified, the virtual balancer changes the destination addresses of packets it receives to that of the selected destination resource before forwarding it to the resource. In this case, the affinity is between the client SSL Session ID and the resource. Otherwise, the operation is similar the TCP virtual balancer.

■   If the HTTP type is specified, the load balancer terminates the connection and opens a new connection to the resource selected for that packet. Data is passed from connection to connection via the load balancer. HTTP load balancing associates the cookie provided by the client with a resource. HTTP response packets are examined for content that may indicate an error before being forwarded to the client.

Examples   To add a new virtual balancer to the router named "web_balancer", with a public IP address of 202.36.163.22, a service on port 80, and an association with the resource pool named "web_farm", use the command:

```
add lb virt=web_balancer publici=202.36.163.22 publicp=80
    resp=web_farm
```

To add a new virtual balancer to the router named "web_balancer1", for the URL www.testsite.com, which is associated with the resource pool named "web_farm", use the command:

```
add lb virt=web_balancer1 publici=202.36.163.23 publicp=80
    doma=www.testsite.com resp=web_farm ty=http pol=lb
```

Related Commands   add loadbalancer respool
delete loadbalancer virtualbalancer
set loadbalancer virtualbalancer
show loadbalancer virtualbalancer

add loadbalancer virtualbalancer httperrorcode

Syntax   ADD LOADBalanceR VIRTualbalancer=*virtualbalancername*
HTTPErrorcode=*httperrorcodelist*

where:

■   *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

■   *httperrorcodelist* is a list of up to 32, 3-digit numbers, all greater than or equal to 500 and less than or equal to 999.

Description   This command adds a new HTTP error code to an existing virtual balancer. Individual error codes can be added to a virtual balancer only once. A virtual

balancer does not have to be disabled to have an HTTP error code added, and once added, the code is immediately applied to all content returned from the virtual balancer's resources. An HTTP error code provides the ability to scan the HTTP response returned by a resource for a particular HTTP error code. This match can then be used to determine the status of the resource. For example, if a client requested a URL that resulted in a resource returning a HTTP 505 error (HTTP version not supported), and there was a HTTP error code configured to match with that code, a trigger could be set to perform recovery actions on the resource or to notify the administrator. HTTP error codes can be added to HTTP type virtual balancers only.

The VIRTUALBALANCER parameter specifies the name of an existing virtual balancer that is receiving an HTTP error code. Each virtual balancer must have a unique name. There is no default.

The HTTPERRORCODE parameter specifies a list of up to 32, 3-digit HTTP status codes greater than or equal to 500. If the status code contained in the status line of an HTTP response returned by a resource in the virtual balancer's resource pool is greater than or equal to 500, it is searched for in the balancer's list of HTTP status codes. If the response status code is found in the HTTPERRORCODE list, the server is marked as DOWN, and all connections to the resource are closed. This parameter can only be set on HTTP type virtual balancers. There is no default.

Examples   To enable the web_balancer virtual balancer to mark any resource in its resource pool as down if it returns a HTTP response message with a status code of 503 (Service Unavailable) or 504 (Gateway Timeout), use the command:

```
add lb virt=web_balancer httpe=503, 504
```

Related Commands   **add loadbalancer virtualbalancer**
delete loadbalancer virtualbalancer httperrorcode
show loadbalancer virtualbalancer

delete loadbalancer resource

Syntax   DELete LOADBalancer RESource=*resourcename*

where *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

Description   This command deletes a resource. A resource can be deleted when it is in the DOWN state. A resource is placed in the DOWN state with the **disable loadbalancer resource** command on page 50-31. The resource can only be deleted when there are no connections to it.

The RESOURCE parameter specifies the resource that is to be deleted. This resource name must match an existing resource.

Examples   To delete the resource "main_webserver1", use the command:

```
del lb res=main_webserver1
```

Related Commands   **add loadbalancer resource**
disable loadbalancer resource
set loadbalancer resource
show loadbalancer resource

delete loadbalancer respool

Syntax   `DELete LOADBalancer RESPool=`*`resourcepoolname`*

where *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

Description   This command deletes a resource pool. A resource pool containing resources can be deleted, but the resources themselves are not deleted with this command. If the resource pool being deleted contains resources with current connections, these complete normally but no new connections are established to resources in this pool. The resource pool is deleted when all connections to its resources are closed. All resources from the deleted resource pool remain in the same state, and can be allocated to another resource pool with the **add loadbalancer resource** command on page 50-20.

The RESPOOL parameter specifies the resource pool that is to be deleted. This resource pool must match an existing resource pool or the command fails. The resource pool does not have to be emptied of resources before deletion.

Examples   To delete the resource pool "web_farm", use the command:

    del lb resp=web_farm

Related Commands   **add loadbalancer respool**
set loadbalancer respool
show loadbalancer respool

delete loadbalancer virtualbalancer

Syntax   `DELete LOADBalancer VIRTualbalancer=`*`virtualbalancername`*

where *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

Description   This command deletes an existing virtual balancer. If the resource pool associated with the virtual balancer being deleted contains resources that have connections to them, the virtual balancer allows existing connections to be completed but does not allow new ones. When there are no connections to resources in the resource pool associated with the virtual balancer, the virtual balancer is deleted. The resource pool still exists and the resources it contains remain in the state they were in before the virtual balancer was deleted.

The VIRTUALBALANCER parameter specifies the name of the virtual balancer to be deleted.

Examples   To delete the virtual balancer named "web_balancer", use the command:

```
del lb virt=web_balancer
```

Related Commands   add loadbalancer virtualbalancer
set loadbalancer virtualbalancer
show loadbalancer virtualbalancer

delete loadbalancer virtualbalancer httperrorcode

Syntax   DELete LOADBalancer VIRTualbalancer=*virtualbalancername*
         HTTPErrorcode=*httperrorcodelist*

where:

- *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

- *httperrorcodelist* is a list of up to 32, 3-digit numbers, all greater than or equal to 500 and less than or equal to 999.

Description   This command deletes an HTTP error code or codes from the specified virtual balancer.

The VIRTUALBALANCER parameter specifies the name of the virtual balancer from which the HTTP error code is deleted.

The HTTPERRORCODE parameter specifies a list of up to 32, 3-digit HTTP status codes greater than or equal to 500, that are removed from the virtual balancer's HTTPERRORCODE list. Only the codes in this list are deleted. Once removed from this list, an HTTP status code is no longer searched for in HTTP response messages returned by resources associated with the virtual balancer. This parameter is valid for HTTP type virtual balancers.

Examples   To delete the HTTP server error status code 505 (HTTP version not supported) from the virtual balancer named web_balancer, use the command:

```
del lb virt=web_balancer httpe=505
```

Related Commands   add loadbalancer virtualbalancer httperrorcode
show loadbalancer virtualbalancer

disable loadbalancer

Syntax   DISable LOADBalancer

Description   This command disables the load balancer. The load balancer can be disabled when all virtual balancers have already been disabled.

Examples   To disable the load balancer, use the command:

```
dis lb
```

Related Commands   enable loadbalancer

disable loadbalancer debug

Syntax   `DISable LOADBalancer DEBug={ALl|HTtp|FIrewall|REdun|TRace}`

Description   This command disables global debugging flags on the load balancer.

The DEBUG parameter specifies the type of debugging to disable on the configured load balancer. ALL disables every type of debugging. HTTP debugging displays information about the processing of requests by HTTP type virtual balancers. FIREWALL debugging displays information about the firewall processing of packets in relation to the load balancer's functionality. REDUN debugging displays the processing, event and state changes that occur within the redundancy protocol. TRACE debugging displays information about a packet as it is processed.

Examples   To disable the FIREWALL debug, use the command:

```
dis lb deb=firewall
```

Related Commands   **enable loadbalancer debug**

disable loadbalancer healthpings

Syntax   `DISable LOADBalancer HEALthpings`

Description   This command disables the background health check pings of all configured resources. Resources under high load sometimes ignore pings and are marked as DOWN even though the resource is still operational and can take connections. This command cannot be executed if the health check pings are currently disabled. After executing this command, reply times for resources are set to zero. Healthpings are enabled by default.

☞   *When health check pings are disabled, any resource pool that is configured to use the Fastest Response algorithm operates as the Round Robin algorithm. This is because all resources return zero response times.*

Examples   To disable the health check pings, use the command:

```
dis lb heal
```

Related Commands   **enable loadbalancer healthpings**
show loadbalancer

disable loadbalancer redundancy

Syntax   DISable LOADBalancer REDUNDancy

Description   This command disables the built-in load balancer redundancy protocol. Once disabled, the load balancer stops sending redundancy protocol messages to the configured redundancy peer. This command does not disable the redundancy protocol operation on the redundant peer, which means that the local system may still receive redundancy messages from its redundant peer when it is enabled. When this is the case, the firewall may detect messages from the redundant peer as a UDP attack. This command can be executed only when the redundancy protocol is enabled.

Examples   To disable the load balancer redundancy protocol on a load balancer where it is currently enabled, use the command:

                       dis lb redund

Related Commands   enable loadbalancer redundancy
                    set loadbalancer redundancy
                    show loadbalancer
                    show loadbalancer redundancy

disable loadbalancer resource

Syntax   DISable LOADBalancer RESource={ALl|*resourcename*}
                       [IMMEdiately]

where *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

Description   This command disables a resource either by moving it from the UP state into the DOWN state, or by moving it from the UP state into the CLOSING state and then into the DOWN state. When a resource moves into the CLOSING state all existing connections remain established. New connections can only be made to a closing resource when they are from a client with an entry in the virtual balancer's affinity table. Once all the entries in the virtual balancer's affinity table for the closing resource have expired, the resource is automatically moved into the DOWN state. In the DOWN state a resource cannot be selected by the virtual balancer to complete a connection, although any existing connections to the resource can complete. If the redundancy protocol is operational then this command performs the same action on the corresponding resource on the slave load balancer.

The RESOURCE parameter specifies the name of the resource to be disabled. This resource name must match an enabled resource or the command fails. If ALL is specified, all resources configured on the load balancer are disabled.

The presence of the IMMEDIATELY parameter specifies whether the resource moves straight from the UP state to the DOWN state. If this parameter is not specified, the resource moves from the UP to the CLOSING state, and when it has no more affinity table entries, it moves to the DOWN state. If this

parameter is specified, then the resource moves straight from the UP state to the DOWN state. There is no default.

Examples   To disable the resource named "main_webserver", use the command:

```
dis lb res=main_webserver
```

Related Commands   add loadbalancer resource
delete loadbalancer resource
enable loadbalancer resource
set loadbalancer resource
show loadbalancer resource

disable loadbalancer virtualbalancer

Syntax   DISable LOADBalancer VIRTualbalancer={ALL|
 virtualbalancername} [IMMEdiately]

where *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

Description   This command disables a previously enabled virtual balancer either by moving it from the UP state to the DOWN state, or by moving it from the UP state to the CLOSING state, and then to the DOWN state. When a virtual balancer is moved into the CLOSING state from the UP state, all connections to all resources in the virtual balancer's resource pool are allowed to complete. New connections to resources are only created when the connection originates from a client with an entry in the virtual balancer's affinity table. Once there are no more entries in the virtual balancer's affinity table, and there are no connections remaining to any resources in the virtual balancer's resource pool, the virtual balancer is moved into the DOWN state. In the DOWN state, existing connections to resources in the virtual balancer's resource pool are allowed to complete, but no new connections can be made to any of its resources. When the redundancy protocol is operational, this command performs the same action on the corresponding virtual balancer on the slave load balancer. The virtual balancer must currently be enabled for this command to succeed.

The VIRTUALBALANCER parameter is the name of the virtual balancer being disabled. If ALL is specified, then all the virtual balancers enabled on the router are disabled. Connections to all resources in their resource pools are closed.

The presence of the IMMEDIATELY parameter specifies whether the virtual balancer moves straight from the UP state to the DOWN state. If this parameter is not specified the virtual balancer moves from the UP to the CLOSING state, and when there are no more entries in its affinity table, it moves into the DOWN state. If this parameter is specified, then the virtual balancer moves straight from the UP state to the DOWN state. There is no default.

Examples   To disable the virtual balancer "web_farm", use the command:

```
dis lb virt=web_farm
```

Related Commands   add loadbalancer virtualbalancer
delete loadbalancer virtualbalancer
enable loadbalancer virtualbalancer
set loadbalancer virtualbalancer
show loadbalancer respool
show loadbalancer virtualbalancer

enable loadbalancer

Syntax   ENAble LOADBalancer

Description   This command enables the load balancer. No other load balancer commands can be executed until the load balancer has been enabled with this command.

When you enable the load balancer, the value configured for the load balancer's ORPHANTIMEOUT parameter overwrites timeout values configured for the firewall policy.

Examples   To enable the load balancer, use the command:

    ena lb

Related Commands   disable loadbalancer
set loadbalancer
set loadbalancer redundancy

enable loadbalancer debug

Syntax   ENAble LOADBalancer DEBug={ALl|HTtp|FIrewall|REdun|TRace}

Description   This command enables global debugging on the load balancer.

The DEBUG parameter specifies the type of debugging to display on the configured load balancer. ALL enables every type of debugging. HTTP debugging displays information about the processing of requests by HTTP type virtual balancers. FIREWALL debugging displays information about the firewall processing of packets in relation to the load balancer's functionality. REDUN debugging displays the processing, event and state changes that occur within the redundancy protocol. TRACE debugging displays information about a packet as it is processed.

Examples   To enable FIREWALL debug, use the command:

    ena lb deb=firewall

Related Commands   disable loadbalancer debug

enable loadbalancer healthpings

Syntax   ENAble LOADBalancer HEALthpings

Description   This command enables the background health check pings of all configured resources. This command can be executed only when the health check pings are disabled. Health check pings are enabled by default.

Examples   To enable the health check pings, use the command:

```
ena lb heal
```

Related Commands   disable loadbalancer healthpings
show loadbalancer

enable loadbalancer redundancy

Syntax   ENAble LOADBalancer REDUNDancy

Description   This command enables the built-in redundancy protocol on a load balancer. Once a load balancer is enabled with this command, the following process occurs. The system pauses for a random time between 0 and 1 seconds to ensure that simultaneous booting of two load balancers does not result in a collision. The load balancer then generates an ARP request for the configured redundant IP address (the address that will be adopted by the redundancy protocol master). If it receives a response to the ARP, the load balancer becomes the redundancy protocol slave. If the ARP times out, the load balancer becomes the redundancy protocol master.

Before load balancing can start, the interface and an associated rule must be added to the load balancer firewall policy. This can be done manually or with a trigger that adds the interface when the load balancer becomes the master.

If the load balancer is the protocol master it adopts the configured redundant IP address on its specified public interface. It can then process requests for the load balanced services provided by virtual balancers with their PUBLICIP parameters set to the redundant IP address. To set this address refer to the **add loadbalancer virtualbalancer** command on page 50-22. The first information that is transferred by the master to the slave is the contents of its affinity table. While this occurs, affinity is turned off for all virtual balancers configured on the master. The master sends heartbeat messages to the configured redundant peer at 1-second intervals to indicate its health. If the slave fails to receive three heartbeat messages from the master, it assumes that the master is not operating correctly, and it takes over as master. The master sends messages to the slave whenever a new affinity table entry is created or updated, to ensure that the affinity tables on both the master and slave are consistent. If fail-over occurs the new master can still direct client connections to the correct resource. The master sends messages to the slave whenever a resource state change has occurred. This may be if the resource health checks fail, or user has marked a resource as UP or DOWN.

Examples   To enable the built-in load balancer redundancy protocol on a load balancer, use the command:

```
ena lb redund
```

After executing this command, you must add the necessary triggers and scripts.

Related Commands   disable loadbalancer redundancy
set loadbalancer redundancy
show loadbalancer redundancy

enable loadbalancer resource

Syntax   ENAble LOADBalancer RESource={ALL|*resourcename*}

where *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

Description   This command enables a configured resource by moving it from the DOWN state into the UP state. Once UP a resource can process requests for its service. If the redundancy protocol is operational then this command performs the same action on the corresponding resource on the slave load balancer.

The RESOURCE parameter specifies the name of the existing resource to enable. The resource must currently be in the DOWN state before it can be enabled. If ALL is specified, all configured resources are enabled.

Examples   To enable the resource "main_webserver", use the command:

```
ena lb res=main_webserver
```

Related Commands   add loadbalancer resource
delete loadbalancer resource
disable loadbalancer resource
set loadbalancer resource
show loadbalancer resource

enable loadbalancer virtualbalancer

Syntax
```
ENAble LOADBalancer VIRTualbalancer={ALl|
    virtualbalancername}
```

where *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

Description
This command enables a virtual balancer by moving it from the DOWN state into the UP state. A virtual balancer must be enabled before connection requests can be accepted for services provided by its resources. Once UP, a virtual balancer can establish connections to resources in its resource pool. If the redundancy protocol is operational then this command performs the same action on the corresponding virtual balancer on the slave load balancer.

The VIRTUALBALANCER parameter specifies the name of the virtual balancer being enabled. Before it can be enabled, the virtual balancer must have been created using the **add loadbalancer virtualbalancer** command on page 50-22. If ALL is specified, then all the configured virtual balancers on the router are enabled.

Examples
To enable a virtual balancer named "web_balancer", use the command:

```
ena lb virt=web_balancer
```

Related Commands
add loadbalancer virtualbalancer
delete loadbalancer virtualbalancer
disable loadbalancer virtualbalancer
set loadbalancer virtualbalancer
show loadbalancer virtualbalancer

reset loadbalancer

Syntax
```
RESET LOADBalancer [RESource={ALl|resourcename}]
    [RESPool={ALl|resourcepoolname}]
```

where:

■ *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

■ *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

Description
This command resets the statistics for all resource pools and resources or specific ones. Resetting the statistics for these parameters resets all the object's counters (displayed in the object's SHOW output) back to zero.

The RESOURCE parameter specifies the name of the resource whose statistics are to be reset. If ALL is specified, all the resources configured on the load balancer are reset. If RESOURCE is specified, you cannot specify RESPOOL.

The RESPOOL parameter specifies the name of the resource pool whose statistics are to be reset. If ALL is specified, all the resource pools configured on the load balancer are reset. If RESPOOL is specified, you cannot specify RESOURCE.

Examples   To reset the statistics of all of the resource pools currently configured on a router, use the command:

```
reset lb resp=all
```

Related Commands   show loadbalancer resource
show loadbalancer respool

set loadbalancer

Syntax   SET LOADBalancer [AFFInitytimeout=0..4294967295]
    [CRITicalrst=0..100] [ORPHantimeout=0..65535]

Description   This command sets the global parameters of the load balancer.

The AFFINITYTIMEOUT parameter specifies the time, in seconds, after which an entry is removed from a virtual balancer's affinity table. An affinity table contains a list of source IP addresses or cookies, and the resource IP addresses they have affinity with. A new connection from an IP address or a cookie that has an entry in the affinity table automatically tries to be made with the matching resource in the affinity table. Every time a new connection is made from a source IP or cookie to a resource using an affinity table entry, the entry is refreshed. This means it is removed after it is unused for the length of time specified in the AFFINITYTIMEOUT parameter. If no connections are made using a particular affinity table entry for longer than the AFFINITYTIMEOUT period, the entry is removed from the affinity table. The default is 6000 seconds.

The CRITICALRST parameter specifies the maximum percentage of the TCP messages received from a server in a resource pool that can be TCP RST messages. If the percentage of RST messages in the total number of messages sent to a server is greater than CRITICALRST, the server is marked unavailable and cannot be selected for a connection. The default is 50.

The ORPHANTIMEOUT parameter specifies the time, in seconds, before a connection is declared to be orphaned and is closed. A connection becomes orphaned when it is open but has had no data transmitted over it for a longer period than that specified in this parameter. A connection is closed as soon as it becomes an orphan. The default is 300 seconds.

☞   *The value specified for the ORPHANTIMEOUT parameter overwrites any values specified for the firewall policy's UDPTIMEOUT, TCPTIMEOUT and OTHERTIMEOUT parameters in the **set firewall policy** command on page 41-73 of Chapter 41, Firewall.*

To set the AFFINITYTIMEOUT parameter to 10000 seconds use the command:

```
set lb affi=10000
```

Related Commands   enable loadbalancer
show loadbalancer

set loadbalancer redundancy

Syntax   SET LOADBalancer REDUNDancy [LISTenport=0..65535]
[PEERip=*ipadd*] [PUBLicint=*interface*] [REDUNIp=*ipadd*]
[REDUNMask=*ipadd*]

where:

- *ipadd* is an IP address in dotted decimal notation.

- *interface* is a valid interface name.

Description   This command configures the built-in redundancy protocol. Redundancy
protocol parameters can only be set when the redundancy protocol is disabled.

All parameters that can be configured by this command must be set before the
redundancy protocol can be enabled with the **enable loadbalancer** command
on page 50-33.

LISTENPORT specifies the UDP port that the load balancer listens on for
redundancy messages from its peer, and is the source of messages sent to its
peer. There is no default.

PEERIP specifies the IP address that the load balancer sends all redundancy
protocol messages to if it is the protocol master, and should receive
redundancy protocol messages from if it is the protocol slave. There is no
default.

PUBLICINT specifies the public interface on which the load balancer adds an
IP address alias for the specified redundant IP address when the load balancer
becomes the master. This parameter must be a physical interface (e.g. ppp1),
not a logical interface (e.g. ppp1-8). It must also be specified as a public
interface in the firewall policy used by the load balancer. If the interface
specified is not a public interface of this policy, the redundancy protocol does
not operate correctly. Valid interfaces are:

- eth (e.g. eth0)

- PPP (e.g. ppp0)

- VLAN (e.g. vlan1)

The interface must already exist. To see a list of all currently available
interfaces, use the **show interface** command on page 7-66 of Chapter 7,
Interfaces. There is no default.

REDUNIP is the IP address that is backed up by the redundancy protocol. This
is the IP address used by the master for load balancing. The address must be
the same as the PUBLICIP of the virtual balancer used for the redundancy
protocol. When the load balancer becomes the redundancy protocol master,
this IP address becomes a new IP address alias on the interface specified in the
PUBLICINT parameter. The logical interface that is created for the redundant
IP address varies depending on the number of other logical interfaces on the
public interface. For example, if there are four other logical interfaces on the

specified public interface, the new redundancy alias is automatically added as the fifth. There is no default.

REDUNMASK specifies the netmask of the IP alias added for the redundant IP address. This must be consistent with the intended public network of the load balancer. There is no default.

Examples   To set the public interface of the redundancy protocol to vlan1, use the command:

```
set lb publ=vlan1
```

Related Commands   disable loadbalancer redundancy
enable loadbalancer
enable loadbalancer redundancy
show loadbalancer redundancy

set loadbalancer resource

Syntax   SET LOADBalancer RESource=*resourcename* [IP=*ipadd*]
    [POrt=0..65535] [RESPool=*resourcepoolname*]
    [WEIght=0..4294967295]

where:

■   *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

■   *ipadd* is an IP address in dotted decimal notation.

■   *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

Description   This command sets the configuration of a resource. When the resource is in the UP state, the WEIGHT parameter can be changed. No parameters can be changed while the resource is in the CLOSING state. Every parameter can be changed when the resource is in the DOWN state. Changes to a resource parameter take effect the next time the resource is used for a connection.

The RESOURCE parameter specifies the name of the resource to be modified.

The IP parameter specifies the IP address used by a virtual balancer to send incoming connections to the resource. This parameter can be changed when a resource is DOWN. There is no default.

The PORT parameter specifies the port used by a virtual balancer to access the resource's services. If the resource is a web server, then the PORT parameter would be 80, and if it is an FTP server the PORT parameter would be 21. If the resource is allocated to a resource pool that is used by a route based virtual balancer, this parameter is ignored and should be set to 0. This parameter must be provided before a new resource can be created. This parameter can be changed when a resource is DOWN. There is no default.

The RESPOOL parameter specifies the resource pool associated with this resource. If this is specified, then no other parameters except for the resource

name can be specified. A resource must be DOWN and have no current connections for it to be reallocated to another resource pool. There is no default.

The WEIGHT parameter specifies the preference the load balancer applies to this resource when selecting one for an incoming connection. The higher the weight of a resource, the more likely it will be selected for an incoming request. This parameter is used when the weighted lottery or weighted least connection resource selection algorithms are used. It is not used for the round robin or fastest response algorithms. This parameter can be changed while a resource is UP or DOWN, but changes do not close connections or modify affinities; changes affect new resource selections from the resource pool. There is no default.

Examples   To shut down the resource "main_webserver", use the command:

```
set lb res=main_webserver
```

Related Commands   add loadbalancer resource
delete loadbalancer resource
show loadbalancer resource

set loadbalancer respool

Syntax   SET LOADBalancer RESPool=*resourcepoolname* [FAIllast={YES|
NO}] [SELectmethod={ROundrobin|WLEastconnect|WLOttery|
FAstestresponse}]

where *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

Description   This command sets the configuration of a resource pool. Changes to the configuration of the resource pool take effect with the resource pool's next connection.

The RESPOOL parameter specifies the name of the resource pool to be modified.

The FAILLAST parameter specifies whether a resource is marked as failed when it is the last resource remaining in a resource pool. If NO is specified, then resources in a resource pool except for the last resource can be marked as failed by the load balancer. If YES is specified, then all resources in a resource pool can be marked as failed. The default is NO.

The SELECTMETHOD parameter specifies the algorithm the load balancer uses when selecting resources from this pool to complete a client request. See the **add loadbalancer respool** command on page 50-21 for details about selection algorithms. The default is ROUNDROBIN.

Examples   To set the selection algorithm of the "web_farm" resource pool to WEIGHTEDLOTTERY, use the command:

```
set lb resp=web_farm sel=wlottery
```

Related Commands   add loadbalancer respool
delete loadbalancer respool
show loadbalancer respool

set loadbalancer virtualbalancer

Syntax   SET LOADBalancer VIRTualbalancer=*virtualbalancername*
[AFFinity={YES|NO}] [COOkieheader=*cookiedetectorstring*]
[DOMAinname=*domainname*] [POLicy=*policyName*]
[PUBLICIp=*ipadd*] [PUBLICPort=0..65535]
[RESPool=*resourcepoolname*] [SSL={ON|OFf}]
[SSLKey=*key-id*] [SSLToresource={ON|OFf}] [TYpe={ROute|
TCp|SSl|HTtp}]

where:

- *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

- *cookiedetectorstring* is a character string 1 to 31 characters long. Valid characters are any printable character.

- *domainname* is a character string 1 to 128 characters long representing a domain name.

- *policy* is a character string 1 to 15 characters long. Valid characters are any printable character. *policy* is not case sensitive.

- *ipadd* is an IP address in dotted decimal notation.

- *resourcepoolname* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcepoolname* is not case sensitive.

Description   This command sets the configuration of an existing virtual balancer. Only a disabled virtual balancer can be modified with this command. Changes made to a virtual balancer with this command take effect when it is re-enabled using the **enable loadbalancer virtualbalancer** command on page 50-36.

The VIRTUALBALANCER parameter specifies the name of the virtual balancer being modified.

☞   *Once created, the name of a virtual balancer cannot be modified with this command. To do this, the virtual balancer must be deleted using the **delete loadbalancer virtualbalancer** command on page 50-27, and then re-created with the new name.*

For a route or TCP type virtual balancer, the AFFINITY parameter specifies whether entries are made in its affinity table for each stateful connection to a resource in its resource pool. For an HTTP type balancer, this parameter specifies whether entries are made in its affinity table for cookies received from clients. The default is YES.

The COOKIEHEADER parameter specifies the text string in a packet's header that the virtual balancer examines to determine whether the packet contains a

cookie. This parameter can only be set for an HTTP type virtual balancer. There is no default.

The DOMAINNAME parameter specifies the domain name for which the virtual balancer handles requests. This must be set for an HTTP type virtual balancer. No two virtual balancers can have the same DOMAINNAME unless they use different public IP addresses and public ports. There is no default.

The POLICY parameter specifies the firewall policy that relates to the virtual balancer being created. This parameter is required for HTTP type virtual balancers only.

The PUBLICIP parameter specifies the IP address on which this virtual balancer receives requests for its resource pool's services. This parameter can be the same for other virtual balancers as long as they all have different PUBLICPORT parameters. There is no default.

The PUBLICPORT parameter specifies on which port this virtual balancer listens for requests for its resource pool's services. If more than one virtual balancer uses the same PUBLICIP, they must all use different PUBLICPORT parameters. There is no default.

The RESPOOL parameter specifies the name of the resource pool that the virtual load balancer uses to allocate resources for a connection. Only one virtual balancer at a time can be associated with a particular resource pool. This parameter must specify an existing resource pool. There is no default.

The SSL parameter enables SSL for this virtual balancer. When set to ON, the virtual balancer accepts SSL secured HTTP connections. When set to OFF, the virtual balancer accepts non-SSL connections. SSL must be activated when the TYPE parameter is set to HTTP. The default is OFF.

The SSLKEY parameter must hold an identification number that refers to a valid private key. SSLKEY is required when the SSL parameter is set to ON. There is no default.

The SSLTORESOURCE parameter informs the virtual balancer whether the resources it connects to require an SSL connection. When this parameter is set to ON, the virtual balancer tries to establish a secure connection to a resource in response to a connection from a client. When a secure connection cannot be made to the resource, the connection to the client is closed. When SSLTORESOURCE is set to OFF, the connection between the virtual balancer and the server is unsecured. The default is OFF.

The TYPE parameter specifies the type of load balancing this virtual balancer performs. Changing the TYPE of an existing virtual balancer can have major implications on the amount of resources it uses and its overall performance. For details see the **add loadbalancer virtualbalancer** command on page 50-22. The default is TCP.

Examples   To change the type of the "web_balancer" virtual balancer to ROUTE from TCP, use the command:

```
set lb virt=web_balancer ty=route
```

Related Commands   **add loadbalancer virtualbalancer**
delete loadbalancer virtualbalancer
disable loadbalancer virtualbalancer
enable loadbalancer virtualbalancer
show loadbalancer virtualbalancer

show loadbalancer

Syntax   SHow LOADBalancer

Description   This command displays information about the general configuration and status for all virtual balancers (Figure 50-7 on page 50-43, Table 50-1 on page 50-43).

Figure 50-7: Example output from the **show loadbalancer** command

```
Global Load Balancer Configuration
-----------------------------------------------------------
Status ........................... ENABLED
Affinity Timeout ................. 60000s
Orphan Timeout ................... 300s
Critical RST ..................... 50%
Total Resources .................. 6
Total Resource Pools ............. 2
Total Virtual Balancers .......... 1
Current Connections .............. 1
Health Check Pings................ ENABLED

Affinity List Populations
   Route affinities ............... 0
   TCP affinities ................. 334
   SSL affinities ................. 0
   HTTP affinities ................ 2445
-----------------------------------------------------------
```

Table 50-1: Parameters in the output of the **show loadbalancer** command

Parameter	Meaning
Status	Whether the Load Balancer is enabled.
Affinity Timeout	Length of time in seconds that a resource is associated with a source IP address or cookie in the affinity table after the connection has been closed. If a new connection is attempted from the source IP address or with the cookie that has an entry in the affinity table, the resource in the entry is tried first. When the connection succeeds, the affinity table entry is refreshed so it persists again for at least the affinity timeout once it is closed.
Orphan Timeout	Length of time in seconds that a connection can exist without having any data transmitted on it. After this period, the connection is declared an orphan and is closed.
Critical RST (%)	Percentage of total messages received from a resource that can be TCP RST messages. Once a resource's Critical RST percentage has been exceeded, it is moved into the DOWN state and no connections can be made to it until it is reset with the ENABLE LOADBALANCER RESOURCE command.
Total Resources	Total number of resources configured on the load balancer.
Total Resource Pools	Total number of resource pools configured on the load balancer.
Total Virtual Balancers	Total number of virtual balancers configured on the load balancer.

Table 50-1: Parameters in the output of the **show loadbalancer** command (continued)

Parameter	Meaning
Current Connections	Total number of current connections to all resources in every resource pool.
Health Check Pings	Whether the background resource health check pings are enabled or disabled.
Affinity list populations	The following entries relate to different virtual balancer's affinity list counters.
Route affinities	Number of affinity entries owned by route-based virtual balancers.
TCP affinities	Number of affinity entries owned by TCP virtual balancers.
SSL affinities	Number of affinity entries owned by SSL virtual balancers.
HTTP affinities	Number of affinity entries owned by HTTP virtual balancers.

Examples   To display the current configuration and status of the load balancer, use the command:

```
sh lb
```

Related Commands   set loadbalancer

show loadbalancer affinity

Syntax   SHow LOADBalancer AFFinity {[CLIentip=*ipadd*]|
[RESource=*resourcename*]|
VIRTualbalancer=*virtualbalancername*]}

where:

■   *ipadd* is an IP address in dotted decimal notation.

■   *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

■   *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

Description   This command displays the affinity tables for all virtual balancers, or for a specific virtual balancer, resource or client IP address (Figure 50-8 on page 50-45, Table 50-2 on page 50-45). The affinity table contains source IP addresses and cookies, and the resources that will be tried first for a new connection. Entries are created in this table whenever a connection is made between source address or cookie and a resource. Entries are also made if a source address or cookie was in the affinity table, but is not connected to the same resource.

Although each type of virtual balancer has its own affinity table, the **show loadbalancer affinity** command incorporates the affinity tables for all types of virtual balancer in its output.

The CLIENTIP parameter specifies the client IP address whose affinity table is to be displayed. If this parameter is specified, no other parameter can be specified. There is no default.

The RESOURCE parameter specifies the name of the existing resource whose affinity table is to be displayed. If this parameter is specified, no other parameter can be specified. There is no default.

The VIRTUALBALANCER parameter specifies the virtual balancer whose affinity table is displayed when a virtual balancer is provided. If this parameter is specified, no other parameter can be specified. There is no default.

Figure 50-8: Example output from the **show loadbalancer affinity** command

```
Virtual Balancer Affinity Tables
----------------------------------
  Number of entries: 2213

Name
  IP/cookie                      Resource       Expiry
---------------------------------------------------------
TCP_balancer
  172.204.1.1                    target56       98s
  172.204.1.1                    target45       71s
HTTP_balancer
  websiteEadz58                  WebServer58    4000s
  websiteRtmz23                  WebServer23    1678s
  websiteElAn24                  Webserver1     200s
  websiteMwze498                 WebServer498   8s
ROUTE_balancer
  172.204.1.1                    Destination1   8846s
  172.204.1.1                    Destination2   376s
  172.204.1.1                    Destination3   30s
SSL_Balancer
  ID_10                          SecureSvr9     564s
  ID_11                          SecureSvr1     20s
  ID_12                          SecureSvr2     1s
  .
  .
  .
---------------------------------------------------------
```

Table 50-2: Parameters in the output of the **show loadbalancer affinity** command

Parameter	Meaning
Name	Name of the virtual balancer whose affinity table is being displayed.
IP/cookie	Either the IP address of the client, or the cookie that has affinity to the given resource.
Resource	Name of the resource that the client IP/cookie has affinity with.
Expiry	The number of seconds left before this entry expires and is removed from the affinity table. An entry is refreshed and the expiry time is reset, every time a connection is made from a client with an entry in the affinity table to its resource. If the connection fails and must be made to another resource, the current entry is removed and an affinity table entry is added for the new connection.

Examples   To display the affinity tables for all virtual balancers, use the command:

```
sh lb aff
```

Related Commands   **add loadbalancer virtualbalancer**
show loadbalancer resource
set loadbalancer virtualbalancer

show loadbalancer connections

Syntax   SHow LOADBlancer CONnections[=*virtualbalancername*]

where *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

Description   This command displays information about all of the TCP connections currently open on a virtual balancer (Figure 50-9 on page 50-46, Table 50-3 on page 50-46).

The CONNECTIONS parameter specifies the name of the virtual balancer whose connection status is displayed. If no name is specified, then connections to all the currently configured virtual balancers are displayed. There is no default.

Figure 50-9: Example output from the **show loadbalancer connections** command.

```
Virtual Balancer Connection

Balancer Name
  Client IP        Port     Resource
-------------------------------------------------------------
web_balancer
  212.72.1.246     45013    main_webserver
  .
  .
-------------------------------------------------------------
```

Table 50-3: Parameters in the output of the **show loadbalancer connections** command.

Parameter	Meaning
Balancer Name	Connections to this virtual balancer are displayed
Client IP	IP address that initiated the connection.
Port	Port on the initiating system that represents the other end of the connection.
Resources	Virtual balancer makes connections to this resource.

Examples   To display the status of the connections to the virtual balancer named "web_balancer", use the command:

```
sh lb con=web_balancer
```

Related Commands   **show loadbalancer affinity**

show loadbalancer redundancy

Syntax   SHow LOADBalancer REDUNDancy

Description   This command displays the current status and configuration of the built-in load balancer redundancy protocol (Figure on page 50-47, Table 50-4 on page 50-47).

Figure 50-10: Example output from the **show lb redundancy** command

```
Load Balancer Redundancy Information
--------------------------------------------------

Redundancy Enabled .................. YES
Peer Type ........................... MASTER
Peer Present ........................ YES
Current State ....................... Normal Operations
Last Event .......................... Heartbeat timer expired
Redundancy Port ..................... 5555
Affinity Transfer ................... NO
Redundant IP address ................ 202.20.92.45
Redundant IP Mask ................... 255.255.255.0
Peer IP address ..................... 192.168.1.2
Public Interface .................... ppp1

Message Counters

Heartbeats Received ................. 2389
Heartbeats Sent ..................... 0
Affinity Entries Received .......... 0
Affinity Entries Sent .............. 21889
Resource Updates Received .......... 0
Resource Updates Sent .............. 12
Slave Registrations Received ....... 1
Slave Registrations Sent ........... 0
Virtual Balancer Updates Received ... 0
Virtual Balancer Updates Sent ....... 3
--------------------------------------------------
```

Table 50-4: Parameters in the output of the **show loadbalancer redundancy** command

Parameter	Meaning
Redundancy Enabled	Whether the built-in load balancer redundancy protocol is currently enabled. One of YES or NO.
Peer Type	Type of redundant peer for this local load balancer, one of master or slave. The default type of a redundant peer is master when the protocol is first enabled, or disabled entirely.
Peer Present	Whether there is a redundant peer present. On the master, this indicates whether a slave is present. On the slave, this indicates whether a master is present.
Current State	Current operation state of the redundancy protocol. If the load balancer is a master then this is one of Start or Normal. If the load balancer is a slave then this is one of Start, Listen, or Stop.

Table 50-4: Parameters in the output of the **show loadbalancer redundancy** command

Parameter	Meaning
Last Event	Last redundancy protocol event processed by the load balancer. The possible events are: No ARP response received; New affinity entry created; Resource down; Heartbeat timer expired; ARP response received; Affinity entry received; Resource state received; and slave register received.
Redundancy Port	UDP port number that the load balancer is listening on for redundancy messages, and is sending redundancy messages out on.
Affinity Transfer	Whether the load balancer is in the process of transferring its affinity table to the redundant peer. This occurs when a load balancer is the protocol master and first detects the presence of a slave, via the receipt of a slave registration message. When YES is displayed, all virtual balancers on the load balancer have affinity turned off. A slave cannot transfer its affinity table to a master.
Redundant IP Address	IP address that the load balancer provides redundancy for. If this is not set, 0.0.0.0 is displayed. This represents the IP address that is added as an alias to the specified public interface when the load balancer is the protocol master.
Redundant IP Mask	Netmask of the redundant IP address. If this is not set, 255.255.255.0 is displayed. This represents the netmask of the IP alias to be added to the public interface when the load balancer is the protocol master.
Peer IP Address	IP address of the load balancer's redundant peer. This is IP address should be on a subnet associated with a private interface. It is the IP address on which the load balancer sends and receives redundancy messages.
Public Interface	Public interface that receives the redundant IP address as an alias when the load balancer is the protocol master.
Heart Beats Received	Number of heartbeat messages received by the load balancer.
Heart Beats Sent	Number of heartbeat messages sent by the load balancer.
Affinity Entries Received	Number of affinity table entries received by the load balancer.
Affinity Entries Sent	Number of affinity table entries sent by the load balancer.
Resource Updates Received	The number of resource state change messages received by the load balancer.
Resource Updates Sent	Number of resource state change messages sent by the load balancer.
Slave Registrations Received	Number of slave registration messages received by the load balancer.
Slave Registrations Sent	Number of slave registration messages sent by the load balancer.
Virtual Balancer Updates Received	Number of virtual balancer state change messages received by the load balancer.
Virtual Balancer Updates Sent	Number of virtual balancer state change messages sent by the load balancer.

Examples To show the current status and configuration of the built-in redundancy protocol, use the command:

```
sh lb redund
```

Related Commands disable loadbalancer redundancy
enable loadbalancer redundancy
set loadbalancer redundancy

show loadbalancer resource

Syntax SHow LOADBalancer RESource[=*resourcename*]

where *resourcename* is a character string 1 to 15 characters long. Valid characters are any printable character. *resourcename* is not case sensitive.

Description This command displays information about either all resources configured for the load balancer (Figure 50-11 on page 50-49, Table 50-5 on page 50-49), or detailed information about a particular resource if a resource name is specified (Figure 50-12 on page 50-50, Table 50-6 on page 50-50).

The RESOURCE parameter specifies the specific resource whose details are shown. If no resource name is provided, then less specific information about all resources is displayed.

Figure 50-11: Example output from the **show loadbalancer resource** command

```
Load Balancer Resources

Name            IP              State    Allocated
-------------------------------------------------------------
main_resource   172.16.100.1    UP       YES
backup resource 172.16.100.2    DOWN *   YES
backup res2     172.16.100.3    DOWN     YES
backup res3     172.16.100.4    CLOSING  YES
-------------------------------------------------------------
```

Table 50-5: Parameters in the output of the **show loadbalancer resource** command

Parameter	Meaning
Name	Name of the resource
IP	IP address that a virtual balancer uses to access the resource
State	Current state of the resource, either UP, DOWN, or CLOSING. A "*" after the state indicates that this resource is allocated to a resource pool where all resources are DOWN or CLOSING, and the FAILLAST parameter is set to FALSE. The resource is still processing traffic but is not operating correctly.
Allocated	Whether the resource is currently allocated to a resource pool.

Figure 50-12: Example detailed output of the **show loadbalancer resource** command

```
Load Balancer Resource Configuration
-----------------------------------------------------------
Name .................. backup resource
IP .................... 172.16.100.2
Port .................. 80
State ................. DOWN - still being selected
Weight ................ 10000
Total Connections ..... 33000
Current Connections ... 24
Refused Connections ... 2
RST% .................. 4
Instantaneous RST% .... 1
Response Time ..........16ms
Resource Pool ......... web_farm
-----------------------------------------------------------
```

Table 50-6: Parameters in the output of the **show loadbalancer resource** command

Parameter	Meaning
Name	The name of the resource
IP	The IP address that a virtual balancer uses to access the resource
Port	The port on which the resource's service is accessed.
State	The current state of the resource, one of UP, DOWN, or CLOSING
Weight	The weight that the virtual balancer applies to this resource when selecting resources for a connection using the WLEASTCONNECT or WLOTTERY selection algorithms.
Total Connections	The total number of successful connections that have been made to this resource while in the UP state.
Current Connections	The total number of connections currently made to the resource.
Refused Connections	The total number of connections that have been refused by this resource while it has been in the UP state.
RST%	The percentage of messages sent to the resource that were TCP RST messages. This is used to determine whether the resource is working correctly.
Instantaneous RST%	The percentage of the last 100 TCP connection attempts to the resource that have resulted in a RST message being sent back by the resource. This determines whether the resource is working correctly.
Response Time	The current average reply time of resources to connection requests. This is relevant when this resource's resource pool uses the FASTESTRESPONSE selection algorithm.
Resource Pool	The resource pool that this resource is currently assigned to.

Examples   To display detailed information about the resource "main_webserver", use the command:

```
sh lb res=main_webserver
```

Related Commands   add loadbalancer resource
                         delete loadbalancer resource
                         set loadbalancer resource

show loadbalancer respool

Syntax   SHow LOADBalancer RESPool[=*resourcepoolname*]

where *resourcepoolname* is a character string 1 to 15 characters long. Valid
characters are any printable character. *resourcepoolname* is not case sensitive.

Description   This command displays general information about all of the load balancer's
resource pools (Figure 50-13 on page 50-51, Table 50-7 on page 50-51). If a
resource pool name is specified, this command shows detailed information
about the given resource pool (Figure 50-14 on page 50-52, Table 50-8 on
page 50-52).

The RESPOOL parameter displays general information about all the resource
pools currently configured. If the name of a resource pool is specified, then this
command displays detailed information about that resource pool.

Figure 50-13: Example general output from the **show loadbalancer respool** command

```
Load Balancer Resource Pools

Name             Resources    Connections
-------------------------------------------------------------
web_farm         2            24
-------------------------------------------------------------
```

Table 50-7: Parameters in the output of the **show loadbalancer resource** command

Parameter	Meaning
Name	Name of a resource pool.
Resources	Total number of resources currently allocated to a resource pool.
Connections	Total number of connections that currently exist to all the resources in a resource pool.

Figure 50-14: Example detailed output from the **show loadbalancer respool** command

```
Load Balance Resource Pool Configuration
----------------------------------------------------------
Name ........................ web_farm
Selection Algorithm ......... WLEASTCONNECT
Fail On Last ................ YES
Total Connections ........... 43002
Failed Connections .......... 6
Total UDP datagrams.......... 0
Resources:
  main_webserver            UP
  backup_server             DOWN - still being selected
----------------------------------------------------------
```

Table 50-8: Parameters in detailed output of the **show loadbalancer respool** command

Parameter	Meaning
Name	Name of the resource pool.
Selection Algorithm	Selection algorithm that is employed by this resource pool when a resource has been selected for a connection. The options are ROUNDROBIN, WLEASTCONNECT, WLOTTERY, and FASTESTRESPONSE.
Fail on Last	Whether a resource is failed when it is the last resource in the resource pool.
Total Connections	Total number of successful connections that have been made to the resources in the resource pool over the lifetime of the resource pool. This does not include retried connections, only connections that succeeded the first time.
Failed Connections	Total number of connections that have failed to resources in the resource pool over the lifetime of the resource pool.
Total UDP datagrams	Total number of UDP packets that have passed through the resource pool.
Resources	List of the names of all resources that are currently allocated to the resource pool. "Still being selected" indicates that this resource is allocated to a resource pool where all resources are DOWN or CLOSING, and the FAILLAST parameter is set to FALSE. The resource is still processing traffic but is not operating correctly.

Examples   To display general information about all of the resource pools configured on a router, use the command:

```
sh lb resp
```

Related Commands   ADD LOADBALANCER RESPOOL
DELETE LOADBALANCER RESPOOL
SET LOADBALANCER RESPOOL

show loadbalancer virtualbalancer

Syntax   SHow LOADBalancer VIRTualbalancer[=*virtualbalancername*]

where *virtualbalancername* is a character string 1 to 15 characters long. Valid characters are any printable character. *virtualbalancername* is not case sensitive.

Description   This command displays general information about all the configured virtual balancers (Figure 50-15 on page 50-53, Table 50-9 on page 50-53). If a virtual balancer name is specified, detailed information about that virtual balancer is shown (Figure 50-16 on page 50-54, Table 50-10 on page 50-54).

If no name is given, the VIRTUALBALANCER parameter specifies that general information about all configured virtual balancers is shown. If a virtual balancer's name is provided, then the command displays detailed information about that virtual balancer.

Figure 50-15: Example general output from the **show loadbalancer virtualbalancer** command

```
Load Balancer Virtual Balancers

Name                Public IP/DOMAIN                  Port
-------------------------------------------------------------
web_balancer        202.36.163.22                    80
web_balancer1       www.testsite.com                 -
-------------------------------------------------------------
```

Table 50-9: Parameters in general output of the **show loadbalancer virtualbalancer** command

Parameter	Meaning
Name	Name of a virtual balancer.
Public IP/URL	IP address or the URL on which the virtual balancer receives requests for the service provided by its resource pool.
Port	Port on which the virtual balancer listens for requests received for the Public IP address. This parameter is not valid if a virtual balancer is associated with a URL.

Figure 50-16: Example detailed output from the **show loadbalancer virtualbalancer** command

```
Virtual Balancer Configuration
-------------------------------------------------------------
Name ........................ web_balancer
Public IP ................... 202.36.163.22
Public Port ................. 80
State ....................... UP
Resource Pool ............... web_farm
Type ........................ TCP
Affinity.................... YES
Domain...................... -
Cookie...................... -
HTTP Error Codes............. -
SSL ........................ OFF
SSL Key .................... -
SSL To Resource ............ OFF
-------------------------------------------------------------
```

Table 50-10: Parameters in detailed output of the **show loadbalancer virtualbalancer** command

Parameter	Meaning
Name	Name of a virtual balancer.
Public IP	IP address on which the virtual balancer receives requests for the service provided by its resource pool.
Public Port	Port on which the virtual balancer receives requests for the service provided by the resource pool associated with it.
State	Whether the virtual balancer is up, down, or closing.
Resource Pool	Name of the resource pool associated with the virtual balancer.
Type	Whether the type of load balancing that this virtual balancer currently performs is TCP, HTTP, SSL, or ROUTE.
Affinity	Whether entries are made in a virtual balancer's affinity table for each stateful connection to a resource in its resource pool if it is a ROUTE or TCP type balancer. For an HTTP type balancer, this specifies whether entries are made in the virtual balancer's affinity table for cookies received from clients.
Domain	Domain name for which the virtual balancer handles requests.
Cookie	Name of the cookie for which the virtual balancer searches in the first packet of a request from a client, or in any response packet so its value can be used in the affinity table.
HTTP Error codes	List of HTTP server error status codes for this virtual balancer. When a resource associated with this virtual balancer returns an HTTP response with a status code matching one of those listed, that resource is marked as DOWN.
SSL	When on, the virtual balancer accepts only SSL connections. When off, it accepts only non-SSL connections.
SSL key	Identification number of the private key used for encryption.

Table 50-10: Parameters in detailed output of the **show loadbalancer virtualbalancer** command (continued)

Parameter	Meaning
SSL to resource	When on, the virtual balancer tries to establish a secured SSL connection to the server in response to a connection from a client. If the connection fails, the connection to the client is closed. When off, the connection to the server is unsecured.

Examples   To display the configuration of the virtual balancer "web_balancer", use the command:

```
sh lb virt=web_balancer
```

Related Commands   add loadbalancer virtualbalancer
delete loadbalancer virtualbalancer
disable loadbalancer virtualbalancer
enable loadbalancer virtualbalancer
set loadbalancer virtualbalancer