FortiBalancer 8.4 User Guide

User Guide

FortiBalancer 8.4 User Guide

18 March 2013

1st Edition

Technical Documentation	http://help.fortinet.com
Knowledge Base	http://kb.fortinet.com
Forums	https://support.fortinet.com/forum
Customer Service & Support	https://support.fortinet.com
Training Services	http://training.fortinet.com
FortiGuard Threat Research & Response	http://www.fortiguard.com
Document Feedback	Email: techdocs@fortinet.com

Table of Contents

Chapter 1 Initial System Setup & Configuration

1.1 Overview

This section will outline the initial connection, basic setup and configuration of the FortiBalancer appliance. The easy to follow setup steps are introduced below.

1.1.1 Connecting to FortiBalancer

There are three ways to connect to the FortiBalancer appliance in order to begin the configuration:

- Console (recommended)
- SSH
- Web UI

1.1.1.1 Console Connection

If you choose the console connection, first connect the console cable (supplied) to the System Console Port on the FortiBalancer appliance, and then set up your console as follows:

Table 1-1 Console Setup

Setting	Value
Emulation	VT 100
Baud	9600
Number of Bits	8
Parity	No
Stop Bits	1
Flow Control	No

Open a connection between the console and the FortiBalancer appliance. Once this connection is opened, users will see the FortiBalancer appliance prompt and may begin the configuration process.

1.1.1.2 SSH Connection

Once the IP parameters are configured and the SSH service is activated, the FortiBalancer appliance is prepared for custom configuration. You may access the command line interface (CLI) using SSH connection. Below gives an example.

Note: If you require SSH software for Windows, Mac OS X or UNIX, it is available on-line at http://www.openssh.com.

To establish an SSH connection:

➢ Step 1 Run the SSH program on your workstation

```
>> # ssh admin@10.3.55.251
```

10.3.55.251 is the FortiBalancer appliance’s IP address.

➢ Step 2 After you establish a connection, the FortiBalancer appliance will ask you for a privilege password.

```
>> # ssh admin@10.3.55.251
>> # admin@10.3.55.251's password:
```

Upon the first startup, the user will be prompted for login username and password. The default username is “**admin**”, and the default password is “**admin**”.

Note: You must have the IP information setup and basic network connectivity in order to access the box through SSH.

1.1.1.3 Web UI Connection

This section introduces the connection method via FortiBalancer web UI (Web User Interface). The FortiBalancer web UI can:

- Improve user experience with fast response time
- Maximize the functionality and performance of the FortiBalancer appliance
- Simplify system management

If administrators want to take full advantage of the Web UI access to the FortiBalancer appliance, please first assign a valid and unique IP address and a port number to the web UI. For example:

```
FortiBalancer(config)#webui ip 10.10.0.2
FortiBalancer(config)#webui port 8888
```

On the FortiBalancer appliance, we use port1's IP address as the default web UI IP address and the port 8888 as the default web UI port.

Then, turn on the web UI function:

```
FortiBalancer(config)#webui on
```

Now open your browser of choice and connect to the FortiBalancer appliance. To do this, simply type in the address bar as such:

```
https://10.10.0.2:8888
```

And now press "Enter". The welcome screen should appear in your browser's window, protected by the familiar prompt asking for user name and password. The default username and password is **admin** and **admin**, just as before. If this screen does not appear, verify the address and port designations for both the port1 interface and web UI port.

The FortiBalancer appliance web UI supports the following browsers:

IE (Recommended)

Firefox

Chrome

Browser resolution should be set to 1024×786 or higher.

1.1.2 Reading the LED

1.1.2.1 LEDs in the Front Panel

The FortiBalancer appliance possesses three LEDs in the front panel: one yellow, one green and one blue. The following is the usage description of each LED in the front panel.

Table 1-2 LEDs in the Front Panel

Color	Meaning	Description
Yellow	Fault	This light is always off when FortiBalancer appliance keeps normal. It means the following problems have come out if this light turns on: • The CPU fan stops working.

Color	Meaning	Description
		• The CPU is overheated (equal to or over 85℃). • The system is overheated (equal to or over 75℃ on 1U appliances, or 85℃ on 2U appliances). • One of the power supply modules breaks down (If the FortiBalancer appliance supports the dual power supply), the redundant power supply will turn on the Buzzer at the same time.
Green	Run	The green LED should blink each second when system is idle. CPU activity will be indicated by the blinking of this light; the faster the rate, the higher the CPU activity.
Blue	Power	Indication of power and the active state (off|on) of the FortiBalancer appliance.

Note: If the yellow LED is lighted, please contact Customer Support. You can view system logs to get more information about the problem.

1.1.2.2 LEDs in the Rear Panel

The FortiBalancer appliance provides two LEDs for every Ethernet port in the rear panel:

- Link LED: indicates the speed mode of the link, which can be 1 Gbps, 10 Mbps or 100 Mbps.
- Activity LED: indicates the activity status of the network port.

The following table describes the meaning of each LED on the onboard and add-on NICs of the FortiBalancer appliance.

Table 1-3 LED for Ethernet Ports in Rear Panel

NIC Type	LED Name	Description
Onboard NIC	Link LED	The Link LED has the following indicator colors: • Amber: The speed mode is 1 Gbps. • Green: The speed mode is 100 Mbps. • Off: No Connection or the speed mode is 10 Mbps.
	Activity LED	The Activity LED has the following indicator colors: • Yellow and blinking: Active • Off: Inactive
Add-on NIC	Link LED	The Link LED has the Yellow indicator color, indicating 1 Gbps, 10 Mbps or 100 Mbps speed mode.
	Activity LED	The Activity LED has the following indicator colors: • Green and blinking: Active • Off: Inactive

1.1.3 Command Line Interface Structure

In this section, you will be provided an overview of the Command Line Interface (CLI) covering the following topics:

- Command Usage Breakdown
- Levels of Access Control

1.1.3.1 Command Usage Breakdown

The CLI allows you to configure and control key functions of the FortiBalancer appliance to better manage the performance of your servers and the accessibility to the contents therein.

The FortiBalancer appliance software has been designed with specific enhancements to make interaction with the Appliance more user friendly, such as Shorthand. Shorthand is the intuitive method by which the Appliance completes CLI commands based on the first letters entered. Other user shortcuts are listed below:

Table 1-4 List of Shortcuts

CLI Shortcuts	Operation
^a/^e	Move the cursor to the beginning/end of a line.
^f/^b	Move the cursor forward/backward one character.
Esc-f	Move the cursor forward one word.
Esc-b	Move the cursor backward one word.
^d	Delete the character under the cursor.
^k	Delete from the cursor to the end of the line.
^u	Delete the entire line.

Note: The symbol "^" indicates holding down the Control (Ctrl) Key while pressing the letter that appears after the symbol.

The FortiBalancer CLI commands will generally adhere to the following style conventions:

Table 1-5 FortiBalancer CLI Style Conventions

Style	Convention
Bold	The body of a CLI command is in Boldface.
Italic	CLI parameters are in Italic.
< >	Parameters in angle brackets < > are required.
[ ]	Parameters in square brackets [ ] are optional. Subcommand such as "**no**", "**show**" and "**clear**" commands.
{x|y|…}	Alternative items are grouped in braces and separated by vertical bars. At least one should be selected.
[x|y|…]	Optional alternative items are grouped in square brackets and separated by vertical bars. One or none is selected.

For example:

ip address *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}*

Note: If a string we input for configuring a parameter starts with figure, or the string contains spaces, we must put the configuration string within double quotes to make sure that we can configure the command correctly.

1.1.3.2 Levels of Access Control

The FortiBalancer appliance's Command Line Interface offers three levels of configuration and access to the OS. The CLI prompt of each level consists of the host name of the FortiBalancer appliance followed by a unique cursor prompt, either ">", "#" or "(config)#".

The first level is for basic network troubleshooting and is called the User level. At this level, the user is only authorized to operate some very basic commands and non-critical functions such as ping and traceroute. Here is how the User level prompt appears in the CLI.

```
FortiBalancer>
```

The second level of administration is the Enable level. Users at this level have access to a majority of view only commands such as "**show version**". Users in the Enable level may execute commands from both the User and Enable levels. In order to gain access to this level of appliance management, the user must employ the command "**enable**". Once this command is entered, the FortiBalancer appliance prompts the user for the appropriate password. If correct password is entered, the CLI prompt will change from "FortiBalancer>" to "FortiBalancer#", which means the user is granted access to the Enable level. The default password for the Enable level is null, i.e. users simply need to press "Enter".

```
FortiBalancer>enable
Enable password:
FortiBalancer#
```

The final access level is the Config level. It is with this level of authority that the user can make changes to the configuration of the box. No two users can access the Config level at the same time. Once a user has gained access to this level, he or she can implement commands in all three levels. To gain access to the full configurable functions of the FortiBalancer appliance, the user must use the following command:

```
FortiBalancer#config terminal
```

Once this command is entered, the CLI prompt will change to:

```
FortiBalancer(config)#
```

In the event that Config level is not available because another Config level session has been opened, the administrator can deploy the following command to gain access to the Config level:

```
FortiBalancer#config terminal force
WARNING:

You are forcing other user to exit configuration mode.
In case the other user is actively changing the system configuration, the result may be
unpredictable.
Do you still want to force into Configuration Mode "YES" or "NO":
```

Type "YES" and press "Enter". You will enter the Config level successfully.

For each level the user can type "?" for available commands. For example, entering "FortiBalancer(config)#**slb real** ?" will prompt users with all the possible parameters or protocols the CLI will accept with the "**slb real**" command.

```
FortiBalancer(config)#slb real ? [enter]
activation     Recovery and warm-up time of real service
disable        Remove real service from load balancing
dns            Define SLB DNS real service
enable         Activate real service for load balancing
ftp            Define SLB FTP real service
…
```

1.2 General Settings Configuration

Now that you are in the configure mode, it is time to assign Port1, Port2 and Gateway IP addresses to truly bring the FortiBalancer appliance into the network infrastructure.

1.2.1 Configuration Guidelines

To better assist you with configuration strategies that maximize the power of the FortiBalancer appliance, please take a moment to familiarize yourself with the basic network architecture.

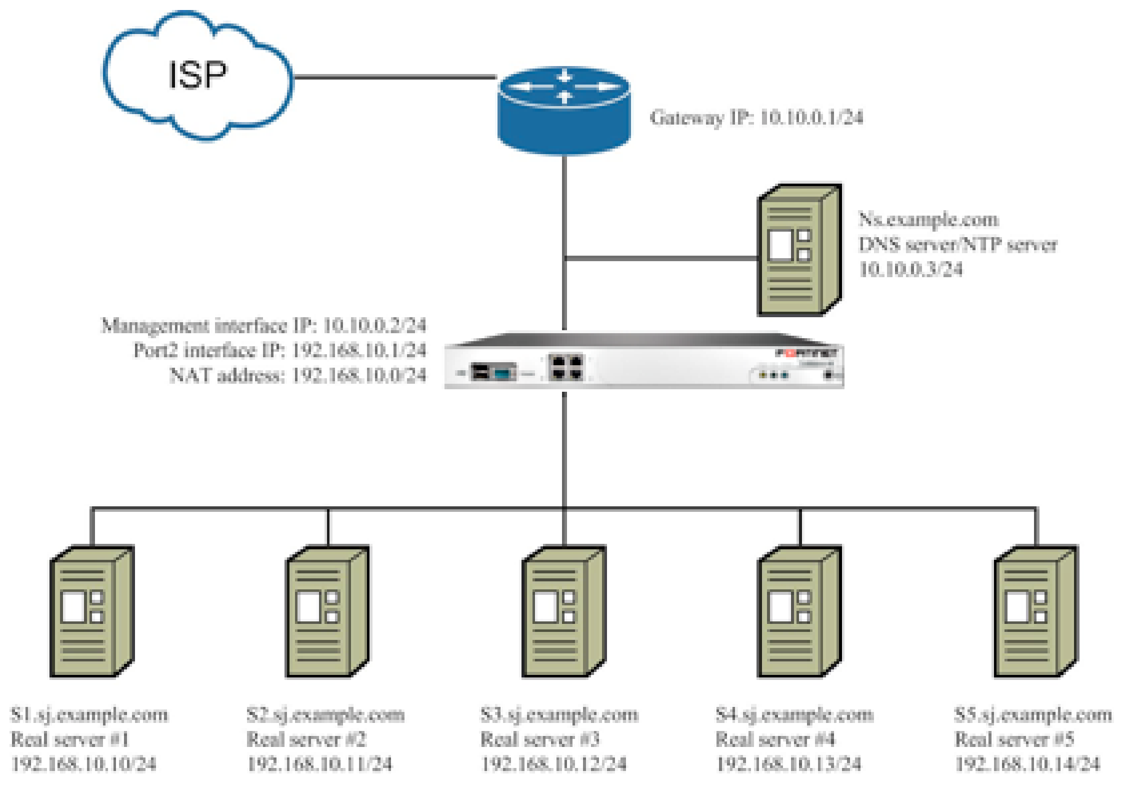


Figure 1-1 Basic Network Architecture

The table below shows the most critical pieces of configurations from the figure above:

Table 1-6 Basic Network Configurations

IP Addess	Description
10.10.0.1/24	Gateway IP Address
10.10.0.2/24	Management IP Address
192.168.10.1/24	Port2 Interface IP Address
192.168.10.0/24	NAT
192.168.10.10	Real Server #1
192.168.10.11	Real Server #2
192.168.10.12	Real Server #3
192.168.10.13	Real Server #4
192.168.10.14	Real Server #5
10.10.0.3	Nameserver/NTP server

Table 1-7 General Settings of Basic Network Configuration

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}*
Configure gateway IP address	**ip route default** *<gateway_ip>*
View IP configurations	**ping** *{ip|hostname}* **show ip address** **show ip route**
Set up web UI	**webui {on|off}** **webui port** *<port>* **webui ip** *<ip_address>*
Assign the host name	**hostname** *<host_name>*
Save the Configurations	**write memory**

1.2.2 Configuration Example via CLI

1.2.2.1 Assigning the IP Address for Interfaces

First, the Port1 Interface IP address needs to be assigned followed by the Port2 Interface, both with the appropriate netmask assignments. Now with our example network addresses and netmask designations, these commands should be executed as such:

```
FortiBalancer(config)#ip address port1 10.10.0.2 255.255.255.0
FortiBalancer(config)#ip address port2 3fff::bb 64
```

The port1 interface and the port2 interface cannot be on the same IP network. The CLI will issue a warning message and will not allow you to configure the two interfaces for the same network.

FortiBalancer supports changing the MAC address of the system interfaces by using the command "**interface mac** <*interface_name*> <*mac_address*>".

```
FortiBalancer(config)#interface mac port1 00:30:48:81:54:9c
```

Note: The administrator will need to provide the method necessary to allow end-users to direct outbound traffic to a preferred route based on the IP and protocol type.

1.2.2.2 Assigning the IP Address for Gateway

The final step in this initial introduction of the FortiBalancer appliance to the network infrastructure requires you to define the Gateway IP address.

To define the gateway IP address:

```
FortiBalancer(config)#ip route default 10.10.0.1
```

1.2.2.3 Viewing the IP Configuration

To verify that FortiBalancer appliance is indeed actively deployed within this network infrastructure, you may ping both the gateway and backend server by using the "**ping**" command.

To ping the gateway:

```
FortiBalancer(config)#ping 10.10.0.1
PING 10.10.0.1(10.10.0.1): 56 data bytes
64 bytes from 10.10.0.1: icmp_seq=0 ttl=128 time=0.671 ms
64 bytes from 10.10.0.1: icmp_seq=1 ttl=128 time=0.580 ms
64 bytes from 10.10.0.1: icmp_seq=2 ttl=128 time=0.529 ms
64 bytes from 10.10.0.1: icmp_seq=3 ttl=128 time=0.486 ms
64 bytes from 10.10.0.1: icmp_seq=4 ttl=128 time=0.638 ms

--- 10.10.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.486/0.581/0.671/0.068 ms
```

To ping the backend server:

```
FortiBalancer(config)#ping 192.168.10.1
PING 192.168.10.1(192.168.10.156 data bytes
64 bytes from 192.168.10.1: icmp_seq=0 ttl=128 time=0.661 ms
64 bytes from 192.168.10.1: icmp_seq=1 ttl=128 time=0.581 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=128 time=0.552 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=128 time=0.484 ms
64 bytes from 192.168.10.1: icmp_seq=4 ttl=128 time=0.632 ms

--- 192.168.10.1 ping statistics ---
```

```
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.486/0.581/0.671/0.068 ms
```

To verify or view the settings after configuring these critical IP addresses:

```
FortiBalancer(config)#show ip address
ip address “port1” 10.10.0.2 255.255.255.0
ip address “port2” 192.168.10.1 255.255.255.0

FortiBalancer(config)#show ip route
Destination        Netmask              Gateway
default                                  10.10.0.1
```

Should changes be required, in most cases, administrators should deploy the “no” version of the command relating to the configured information to remove any incorrect information before entering the desired corrections. For example, executing the command “**no ip address port1**”, will remove the port1 IP address for you to then reenter the correct information.

1.2.2.4 Setting up the web UI

If administrators want to take full advantage of the web UI access to the FortiBalancer appliance, at least one unique IP address is required.

In our example, we use the port1 interface IP address as the default web UI IP address and the default port 8888 as the web UI port. At last, turn on the web UI function:

```
FortiBalancer(config)#webui on
```

It is time to open your browser of choice and point it to the FortiBalancer appliance. To do this, simply type in the address as such:

```
https://10.10.0.2:8888
```

Note: The IP addresses and other parameters throughout these examples are meant for demonstration purposes. To actually access your FortiBalancer appliance, you can designate the web UI IP address and port via the commands “webui ip” and “webui port”.

And now press “Enter”. The welcome screen should appear in your browser’s window, protected by the familiar prompt asking for user name and password. The response to this prompt is **admin** and **admin**, just as before. If this screen does not appear, verify the address and port designations for both the port1 interface and web UI port.

The FortiBalancer appliance web UI supports the following browsers:

IE (Recommended)

Firefox

Chrome

Browser resolution should be set to 1024×786 or higher.

1.2.2.5 Assigning the Host Name

With clustering technology, more than one FortiBalancer appliance may be used within a single network server farm. With this in mind, the OS allows you to assign a “name” to each FortiBalancer appliance for monitoring each device’s performance and configuration specifications. Once you’ve named your FortiBalancer appliance, the prompt will change from the default “FortiBalancer” to the newly assigned name:

```
FortiBalancer(config)#hostname SJ-Box1
SJ-Box1(config)#
```

1.2.2.6 Saving the Configuration

To save your configuration, use the following commands:

```
SJ-Box1(config)#write memory
```

Now your configuration is saved into the startup file which the FortiBalancer appliance calls upon at reboot.

Chapter 2 Advanced Network Configuration

2.1 Overview

This section focuses on introducing the advanced network configurations, including VLAN, MNET, Port Forwarding, NAT, Dynamic Routing and IP Pool functionalities and configurations on the FortiBalancer appliance.

2.1.1 VLAN

VLAN (Virtual Local Area Network) is used to logically segment a network into smaller networks by application, or function, without regard to the physical location of the users. Each VLAN is considered a separate logical network. There are two types of VLAN specifications for Ethernet network.

- Port-based VLAN

Define VLAN based on port number of the switch. Port-based VLAN is easy to configure but often limited to one single switch.

- Tag-based VLAN

Tag-based VLAN allows a group of devices on different physical LAN segments to communicate with each other as if they were all on the same physical LAN segment. In tag-based VLAN, an identifying number, called a "VLAN ID" or a "tag", is written into the Ethernet frame itself, so that switches and routers can use this information to make switching decisions. A tagged frame is four bytes longer than an untagged frame and contains two bytes of Tag Protocol Identifier (TPID) and two bytes of Tag Control Information (TCI).

The FortiBalancer appliance supports Tag-based VLAN on all interfaces. Tag-based VLAN (also known as Trunking by some vendors) is where a tag is inserted into the Ethernet frame so that switches and routers can use this information to make switching decisions.

The FortiBalancer appliance's VLAN can work in both the IPv4 and IPv6 network environments. Administrator can view all the IPv4 and IPv6-based VLAN configurations by executing the command "**show interface**".

For example:

IPv4-based VLAN:

```
FortiBalancer(config)#show interface
……
V1(vlan1): flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 1500
        inet 10.8.66.96 netmask 0xffffff00 broadcast 10.8.66.255
        ether 00:25:90:39:97:f3
        media: autoselect
        status: no carrier
        vlan : 3 parent interface: port1
        webwall status: OFF
        packet drop (not permit): 0
                tcp  0          udp 0          icmp  0          ah  0          esp  0
        packet drop (deny): 0
                tcp  0          udp  0         icmp  0          ah  0          esp  0
        5 minute input rate 0 bits/sec, 0 packets/sec
        5 minute output rate 0 bits/sec, 0 packets/sec
```

IPv6-based VLAN:

```
FortiBalancer(config)#show interface
……
```

```
V2(vlan2): flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 1500
        inet6 fe80::230:48ff:fe93:a73e prefixlen 64 scopeid 0xc
        ether 00:30:48:93:a7:41
        media: autoselect
        status: no carrier
        vlan : 20 parent interface: port2
        webwall status: OFF
        packet drop (not permit): 0
                tcp  0          udp  0          icmp  0         ah  0          esp  0
        packet drop (deny): 0
                tcp  0          udp  0          icmp  0         ah  0          esp  0
        5 minute input rate 0 bits/sec, 0 packets/sec
        5 minute output rate 0 bits/sec, 0 packets/sec
```

2.1.2 MNET

MNET (Multi-Netting) is used to assign more than one IP address on a physical interface. Here is an example for MNET:

A new Internet site is under development for a small corporation. The network administrator knows that the site will grow in the future but today there is no need for a complex network. A server is installed that will be used as Web server, FTP server, mail server, and the corporation's DNS server. Later, when the use of the network services grows, new servers will be used for each of the functions.

When the time comes to address the current server, the administrator has a choice. A single IP address can be used on the server. Later when the new servers are needed, new IP addresses can be assigned to them.

Another way of assigning addresses can be used. The administrator can assign four IP addresses to the server. Each IP address will match the IP address to be used in the future on the new servers. The administrator now knows what addresses will be used and can create DNS entries for the new devices with the correct addresses. This process of providing more than one IP address on an interface is often called multi-netting.

The FortiBalancer appliance's MNET can work in both the IPv4 and IPv6 network environments.

2.1.3 Port Forwarding

Port forwarding allows you to forward any traffic that is destined to an IP/port pair on the FortiBalancer appliance, to an IP/port pair on the inside network. External clients can directly access the internal servers via an address on the outside subnet.

Note: Port Forwarding feature cannot support FTP, users are recommended to use SLB feature instead.

For example, say that you are running SSH on a real server on the port2 interface subnet, and you want to connect to the server from a client in the outside. To get this information from the outside world to the desired inside IP/port pair, we will have to add a port forward on the FortiBalancer appliance.

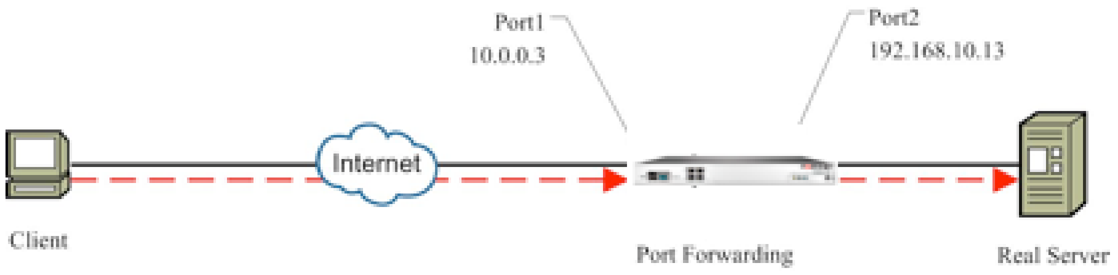


Figure 2-1 Port Forwarding

2.1.4 NAT

NAT (Network Address Translation) is the translation of an IP address used within one network to a different IP address known within another network. One network is designated the inside network and the other is the outside. NAT is used when you want your real servers on the inside network to access the outside network. Using NAT, all packets will appear as though they came from the FortiBalancer appliance.

2.1.4.1 How NAT Works

When a client on the inside network contacts a machine on the outside network, it sends out IP packets destined for that machine. These packets contain all the addressing information necessary to get them to their destination.

When the packets pass through the NAT gateway, they will be modified so that they appear to be coming from the NAT gateway itself. The NAT gateway will record the changes it makes in its state table so that it can reverse the changes on return packets and ensure that return packets are passed through the firewall without being blocked.

NAT Traversal of PPTP

The FortiBalancer appliance supports NAT traversal of PPTP (Point-to-Point Tunneling Protocol).

PPTP is a tunneling mechanism to transfer PPP (Point-to-Point Protocol) frames across intermediate networks. PPTP uses GRE (Generic Routing Encapsulation) encapsulation for PPP payload. However, typically GRE tunnels cannot traverse the NAT device (i.e. the FortiBalancer appliance). To resolve this problem, a PPTP NAT editor is used on the FortiBalancer appliance. This editor is an additional software component on the NAT device to perform translation for IP addresses, TCP ports, and call ID.

2.1.4.2 Supported NAT Types

The FortiBalancer appliance supports four types of NAT:

- **Static NAT**

Mapping an IP address on one-to-one basis. By configuring static NAT, the FortiBalancer appliance maps an inside real IP address to a VIP address on the outside. For inbound traffic directed from an outside VIP, the traffic will be forwarded to a corresponding inside real IP without any change in the port number or protocol value. Thus, hosts on the inside network will be directly accessible via the VIP on the port1 interface. The outbound traffic coming from an inside host will use the corresponding outside VIP as the source IP for the outgoing traffic. The port number and protocol remain unchanged. TCP, UDP and ICMP are supported for static NAT.

In static NAT, the computer with the IP address (192.168.10.13) will be always translated into 10.10.0.3:

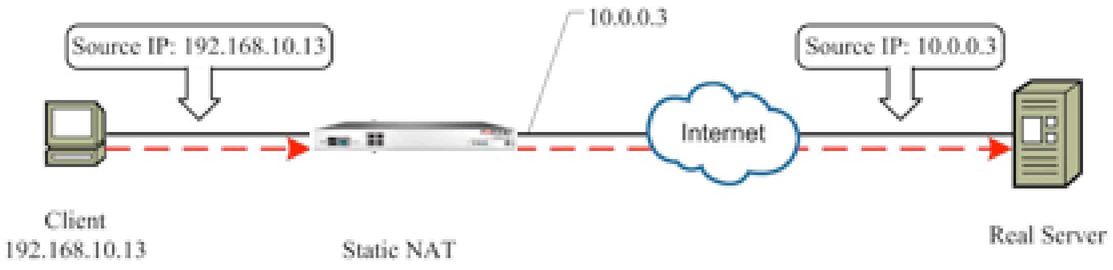


Figure 2-2 Static NAT

- **Port-level NAT**

Mapping multiple inside real IP addresses to a single VIP address with a different port number assignment on the port1 interface. By configuring port-level NAT, the group of hosts on the inside network will be directly accessible via the VIP on the port1 interface.

In port-level NAT, the computers with the IP address in the range from 192.168.10.11 to 192.168.10.13 will be translated into 10.0.0.3 with a different port on the outside network:

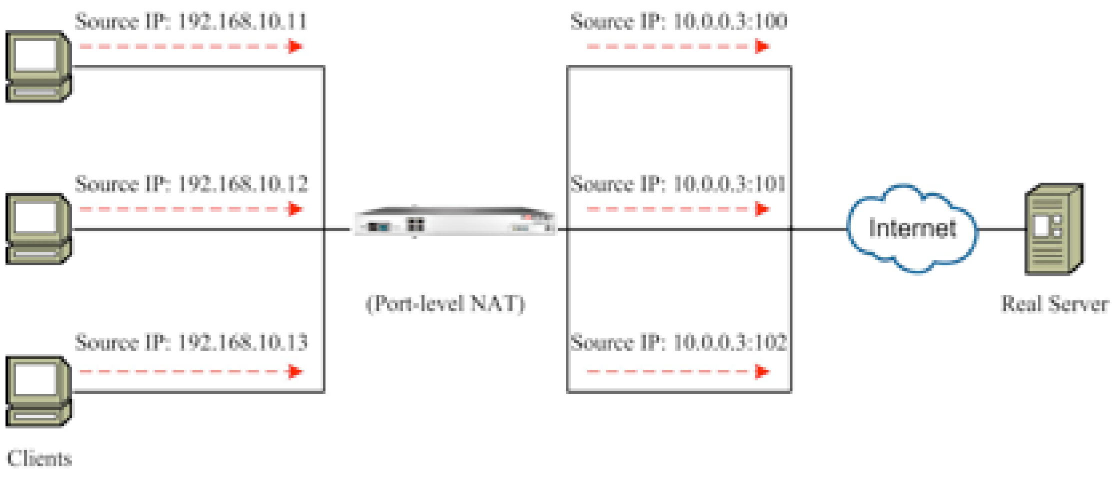


Figure 2-3 Port-level NAT

If a port-level NAT is configured for an inside real IP address, static NAT should take precedence over the regular NAT policy. VIPs used by static NAT should not be used by regular NAT. Also, one static NAT VIP should not map to multiple real IP addresses.

➢ **IP pool-based dynamic NAT**

Mapping one or multiple inside real IP addresses to an IP pool which contains multiple IP addresses. By configuring IP pool-based dynamic NAT, a group of hosts on the inside network will be translated into via the multiple IP addresses in IP pool.

As shown below, in IP pool-based dynamic NAT, the client IP addresses in the range from 192.168.10.11 to 192.168.10.13 will be translated into 10.0.0.1 to 10.0.0.3 and get connected to the Internet.

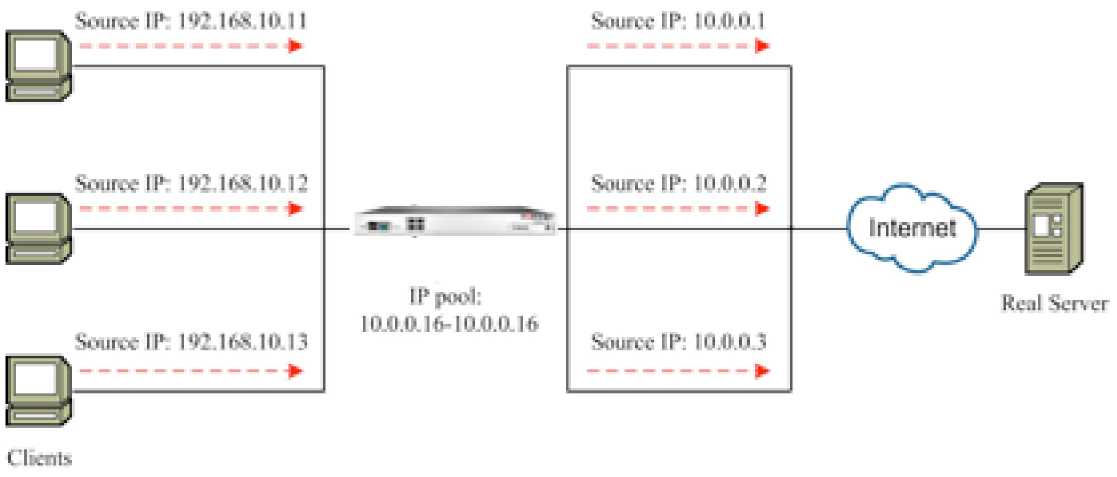


Figure 2-4 IP Pool-based Dynamic NAT

➢ **Destination IP based NAT**

FortiBalancer supports NAT based on destination IP address. The destination IP based NAT can be performed only when the destination IP address is in the specified network segment and in the same network segment as the route gateway. If the gateway is set to the default value 0.0.0.0, the destination IP/IP pool for NAT and the route gateway should be within the same network segment.

Note: For IPv6 address, only TCP and UDP packets can be NATTed and no gateway can be configured.

Configuration Example via CLI

➢ Step 1 Configure IP pool

```
FortiBalancer(config)#ip pool "pool1" 172.16.72.45 172.16.72.55
FortiBalancer(config)#ip pool "pool3" 3ffd::45 3ffd::46
```

➢ Step 2 Configure destination NAT

```
FortiBalancer(config)#nat portdst "pool1" 172.16.72.0 255.255.255.0 500 172.16.72.101
FortiBalancer(config)#nat portdst "pool3" 3ff1:: 65 50
```

Check NAT configurations and statistics

The following two commands can be used to check the NAT configurations and statistics.

➢ Check NAT table

```
FortiBalancer(config)#show nat table
From 172.16.73.108(5208) through 172.16.72.53(50776) to 172.16.72.77(80)
From 172.16.73.108(5200) through 172.16.72.53(50768) to 172.16.72.77(80)
From 172.16.73.108(5880) through 172.16.72.53(51448) to 172.16.72.77(80)
From 172.16.73.108(6008) through 172.16.72.53(51576) to 172.16.72.77(80)
From 3ffd::108(42720) through 3ffd (45672) to 3ff1::77(80)
From 3ffd::108(42728) through 3ffd (45680) to 3ff1::77(80)
From 3ffd::108(43336) through 3ffd (46288) to 3ff1::77(80)
From 3ffd::108(43376) through 3ffd (46328) to 3ff1::77(80)
From 3ffd::108(43464) through 3ffd (46416) to 3ff1::77(80)
From 3ffd::108(43408) through 3ffd (46360) to 3ff1::77(80)
From 3ffd::108(43840) through 3ffd (46792) to 3ff1::77(80)
From 3ffd::108(43896) through 3ffd (46848) to 3ff1::77(80)
```

➢ Check NAT statistics

```
FortiBalancer(config)#show statistics nat
nat portdst "pool1" 172.16.72.0 255.255.255.0 500 172.16.72.101
            protol(total/current):icmp(1/1) udp(0/0) tcp(23724/251)
nat portdst "pool3" 3ff1:: 65 50
            protol(total/current):icmp(0/0) udp(0/0) tcp(23716/246)
```

2.1.5 Dynamic Routing

Dynamic Routing is a process in which routers automatically adjust to changes in network topology or traffic. It is more robust than static routing. Now, there are several protocols used to support dynamic routing including RIPv1 (Routing Information Protocol version 1), RIPv2 (Routing Information Protocol version 2) and OSPFv2 (Open Shortest Path First version 2) and OSPFv3 (Open Shortest Path First version 3).

Dynamic Routing is especially suitable for today’s large, constantly changed networks. It improves performance by allowing network routers to adjust to changes in the network topology. And it distributes routing information between routers and chooses the best path for the network, saving money and improving performance.

2.1.6 IP Pool

An IP pool contains multiple IP addresses from one IP segment. Administrators can use the pre-defined IP pools for NAT and SLB configurations.

For the NAT module, IP pool can be defined on the outside interface to realize translation of multiple outgoing IP addresses. This helps increase the concurrent capacity and fully utilize the IP resources. For the SLB module, IP pool can be defined for real server groups. Under the SLB reverse mode, when different real server groups are selected, FortiBalancer can select different IP addresses in IP pools to connect to real servers. This not only increases the concurrent connection capacity, but also provides a more flexible configuration method for administrators.

FortiBalancer appliance supports multiple IP pools, and at most 256 IP addresses can be added in one IP pool. The maximum number of IP pools allowed on the FortiBalancer appliance varies with different system memories. Please see the table below for details.

Table 2-1 Maximum IP Pool Entry

System Memory	Maximum IP Pool Number
4GB	32
8GB	64
16GB	128
32GB	256

Both IPv4 and IPv6 addresses can be configured in the IP pool.

When configuring the IP pool, please note:

- Each IP pool should be assigned with a unique name in the system.
- The IP addresses in one IP pool must belong to the same subnet.
- The subnet can be the same between two or more pools.
- One IP address can belong to multiple IP pools.
- IP segment is composed of continuous IPs, and an IP pool can be composed of multiple IP segments.
- IP addresses in IP pool must be legal.
  - IP addresses which are not covered by any interface subnet are illegal.
  - Broadcast IP address is illegal.
  - IP with hostID 0 is illegal.

2.2 Advanced Network Configuration

2.2.1 Configuration Guidelines

To better assist you with configuration strategies that maximize the power of the FortiBalancer appliance, please take a moment to familiarize yourself with the network architecture for advanced network configuration.

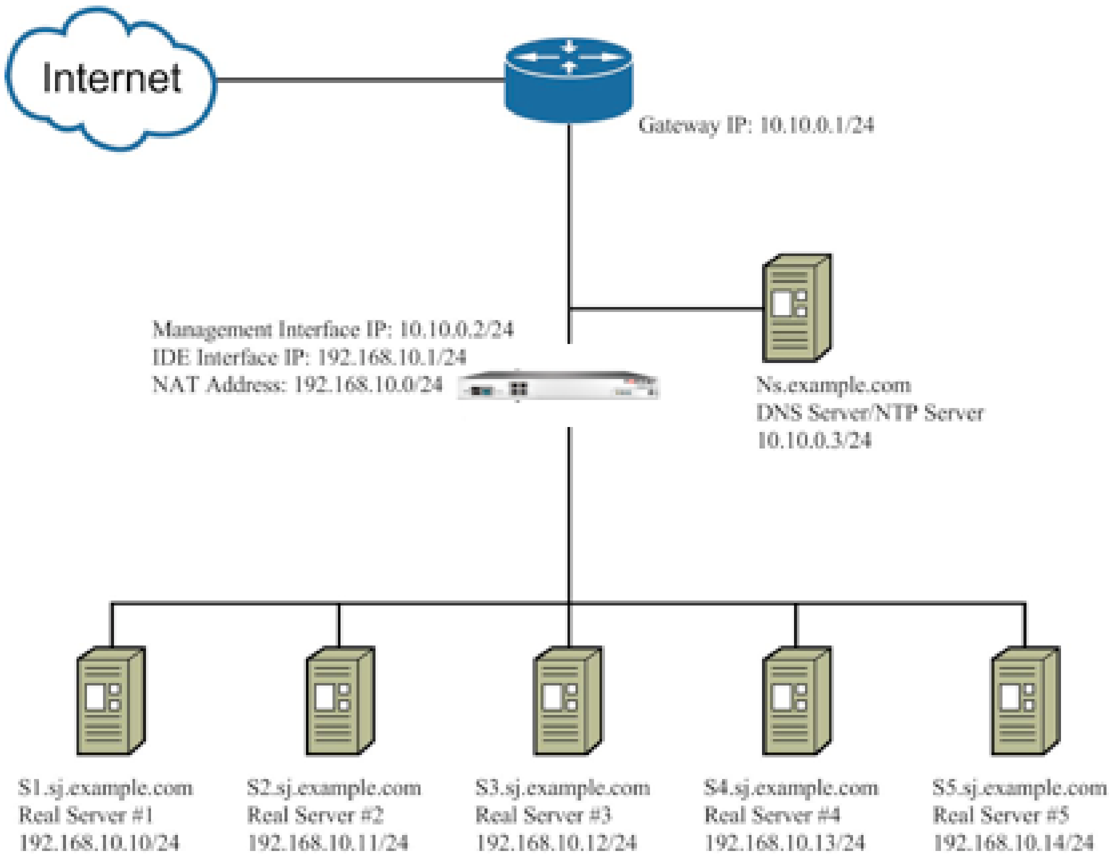


Figure 2-5 Advanced Network Architecture

The table below shows the most critical pieces of configurations from the figure above:

Table 2-2 Advanced Network Configurations

IP Address	Description
10.10.0.1/24	Gateway IP Address
10.10.0.2/24	Management IP Address
192.168.10.1/24	Port2 Interface IP Address
192.168.10.0/24	NAT
192.168.10.10	Real Server #1
192.168.10.11	Real Server #2
192.168.10.12	Real Server #3
192.168.10.13	Real Server #4
192.168.10.14	Real Server #5
10.10.0.3	Nameserver/NTP server

Table 2-3 General Settings of Advanced Network Configuration

Operation	Command
Configure VLAN	**vlan** *{system_ifname|bond_ifname} <user_interface_name> <vlan_tag>*
Configure MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
Configure Port Forwarding	**fwd tcp** *<local_ip> <local_port> <remote_ip> <remote_port> [timeout]* **fwd udp** *<local_ip> <local_port> <remote_ip> <remote_port> [timeout]*
Configure NAT	**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]* **nat static** *<vip> <network_ip> [timeout] [gateway] [description]*
Configure Dynamic Routing	**rip {on|off}** **rip network** *<ip_address> <netmask>*

Operation	Command
	ospf {on|off} **ospf network** *<ip_address> <netmask> <area_id>*
Configure IP pool	**ip pool** *<pool_name> <start_ip> [end_ip]* **slb proxyip global** *<pool_name>* **slb proxyip group** *<group_name> <pool_name>*

2.2.2 Configuration Example via CLI

2.2.2.1 VLAN Configuration

In our example, we are going to create two VLANs, "inside-vlan1" and "inside-vlan2". The "inside-vlan1" has a tag of 500 and "inside-vlan2" has a tag of 3001. These tags are inserted into the Ethernet frame.

➢ Step 1 Define a VLAN interface by using the "**vlan**" command

```
FortiBalancer(config)#vlan port2 inside-vlan1 500
FortiBalancer(config)#vlan port2 inside-vlan2 3001
```

➢ Step 2 Assign an IP address to each VLAN interface by using the "**ip address**" command

```
FortiBalancer(config)#ip address inside-vlan1 192.168.1.1 255.255.255.0
FortiBalancer(config)#ip address inside-vlan2 192.168.2.1 255.255.255.0
```

For the interface with VLAN configuration, it needs to be connected to a switch or router with Tag VLAN or Trunking turned on. See your switch vendors' documentation on how to setup Tag VLAN.

2.2.2.2 MNET Configuration

Configuring MNET on the port2 interfaces is very similar to VLAN configuration. For our example network, we will run two networks over the port2 interface, 192.168.1.1/24 and 192.168.2.1/24.

➢ Step 1 Define our mnet interfaces by using the "**mnet**" command

```
FortiBalancer(config)#mnet port2 mnet1
FortiBalancer(config)#mnet port2 mnet2
```

➢ Step 2 Assign an IP address to each MNET by using the "**ip address**" command

```
FortiBalancer(config)#ip address mnet1 192.168.1.1 255.255.255.0
FortiBalancer(config)#ip address mnet2 192.168.2.1 255.255.255.0
```

Again you need to refer to your vendor's switch/router documentation on how to setup their interface for use with MNET.

2.2.2.3 Port Forwarding Configuration

For our example configuration, we will be adopting the TCP port forwarding protocols as such:

```
FortiBalancer(config)#fwd tcp 10.10.0.2 4000 192.168.10.10 22 300
```

We picked an arbitrary high port to use. You should not use a port below 1024 on the FortiBalancer appliance since other services might be listening on those ports, i.e. 443 (for SSL) and 80 (for HTTP). We can choose a port below 1024 on the real server since that is the service that we want to connect to. To view or alter these forwarding instructions, employ the show, no or clear versions of the above commands.

2.2.2.4 NAT Configuration

For our configuration example strategy, use the command as:

```
FortiBalancer(config)#nat port 10.10.0.2 192.168.10.0 255.255.255.0 60 10.10.0.1
```

This command will perform NAT on the 192.168.10.0/24 network. In our example, the VIP 10.10.0.2 and the route gateway 10.10.0.1 are within the same network segment. Therefore the parameter “gateway” in the command “**nat port**” can be set to the default value 0.0.0.0 or the route gateway. If the VIP and the route gateway are not in the same network segment, the parameter “gateway” in the command “**nat port**” must be set to the route gateway.

We can change the netmask to allow only certain blocks of your inside network to access the external network. For example, the following command will only allow the IP addresses ranging 192.168.10.0 through192.168.10.128, to access the external network:

```
FortiBalancer(config)#nat port 10.10.0.2 192.168.10.0 255.255.255.128 60 0.0.0.0
```

If we want to allow the top half of the IP address space range that is left over (192.168.10.129-192.168.10.254), to access the external network, we will do the following:

```
FortiBalancer(config)#nat port 10.10.0.2 192.168.10.129 255.255.255.128 60 0.0.0.0
```

If we want to allow one real IP address to access the external network, we will configure static NAT:

```
FortiBalancer(config)#nat static 10.10.0.2 192.168.10.12
```

2.2.2.5 Dynamic Routing Configuration

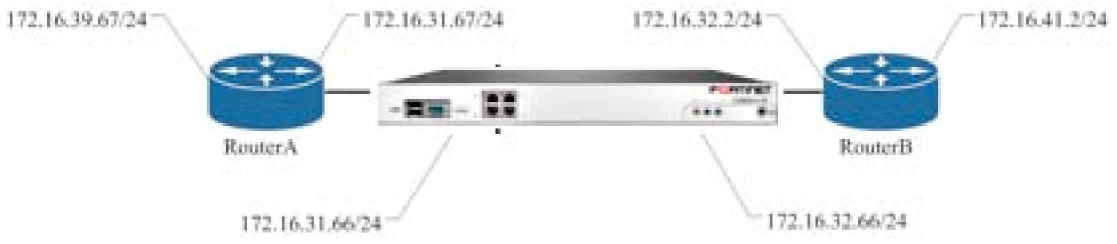


Figure 2-6 Dynamic Routing Configuration

- Step 1 RIP Configurations

```
FortiBalancer(config)#rip on
FortiBalancer(config)#rip version 2
FortiBalancer(config)#rip network 172.16.31.0 255.255.255.0
FortiBalancer(config)#rip network 172.16.32.0 255.255.255.0
```

- Step 2 OSPF Configurations

```
FortiBalancer(config)#ospf on
FortiBalancer(config)#ospf network 172.16.32.0 255.255.255.0 0
FortiBalancer(config)#ospf network 172.16.31.0 255.255.255.0 0
```

After these configurations, you can view the dynamically generated routes by using the “**show ip route**” command.

```
FortiBalancer(config)#show ip route
Destination      Netmask          Gateway
RIP routes:
Destination      Netmask          Gateway
172.16.39.0      255.255.255.0    172.16.31.67
```

```
OSPF routes:
Destination        Netmask             Gateway
172.16.41.0        255.255.255.0      172.16.32.2
```

Now that the very basics of our example network configurations are implemented, it is time to move forward to configure the FortiBalancer appliance to operate seamlessly within the network architecture.

2.2.2.6 IP Pool Configuration

Configuration Example for NAT IP Pool via CLI

In our example, we are going to configure IP pools for NAT.

- Step 1 Define IP pools by using the "**ip pool**" command

```
FortiBalancer(config)#ip pool "pool1" 124.0.0.22 124.0.0.22
FortiBalancer(config)#ip pool "pool2" 124.0.1.22 124.0.1.22
```

- Step 2 Define the IP pool for NAT via the "**nat port**" command

```
FortiBalancer(config)#nat port "pool1" 1.1.1.0 255.255.255.0 60 124.0.0.125
FortiBalancer(config)#nat port "pool2" 1.1.1.0 255.255.255.0 60 124.0.1.125
```

Configuration Example for SLB IP Pool via CLI

In our example, we are going to configure IP pools for SLB.

- Step 1 Define IP pools by using the "**ip pool**" command

```
FortiBalancer(config)#ip pool "pool1" 124.0.0.22 124.0.0.22
FortiBalancer(config)#ip pool "pool2" 124.0.1.22 124.0.1.22
```

- Step 2 Define the IP pool as the global proxy IP pool by using the "**slb proxyip global**" command

```
FortiBalancer(config)#slb proxyip global "pool2"
```

- Step 3 Assign the IP pools for SLB group

```
FortiBalancer(config)#slb proxyip group "gpi" "pool1"
```

Note: The priority of group IP pools is higher than global IP pools.

Chapter 3 Link Aggregation

3.1 Overview

This section describes link aggregation functionality of the network. Link Aggregation is also called trunking, which can greatly improve network performance and stability.

3.2 Understanding Link Aggregation

Link Aggregation or trunking is a method of combining physical network links into a single logical link for increased bandwidth. With Link Aggregation, we are able to increase the capacity and availability of the communication channel between devices. Two or more Gigabit Ethernet connections are combined in order to increase the bandwidth capability and create resilient and redundant links.

The FortiBalancer appliance supports at most 6 bond interfaces, and at most 12 system interfaces can be added to a bond interface. The bond interface will check the status of the system interfaces. If a system interface becomes down, the traffic processed by this interface will be directed to other working system interfaces in the bond interface.

To add a system interface into a bond interface, the administrator can further set the interface as the primary or backup interface in the bond. Multiple primary or backup interfaces can be set in a bond. When all the primary interfaces in the bond fail, the backup interfaces will take the place of primary interfaces to work.

Note: To bind a system interface with a bond interface, the system interface should be configured with no IP address information. If there is IP configuration on the system interface, the administrator needs to remove the IP configuration first. If otherwise, the system will refuse to add the system interface into the bond.

In addition, the FortiBalancer appliance also supports configuring MNET or VLAN on bond interface. The bond interface configuration must be performed before configuring MNET or VLAN on it.

3.3 Link Aggregation Configuration

3.3.1 Configuration Guidelines

Before you start to configure link aggregation, please take a moment to familiarize yourself with the network architecture for link aggregation configuration.

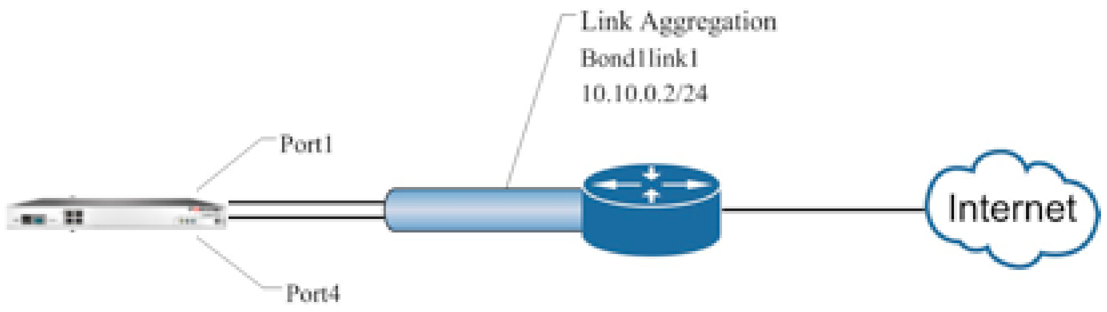


Figure 3-1 Link Aggregation Configuration

Table 3-1 General Settings of Link Aggregation

Operation	Command
Bond system interfaces	**bond interface** *<bond_name> <interface_name> [1|0]*

Operation	Command
Assign a name for the bond interface	**bond name** *<bond_id> <bond_name>*
Assign IP address to the bond interface	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}*
Assign default route	**ip route default** *<gateway_ip>*

3.3.2 Configuration Example via CLI

➢ Step 1 Bind system interfaces with bond interface

In our example, we bind the "port1" interface and the "port4" interface with the bond interface bond1, and further set the "port1" interface as the primary, and "port4" as the backup in the bond interface.

```
FortiBalancer(config)#bond interface bond1 port1 1
FortiBalancer(config)#bond interface bond1 port4 0
```

➢ Step 2 Assign a name for the bond interface

We can set the bond name for the configured bond interface by using the "**bond name**" command.

```
FortiBalancer(config)#bond name bond1 link1
```

➢ Step 3 Assign an IP address and netmask to the bond interface

```
FortiBalancer(config)#ip address link1 10.10.0.2 255.255.255.0
```

➢ Step 4 Set the gateway IP address

```
FortiBalancer(config)#ip route default 10.10.0.1
```

To verify that FortiBalancer appliance is indeed actively deployed within this network infrastructure, you may ping the gateway IP by using the "**ping**" command.

If these configurations are entered correctly, you will receive the following return messages.

```
FortiBalancer(config)#ping 10.10.0.1
PING 10.10.0.1(10.10.0.1): 56 data bytes
64 bytes from 10.10.0.1: icmp_seq=0 ttl=128 time=0.671 ms
64 bytes from 10.10.0.1: icmp_seq=1 ttl=128 time=0.580 ms
64 bytes from 10.10.0.1: icmp_seq=2 ttl=128 time=0.529 ms
64 bytes from 10.10.0.1: icmp_seq=3 ttl=128 time=0.486 ms
64 bytes from 10.10.0.1: icmp_seq=4 ttl=128 time=0.638 ms


--- 10.10.0.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.486/0.581/0.671/0.068 ms
```

Chapter 4 Clustering

4.1 Overview

The clustering function allows you to maintain high availability within a local site. With other options you can also distribute load across multiple boxes within a cluster.

4.2 Understanding Clustering

The Clustering function allows two or more FortiBalancer appliances to be grouped together to form a logical device, which provides scalability and high availability within a local site. Please refer to the following figure.

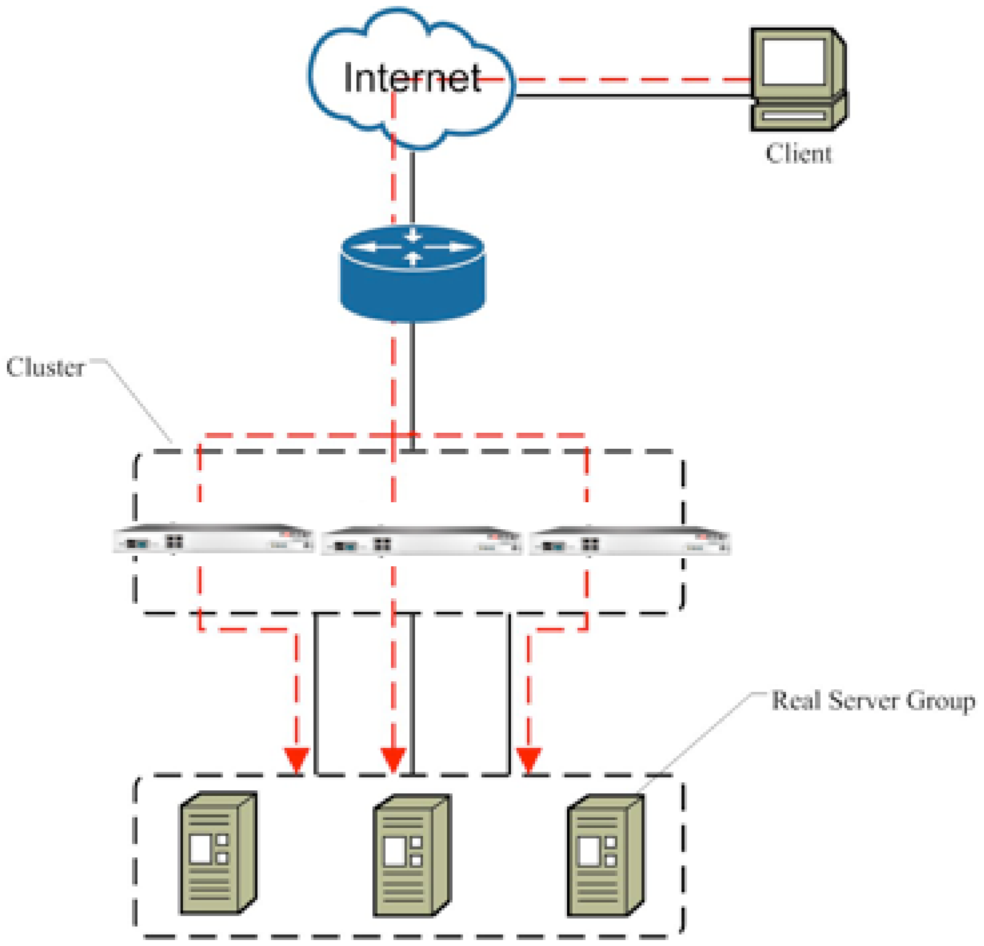


Figure 4-1 FortiBalancer Clustering

Clustering can be configured in Active-Standby (A/S) or Active-Active (A/A) mode:

Active-Standby mode – In Active-Standby mode, all VIPs on one FortiBalancer appliance in the cluster will be the master, and all VIPs on the other FortiBalancer appliances in the cluster are standby. In this mode, clustering supports fast failover.

Active-Active mode – In Active-Active mode, each FortiBalancer appliance in the cluster has a different master VIP or cluster ID.

4.2.1 Fast Failover

The Fast Failover (FFO) mechanism uses a new additional serial port (fast failover port on the FortiBalancer appliance mother board) to detect each other's status transparently in a cluster (refer to the following figure). When one system powers off, panics, reboots or its interface losses carrier (link disconnection), all the traffic will be immediately switched to the other. The Clustering function with fast failover mechanism provides higher availability and much faster response time than the typical Clustering.

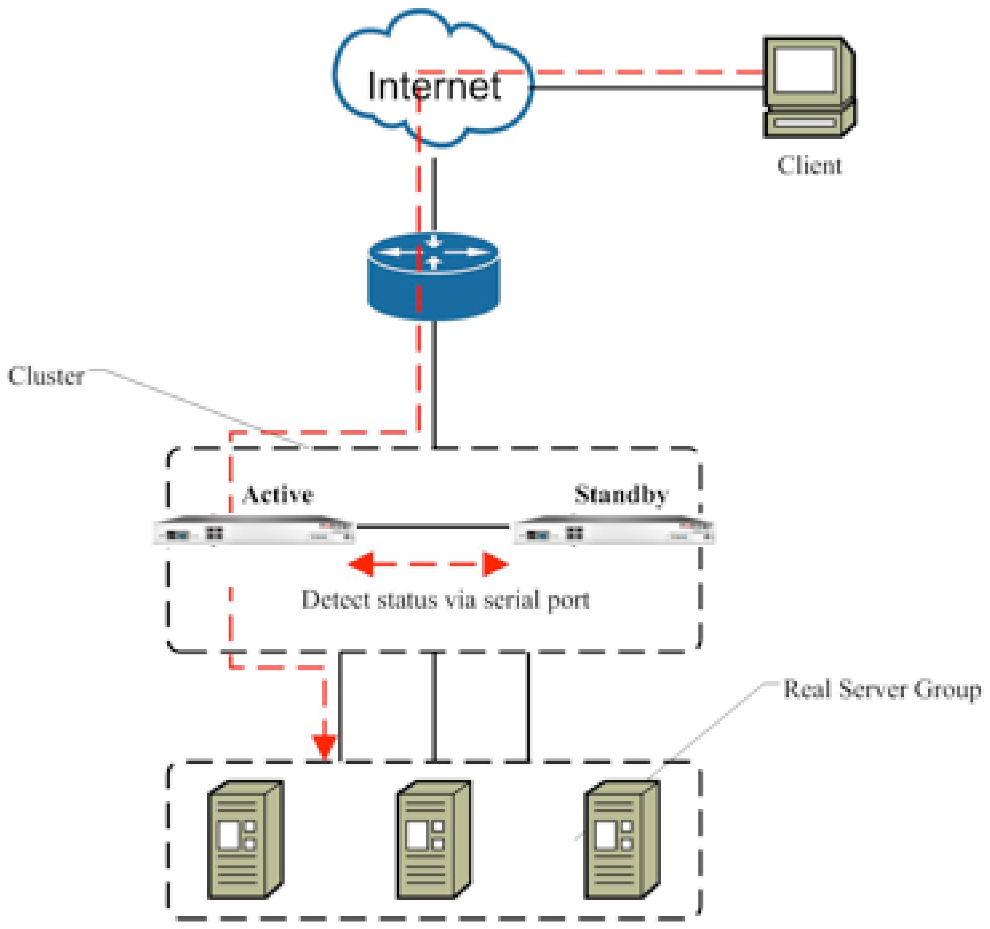


Figure 4-2 Clustering FFO Mode

4.2.2 Discreet Backup Mode

For traditional clustering, a backup and a master communicate each other's state information through the network. If the backup does not receive the VRRP (Virtual Router Redundancy Protocol) multicast packets from the master within a specified time, it will mandatorily preempt the master. However, because of the network complexity, when something totally unexpected happens, this way may lead to a double-master state.

Discreet Backup mode is designed to prevent a double-master state. In this mode, the system determines whether a state transition is needed for the devices based on their state information detected by a heartbeat cable. This mode makes the state transition more reliable, and any VRRP packet loss will not result in double-master state.

The following shows how the Discreet Backup mode works.

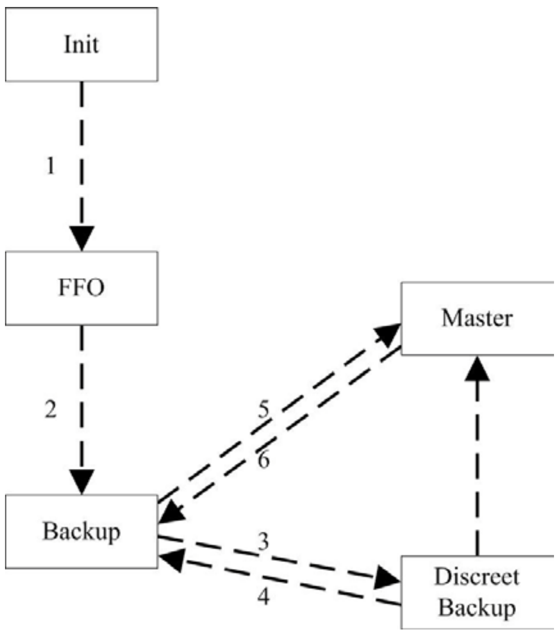


Figure 4-3 Discreet Backup Mode Working Mechanism

1. After turning on clustering, the device enters into Init state. Then, in order to check the health of the heartbeat cable, the Init device switches to FFO state.
2. The device collected the health information of the heartbeat cable. If the heartbeat cable is well connected, it will switch to Backup state.
3. Note: Even though the heartbeat cable is disconnected, the device will still switch to Backup state, and clustering will work well. However, the discreet mode is invalid.
4. If the backup receives a higher priority VRRP packet, it will switch to Discreet Backup state.
5. In the following events, the discreet backup will switch to Backup state:
6. The device in Discreet Backup state receives a lower priority VRRP packet (after the successful state transition, the backup will go on to switch to Master state.).
7. The device in Discreet Backup state will check the heartbeat cable health. If the heartbeat cable is disconnected, it will log out to Backup state.
8. In the following events, the backup will switch to Master state:
9. The backup receives a lower priority VRRP packet (in Preemption mode).
10. In three continuous broadcast intervals (the default interval is 5 seconds, three intervals are 15 seconds), the backup does not receive the VRRP packet from the master.
11. If the master receives a higher priority VRRP packet, it will switch to Backup state.
12. If the heartbeat cable detected the master’s NIC is down, the discreet backup will switch to Master state directly.

Note: All cluster state transitions can be traced by the command “show cluster virtual transition”.

By default, discreet backup mode is turned off.

To configure the discreet backup mode, the following two commands MUST be configured first to turn on the discreet backup mode.

```
FortiBalancer(config)#cluster virtual ffo on
FortiBalancer(config)#cluster virtual discreet on
```

4.2.3 IPv6 Support for Clustering

The FortiBalancer Clustering function now supports IPv6 VIPs switchover. Both IPv4 and IPv6-based VRRP packets can be processed by the FortiBalancer appliance.

If the interface for Clustering is configured with both the IPv4 and IPv6 addresses or with only the IPv4 address, then the IPv4-based VRRP packets will be used for communication between the FortiBalancer appliances. If only the IPv6 address is configured on the interface for Clustering, then the IPv6-based VRRP packets will be used.

Note: The VRRP packets are incompatible with each other among different OS versions. So please use the same OS version for the FortiBalancer appliances in a cluster.

4.3 Clustering Configuration

4.3.1 Clustering SLB VIPs

When using the clustering capabilities of the FortiBalancer appliance, we will first define our SLB virtual IPs that we want to use in the cluster. Each of the following sections will define the virtual IPs that we will use.

For information about SLB, please refer to the chapter Server Load Balancing (SLB).

4.3.1.1 Active-Standby: Two Nodes

Configuration Guidelines

In Active-Standby mode, one node in the cluster will be the master of the VIP, and thus active. The other node in the cluster will be in standby mode. Upon failure of the active node, the standby node will take over the VIP and become master. If preemption has been enabled on the initial master node, it will reassume mastership when it returns to a working state. Otherwise, the VIP will stay with the new master node until the node fails.

Refer to the following figure for the typical layout of Active-Standby architecture, in which:

- FortiBalancer1 is the current master, and handles SLB traffic for VIP.
- FortiBalancer2 is the backup, and listens for advertisements from the master. It will resume master status if FortiBalancer1 stops sending advertisements (i.e. FortiBalancer1 fails).

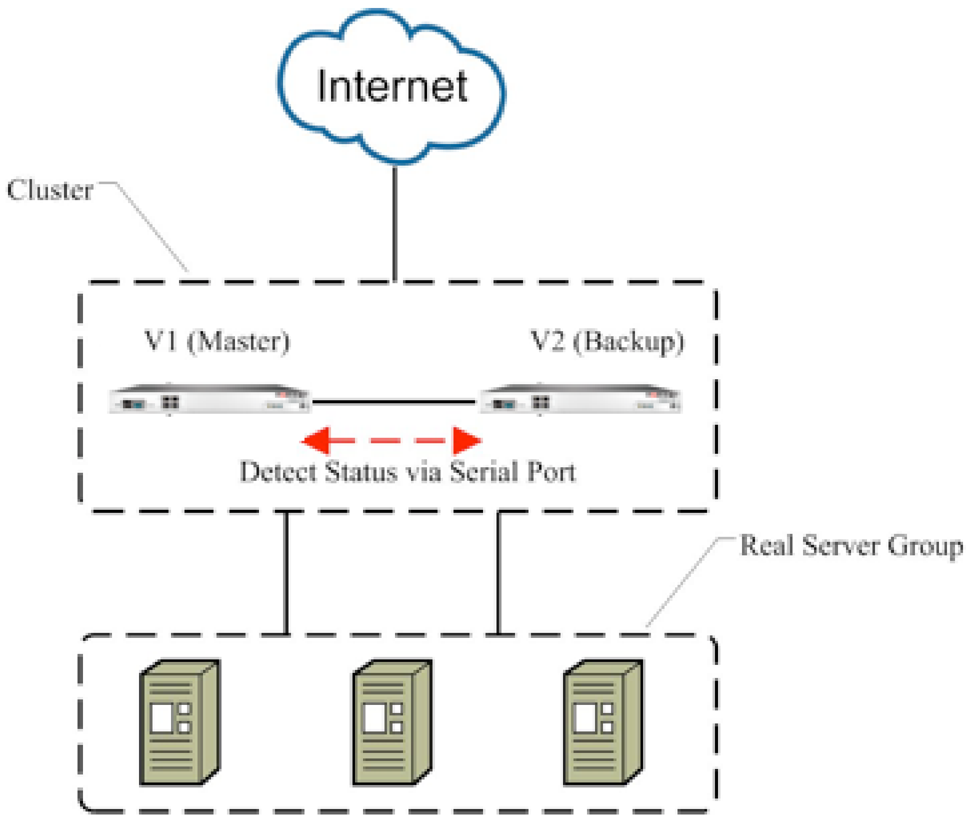


Figure 4-4 Active-Standby Two-Node Architecture

Table 4-1 General Settings of Active-Standby Two-Node Clustering

Operation	Command
Configure SLB	Refer to the SLB Configuration section.
Configure a virtual interface	**cluster virtual ifname** *<interface_name> <cluster_id>*
Configure virtual cluster authentication	**cluster virtual auth** *<interface_name> <cluster_id> {0|1} [password]*
Configure preemption	**cluster virtual preempt** *<interface_name> <cluster_id> <mode>*
Configure virtual IP	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
Configure priority	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
Enable the virtual cluster	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*
Enable fast failover feature	**cluster virtual ffo {on|off}** **cluster virtual ffo interface carrier loss timeout** *<interface_timeout>*

Configuration Example for Active-Standby SLB Clustering via CLI

Now let’s start to configure FortiBalancer1 and FortiBalancer2:

➢ Step 1 Configure SLB for both FortiBalancer1 and FortiBalancer2

```
FortiBalancer1(config)#slb real http “server1” 192.168.1.50 80 1000 tcp 1 1
FortiBalancer1(config)#slb real http “server2” 192.168.1.51 80 1000 tcp 1 1
FortiBalancer1(config)#slb group method “group1” rr
FortiBalancer1(config)#slb group member “group1” “server1” 1
FortiBalancer1(config)#slb group member “group1” “server2” 1
FortiBalancer1(config)#slb virtual http “vip1” 192.168.2.100 80
FortiBalancer1(config)#slb policy default “vip1” “group1”
FortiBalancer2(config)#slb real http “server1” 192.168.1.50 80 1000 tcp 1 1
```

```
FortiBalancer2(config)#slb real http “server2” 192.168.1.51 80 1000 tcp 1 1
FortiBalancer2(config)#slb group method “group1” rr
FortiBalancer2(config)#slb group member “group1” “server1” 1
FortiBalancer2(config)#slb group member “group1” “server2” 1
FortiBalancer2(config)#slb virtual http “vip1” 192.168.2.100 80
FortiBalancer2(config)#slb policy default “vip1” “group1”
```

➢ Step 2 Configure a virtual interface name

```
FortiBalancer1(config)#cluster virtual ifname “port1” 100
FortiBalancer2(config)#cluster virtual ifname “port1” 100
```

➢ Step 3 Configure virtual cluster authentication

It is recommended that you run clustering with an authentication string to avoid unauthorized participation in your cluster.

```
FortiBalancer1(config)#cluster virtual auth port1 100 0
FortiBalancer2(config)#cluster virtual auth port1 100 0
```

➢ Step 4 Configure virtual cluster preemption

Now we configure FortiBalancer1 to preempt the VIP when the initial master returns online. For FortiBalancer2, it will not preempt the VIP from the master node, but will take over if the master ceases operations.

```
FortiBalancer1(config)#cluster virtual preempt port1 100 1
FortiBalancer2(config)#cluster virtual preempt port1 100 0
```

➢ Step 5 Define the VIP by the “**cluster virtual vip**” command

```
FortiBalancer1(config)#cluster virtual vip “port1” 100 192.168.2.100
FortiBalancer2(config)#cluster virtual vip “port1” 100 192.168.2.100
```

➢ Step 6 Define the priority

Cluster priority determines which node becomes the master. The node with highest priority becomes the master. Since we want FortiBalancer1 to always be master of the VIP, we will set its priority to 255. For FortiBalancer2, we will leave its priority at 100. In a two-node cluster, this is permissible. Though, when you include more nodes in your cluster, you will need to set a unique priority for each VIP to properly communicate and fail-over. To do this, use the following command:

```
FortiBalancer1(config)#cluster virtual priority port1 100 255
FortiBalancer2(config)#cluster virtual priority port1 100 100
```

Note: The state is the backup on FortiBalancer2. This is expected since it is of lower priority than the master.

➢ Step 7 Turn on the clustering

```
FortiBalancer1(config)#cluster virtual on
FortiBalancer2(config)#cluster virtual on
```

➢ Step 8 Turn on fast failover

```
FortiBalancer1(config)#cluster virtual ffo on
FortiBalancer1(config)#cluster virtual ffo interface carrier loss timeout 1000
FortiBalancer2(config)#cluster virtual ffo on
FortiBalancer2(config)#cluster virtual ffo interface carrier loss timeout 1000
```

4.3.1.2 Active-Active: Two Nodes

Configuration Guidelines

In Active-Active mode, node 1 will be the master for VIP1, and the backup for VIP2. Node 2 will act as the master for VIP2, and serve as the backup for VIP1. This increases the performance of your site while maintaining high availability.

The next illustration shows a typical deployment. To achieve active-active status, we need to have two virtual cluster IDs (VCID), each containing at least one VIP.

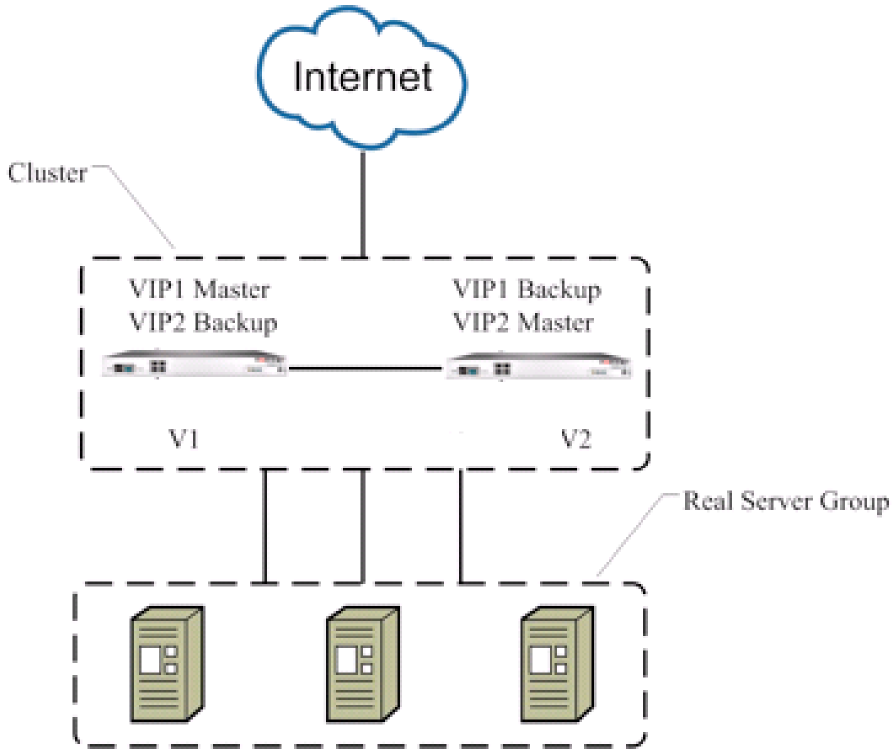


Figure 4-5 Active-Active Two-Node Architecture

In the above figure, FortiBalancer1 is the master for VIP1 and the backup for VIP2 and FortiBalancer2 is the master for VIP2 and the backup for VIP1.

VCID 1 will have VIP1 (192.168.2.100) and VCID 2 will have VIP2 (192.168.2.101).

Table 4-2 General Settings of Active-Active Two-Node Clustering

Operation	Command
Configure SLB	Refer to the SLB Configuration section.
Configure a virtual interface	**cluster virtual ifname** *<interface_name> <cluster_id>*
Configure virtual cluster authentication	**cluster virtual auth** *<interface_name> <cluster_id> {0|1} [password]*
Configure preemption	**cluster virtual preempt** *<interface_name> <cluster_id> <mode>*
Configure virtual IP	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
Configure priority	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
Enable the virtual cluster	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*

Configuration Example for Active-Active SLB Clustering via CLI

We will setup node 1 as the master of VIP1 and the backup of VIP2. Node 2 will be the master of VIP2 and the backup for VIP1.

- Step 1 Configure SLB for both FortiBalancer1 and FortiBalancer2

```
FortiBalancer1(config)#slb real http “server1” 192.168.1.50 80 1000 tcp 1 1
FortiBalancer1(config)#slb real http “server2” 192.168.1.51 80 1000 tcp 1 1
FortiBalancer1(config)#slb group method “group1” rr
FortiBalancer1(config)#slb group member “group1” “server1” 1
FortiBalancer1(config)#slb group member “group1” “server2” 1
FortiBalancer1(config)#slb virtual http “vip1” 192.168.2.100 80
FortiBalancer1(config)#slb virtual http “vip2” 192.168.2.101 80
FortiBalancer1(config)#slb policy default “vip1” “group1”
FortiBalancer1(config)#slb policy default “vip2” “group1”

FortiBalancer2(config)#slb real http “server1” 192.168.1.50 80 1000 tcp 1 1
FortiBalancer2(config)#slb real http “server2” 192.168.1.51 80 1000 tcp 1 1
FortiBalancer2(config)#slb group method “group1” rr
FortiBalancer2(config)#slb group member “group1” “server1” 1
FortiBalancer2(config)#slb group member “group1” “server2” 1
FortiBalancer2(config)#slb virtual http “vip1” 192.168.2.100 80
FortiBalancer2(config)#slb virtual http “vip2” 192.168.2.101 80
FortiBalancer2(config)#slb policy default “vip1” “group1”
FortiBalancer2(config)#slb policy default “vip2” “group1”
```

- Step 2 Configure a virtual interface name

```
FortiBalancer1(config)#cluster virtual ifname “port1” 100
FortiBalancer1(config)#cluster virtual ifname “port1” 101
FortiBalancer2(config)#cluster virtual ifname “port1” 100
FortiBalancer2(config)#cluster virtual ifname “port1” 101
```

- Step 3 Configure virtual cluster authentication

It is recommended that you run clustering with an authentication string to avoid unauthorized participation in your cluster.

```
FortiBalancer1(config)#cluster virtual auth port1 100 0
FortiBalancer1(config)#cluster virtual auth port1 101 0
FortiBalancer2(config)#cluster virtual auth port1 100 0
FortiBalancer2(config)#cluster virtual auth port1 101 0
```

- Step 4 Configure virtual cluster preemption

```
FortiBalancer1(config)#cluster virtual preempt port1 100 1
FortiBalancer1(config)#cluster virtual preempt port1 101 0
FortiBalancer2(config)#cluster virtual preempt port1 100 0
FortiBalancer2(config)#cluster virtual preempt port1 101 1
```

- Step 5 Define the VIP by the “**cluster virtual vip**” command

```
FortiBalancer1(config)#cluster virtual vip “port1” 100 192.168.2.100
FortiBalancer1(config)#cluster virtual vip “port1” 101 192.168.2.101
FortiBalancer2(config)#cluster virtual vip “port1” 100 192.168.2.100
FortiBalancer2(config)#cluster virtual vip “port1” 101 192.168.2.101
```

- Step 6 Define the priority

Cluster priority determines which node becomes the master. The node with highest priority becomes the master.

```
FortiBalancer1(config)#cluster virtual priority port1 100 255
FortiBalancer1(config)#cluster virtual priority port1 101 100
FortiBalancer2(config)#cluster virtual priority port1 100 100
FortiBalancer2(config)#cluster virtual priority port1 101 255
```

- Step 7 Turn on the clustering

```
FortiBalancer1(config)#cluster virtual on
FortiBalancer2(config)#cluster virtual on
```

4.3.2 Clustering Inside Interfaces

Clustering on the inside requires a little different train of thought than that of clustering the SLB VIPs.

Note: NATing is highly recommended if the machines in your inside network need to communicate to other networks via the FortiBalancer appliance.

There are two methods of setting up the inside interface. The first is to use one VIP that will belong to one of the appliances in the Virtual Cluster. If you want to or need to share the load between the nodes you will have to setup an Active-Active configuration for the inside interfaces. We will cover how to setup both scenarios in this section.

4.3.2.1 Active-Standby (One VIP)

Configuration Guidelines

In Active-Standby mode, one box will serve as the gateway for the inside network. Upon unexpected failure of the master node, the standby node in the cluster will take over. For our purpose, we are going to pick an unused IP address on the inside network (192.168.1.3) and use it as the gateway for our inside network.

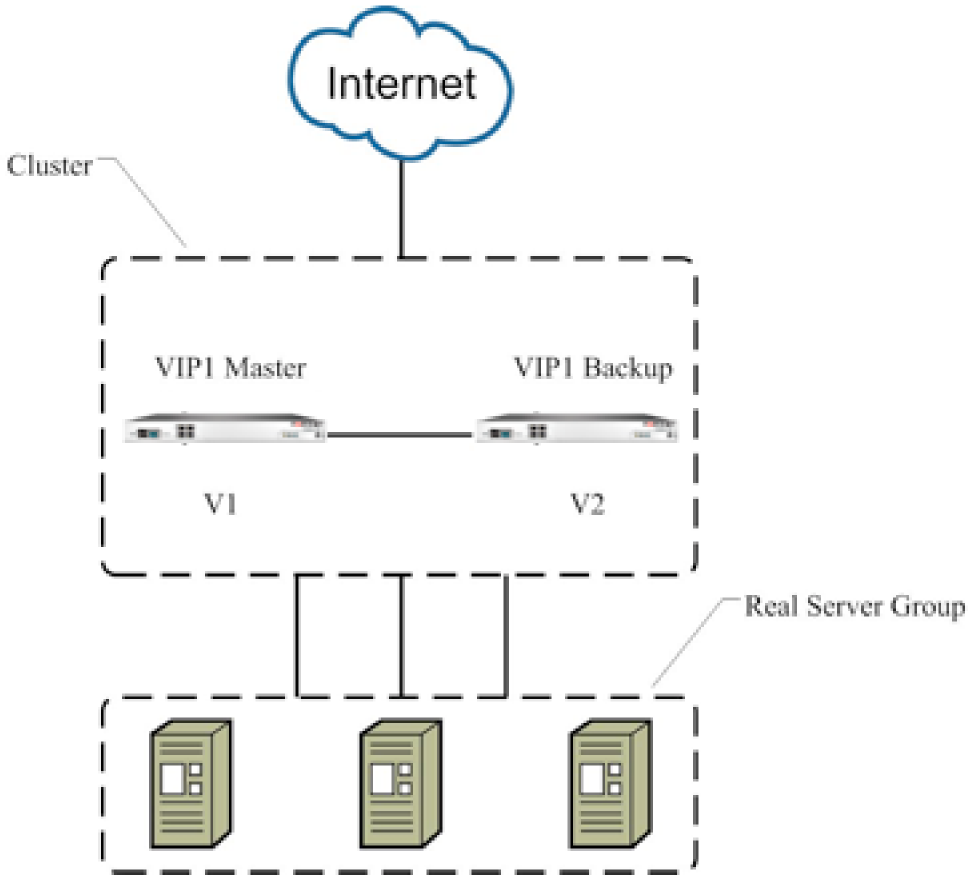


Figure 4-6 Inside Interface Active-Standby Mode

Table 4-3 General Settings of Inside Interface Active-Standby Clustering

Operation	Command
Configure a virtual interface	**cluster virtual ifname** *<interface_name> <cluster_id>*
Configure virtual IP	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
Configure priority	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
Enable the virtual cluster	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*

Configuration Example for Active-Standby Clustering Inside Interface via CLI

➢ Step 1 Configure a virtual interface and its cluster ID

```
FortiBalancer1(config)#cluster virtual ifname “port2” 100
FortiBalancer2(config)#cluster virtual ifname “port2” 100
```

➢ Step 2 Define the VIP by the “**cluster virtual vip**” command

```
FortiBalancer1(config)#cluster virtual vip “port2” 100 192.168.1.3
FortiBalancer2(config)#cluster virtual vip “port2” 100 192.168.1.3
```

➢ Step 3 Define the priority

Cluster priority determines which node becomes the master. The node with highest priority becomes the master.

```
FortiBalancer1(config)#cluster virtual priority port2 100 255
FortiBalancer2(config)#cluster virtual priority port2 100 100
```

➢ Step 4 Turn on the clustering

```
FortiBalancer1(config)#cluster virtual on
FortiBalancer2(config)#cluster virtual on
```

4.3.2.2 Active-Active (Two VIPs)

Configuration Guidelines

In Active-Active configuration, we will create two VIPs to serve as gateways. Half of your servers' default routes will point to the first VIP and the other half will point to the second VIP, thus equally dividing the load between the FortiBalancer appliances.

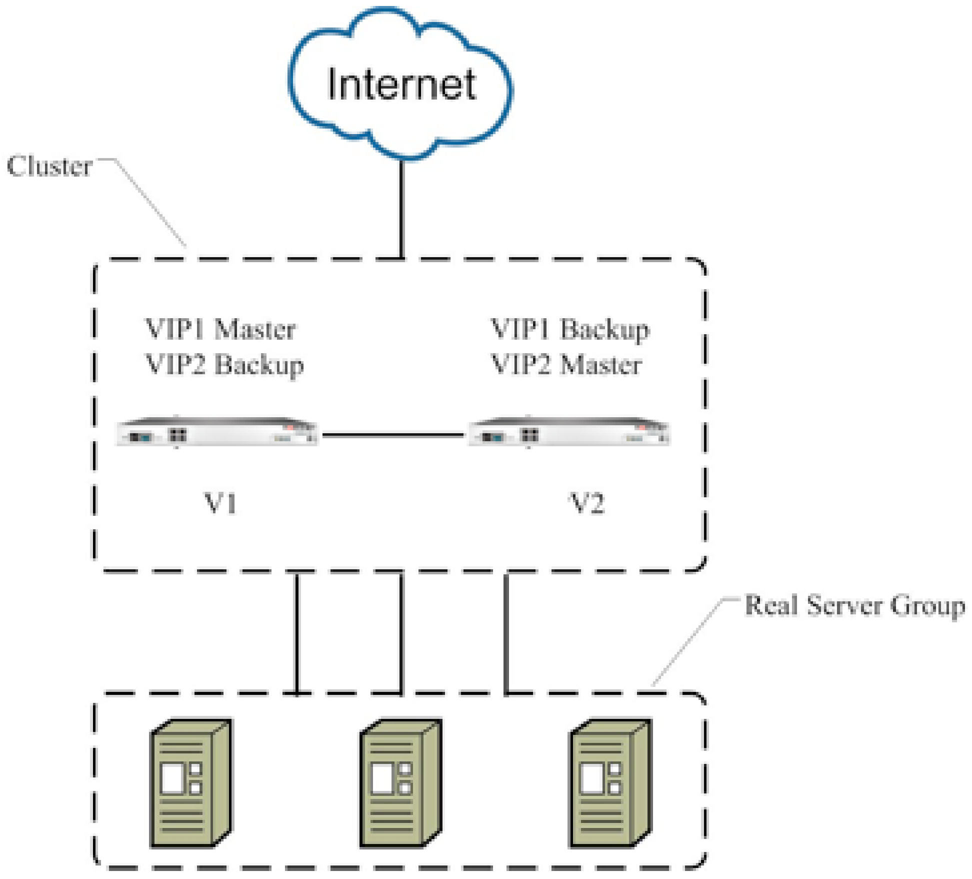


Figure 4-7 Inside Interface Active- Active Mode

Table 4-4 General Settings of Inside Interface Active-Active Clustering

Operation	Command
Configure a virtual interface	**cluster virtual ifname** *<interface_name> <cluster_id>*
Configure virtual IP	**cluster virtual vip** *<interface_name> <cluster_id> <vip>*
Configure priority	**cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
Enable the virtual cluster	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*

Configuration Example for Active-Active Clustering Inside Interface via CLI

We proceed along these lines by executing the following:

➢ Step 1 Configure a virtual interface and its cluster ID

```
FortiBalancer1(config)#cluster virtual ifname "port2" 100
```

```
FortiBalancer1(config)#cluster virtual ifname “port2” 101
FortiBalancer2(config)#cluster virtual ifname “port2” 100
FortiBalancer2(config)#cluster virtual ifname “port2” 101
```

- Step 2 Define the VIP by the “**cluster virtual vip**” command

```
FortiBalancer1(config)#cluster virtual vip “port2” 100 192.168.1.3
FortiBalancer1(config)#cluster virtual vip “port2” 101 192.168.1.4
FortiBalancer2(config)#cluster virtual vip “port2” 100 192.168.1.3
FortiBalancer2(config)#cluster virtual vip “port2” 101 192.168.1.4
```

- Step 3 Define the priority

Cluster priority determines which node becomes the master. The node with highest priority becomes the master.

```
FortiBalancer1(config)#cluster virtual priority port2 100 255
FortiBalancer1(config)#cluster virtual priority port2 101 100
FortiBalancer2(config)#cluster virtual priority port2 100 100
FortiBalancer2(config)#cluster virtual priority port2 101 255
```

- Step 4 Turn on the clustering

```
FortiBalancer1(config)#cluster virtual on
FortiBalancer2(config)#cluster virtual on
```

Chapter 5 High Availability (HA)

5.1 Overview

As the network applications develop, customers have higher and higher requirements for the reliability of the network and network appliances. During network planning and design, to improve the reliability of the network, some critical network appliances must have redundancy protection mechanisms. The Clustering function mentioned in the “Clustering” chapter uses the VRRP technology to solve the single-point failure. This chapter will introduce the High Availability (HA) function that newly provided by FortiBalancer appliances. The HA function not only solves the single-point failure, but also provides more policies to ensure the network reliability.

The HA function allows two or more FortiBalancer appliances to continuously exchange the running status with each other, and keep their configurations synchronized. When an appliance becomes down, other available appliances will take over the application services on the faulty appliance, which ensures the high availability of application services.

Besides, the HA function provides the Stateful Session Failover (SSF) function. With the SSF function, when a service failover occurs, connections on the service will be switched to the new appliance. This avoids the interruption of connections and therefore improves user experience.

The HA function can be deployed flexibly. Besides the Active/Active and Active/Standby deployment scenarios, the HA function can be deployed among multiple appliances to achieve mutual-backup.

5.2 HA Basics

5.2.1 HA Domain and Unit

The HA domain comprises a group of appliances that provide the HA function. The appliances in the HA domain are called unit. Each HA domain can comprise a maximum of 32 units.

5.2.2 Floating IP Group

Usually, the service Active/Standby failover on a unit is achieved by using the floating IP address. The same floating IP address can be defined on multiple units. However, the floating IP address on only one unit can be in the “Active” state at the same time.

To ensure the consistency and flexibility of service failover, HA technology groups the floating IP addresses and switches floating IP addresses by group. The floating IP addresses can be switched only after they are added to a floating IP group. At the same moment, the status of all floating IP addresses in the same floating IP group are the same. The status is also called the status of the floating IP group.

The status of a floating IP group is determined by the group priority, failover mode, and results of the health checks related to the group. After the floating IP group is configured correctly, the HA module will the check the running environment of the group based on the configured health check conditions. Based the health check results, the group status can be one of the following two types:

- Active/Standby: The results of all health checks related to the group are “Up”, indicating the group is ready to provide services. In this case, the group status is “Active” or “Standby”. If the group status is “Active”, this unit will obtain all the floating IP addresses of the group and provide services. If the group status is “Standby”, this unit will provides backup for services and will take over services in case of service failover.
- Init: Initial group status. If the result of any health checks related to the group is “Down”, the group status is “Init”, which indicates that this unit is not qualified for provides services of the group. Even if service failover occurs on the group, this unit cannot take over services.

Note: When the group status is "Init", check the group configurations or the health check results to make the group status change to "Active" or "Standby" so that the unit will provide services or backup for services.

On one unit, multiple floating IP groups can be configured. The status of every groups are independent from each other. If all groups on a unit need to be switched over together, the "Unit_Failover" mode (see the section "Failover Rules") is required.

The floating IP address are configured by using the "**ha group fip**" or "**ha group fiprange**" command. For details, please see the FortiBalancer 8.4 CLI Reference.

5.2.3 Group Failover Mode

The HA function supports two group failover modes: non-preempt and preempt modes.

When a floating IP group is enabled on multiple units:

- If the non-preempt mode is enabled, the group status on the local unit will not change until a failover occurs.
- In the preempt mode, if the group priority on the local unit is higher than those of all peer units, the group status on the local unit will be forcibly switched to "Active". If the group status on a peer unit was "Active" before this, its group status will be forcibly switched to "Standby".

5.2.4 Floating MAC

The HA function supports the floating MAC function. With this function enabled, the floating MAC address (configured by using the command "**ha floatmac mac**") is switched to the interface of the new unit on which the group status is "Active". In this way, after group status switches, the clients will not be aware that the appliance that provides the application services has been changed, because the MAC addresses of the appliances that provide application services before and after group status switch remain unchanged.

Note:

- ➢ By default, the floating MAC function is disabled. Before this function is enabled, the HA function must be first disabled by executing the command "**ha off**".
- ➢ The parameter "interface_name" in the command "**ha floatmac mac**" determines that the floating MAC function takes effect on the floating IP group with which the interface is associated. The floating MAC addresses configured for different floating IP groups cannot be the same.

5.2.5 HA Deployment Scenarios

The HA function can be deployed flexibly. Besides the Active/Active and Active/Standby deployment scenarios, the HA function can be deployed among multiple appliances to achieve mutual-backup.

- Active/Active deployment scenario: The HA domain comprises two units; on each unit, there are "Active" floating IP groups and "Standby" floating IP groups, the status of which are "Active" on the peer unit. The HA domain comprises two units. On each unit, there is "Active" floating IP groups, and the status of the "Active" group on the peer unit is "Standby".
- Active/Standby deployment scenario: The HA domain comprises two units; the status of all floating IP groups are "Active" on one unit and are "Standby" on the other unit.
- When the HA function is deployed among multiple appliances to achieve mutual-backup, the HA domain comprises multiple units to provide services or backup for services. Among these scenarios, the "N+1" deployment scenario is commonest one. In the "N+1" deployment scenario, the HA domain comprises N+1 units. Among these

units, the group status on N units are all “Active” and all the group status on the remaining one are “Standby”.

5.3 Reliable Communication Links

The units in an HA domain can use the following three types of communication links to exchange their status messages to ensure the high reliability of the communication.

- Fast Failover (FFO) Link
- Primary Link
- Secondary Link

The FFO link is established by directly connecting two HA units’ FFO ports through a dedicated FFO cable. Therefore, it can only be used for the Active/Active and Active/Standby deployment scenarios with two units. By default, the FFO link is disabled. The main functions of the FFO link are as follows:

- Heartbeat packet transmission: When the HA function is running, the local unit can use the FFO link to send the heartbeat packets to detect the peer unit’s status.
- Bootup Synconfig: When the FFO link and Bootup Synconfig are both enabled, the local unit can synchronize the HA unit and link configurations from the peer unit.
- Fast Failover: When the local unit is down, the peer unit can perform fast failover through the FFO link.

The primary and secondary links are also called network links, because both of them connect the two units through ordinary network cables. Only one primary link can be established between any two units, while at most 31 secondary links are allowed between any two units. By defaults, the network links are enabled.

After adding multiple units for an HA domain, the system will establish primary link connections between each two units automatically. The main functions of the primary link are as follows:

- Heartbeat packet transmission: The local unit can send the heartbeat packets to its peer units through the primary link to detect the peer units’ status.
- Bootup Synconfig: With both Bootup Synconfig and the HA function enabled, the local unit can synchronize the configurations (except HA link configurations) from the peer units. This behavior is the same as executing the command “synconfig from”.
- Runtime Synconfig: With Runtime Synconfig enabled, when the configurations (such as HA, SLB and IP pool) of the local unit are modified, the unit can synchronize the modifications to the peer units via the primary link. This behavior is the same as executing the command “synconfig to”.

The secondary link is optional and just used for heartbeat packets transmission. The administrator has to manually set up the same secondary link configurations on the local unit and the peer units. Please be noted that to establish a secondary link between two units, you need to configure a secondary link with the same ID on the two units respectively.

For example, the IP address of two HA units “u1” and “u2” are 192.168.1.1 and 192.168.1.2 respectively. To establish a secondary link between the two units, the following two commands must be executed on both units:

```
FortiBalancer(config)#ha link network secondary u1 1 192.168.1.1 65521
FortiBalancer(config)#ha link network secondary u2 1 192.168.1.2 65521
```

After the above configurations are finished on both units, a secondary link with ID “1” is established between “u1” and “u2”.

The table below shows the differences and similarities among the three types of HA communication links.

Table 5-1 Differences and Similarities Among the HA Links

Item		FFO Link	Primary Link	Secondary Link
Differences	**Connection Method**	Directly connected through dedicated FFO cable.	Networked or directly connected through ordinary network cables.	
	Required Configuration	No configuration required.	The primary link is automatically established after the local unit and peer units join an HA domain.	The same secondary link configurations have to be manually set up on the local and the peer units.
	Application Scenario	Only applicable to Active/Active and Active/Standby scenarios.	All scenarios.	
	Way of Joining the HA Domain	Enable the FFO link and the HA function.	1. Configure the IP addresses of the local unit and the peer units. 2. Enable the HA function.	N/A.
Similarities		1. All the three types of HA links can be used to transmit heartbeat packets. The HA units send the heartbeat packets to exchange their health check conditions and group status. 2. In the Active/Active and Active/Standby scenarios, the three types of HA links can be backup for one another. 3. When all the three types of HA links become down, the peer unit will be considered as failed.		

5.4 Failover Rules

The HA function is capable of performing health check on the system status and network condition in an HA domain. Once any failure is detected by health check and it matches one of the pre-configured failover conditions, the corresponding failover action will be taken. Usually, the system will select another unit which is with the highest priority among the available units and change the status of the floating IP group enabled on that unit to be "Active" forcibly. To do this, HA provides failover rules to control the switchover of group status.

Failover rules are defined by associating failover conditions with failover actions. Failover conditions indicate the monitoring status on system hardware or software, such as network interface status, CPU utilization and so on. Failover actions are the operations to be performed by the system when the associated failover conditions occur. HA provides three failover actions:

- Group_Failover: Switch over the status of the floating IP group. For this action, the system will select a new unit based on the health condition and group priority, and change the status of the floating IP group enabled on that unit to be "Active" to take over the services.
- Unit_Failover: Switch over the status of all the floating IP groups enabled on a unit.
- Reboot: Switch over the status of all the floating IP groups enabled on a unit, and then restart the unit.

To facilitate use of administrators, HA also provides built-in network connectivity check to detect network exceptions, such as network interface failure and network interruption among units. Once any of these exceptions occur, the system will perform failover actions automatically.

Note: Only when the network connections of all interfaces in a bond interface become down, will the "Group_Failover" action be taken for the floating IP group to which the IP addresses of the bond interface belong.

While providing built-in failover rules, the HA function also allows administrators to manually configure multiple failover rules. To do this, the following software or hardware health check conditions can be configured as failover conditions:

- Hardware:
  - CPU overheat health check condition
  - SSL card health check condition
  - Port health check condition
- Software:
  - CPU utilization health check condition
  - ATCP zone memory utilization health check condition
  - System memory health check condition
  - Network packet memory health check condition
  - Process health check condition
- Network condition:
  - Gateway health check condition

In some complex application environments, more complicated failover rules are required. For example, theoretically, in the environment with a bond interface deployed, only when the network connections of all interfaces in the bond interface become down will failover action be taken. However, in practical application, it is required to take failover action as long as the network connection of one interface becomes down. To meet this kind of complicated applications, HA further introduces the concept of health check condition group (vcondition). A vcondition comprises multiple health check sub-conditions. A sub-condition can be a real health check condition or another vcondition, which further comprises sub-conditions. The logical relationship among multiple sub-conditions can be either "AND" or "OR". To apply vcondition to the above application, administrators can first define health check conditions for each of the network interfaces in a bond interface, and combine these conditions into a vcondition by setting the logical relationship to "OR". Then, associate the vcondition with some failover actions.

5.5 Configuration Synchronization

The HA function provides configuration synchronization to simplify configurations on units and ensure the consistency of configurations on all units in an HA domain. HA supports two kinds of configuration synchronization: Bootup Synconfig and Runtime Synconfig. The two configuration synchronization functions can be both enabled.

5.5.1 Bootup Synconfig

Bootup Synconfig is to synchronize the configurations of the peer unit to the local unit after the local unit logs into the HA domain.

The local unit can log into the HA domain in two methods:

- FFO Login: The unit logs into the HA domain through the FFO link.
- Network Login: The unit logs into the HA domain through the network link.

To use the FFO Login method, administrators need to perform the following operations on the local unit:

1. Execute the "**ha link ffo on**" command to enable the FFO link.
2. Execute the "**ha synconfig bootup on**" command to enable the Bootup Synconfig function of HA.

Note: After the FFO link and the Bootup Synconfig function are enabled, the local unit will start to synchronize configurations about HA units and links from the peer unit.

3. Execute the “**ha link network on**” command to enable the network links.
4. Execute the “**ha on**” command to enable the HA function.

To use the Network Login method, administrators need to perform the following operations on the local unit:

1. Execute the “**ha unit**” command to add the local unit and the peer unit.
2. Execute the “**ha synconfig bootup on**” command to enable the Bootup Synconfig function of HA.
3. Execute the “**ha link network on**” command to enable the network links.
4. Execute the “**ha on**” command to enable the HA function.

Note: In Bootup Synconfig, only the configurations saved by executing the command “**write memory**” can be synchronized.

5.5.2 Runtime Synconfig

Runtime Synconfig is to synchronize the add/deletion/change of configurations on the local unit to other units in the same HA domain automatically while the HA is being running. This ensures that the configurations on all units in one HA domain are always the same.

Note: To synchronize the configuration changes from the local unit to the peer units, please make sure that Runtime Synconfig is enabled on both the local unit and the peer units

5.6 Stateful Session Failover (SSF)

The Stateful Session Failover (SSF) function can be applied to both the Active/Active and Active/Standby HA scenarios. With SSF enabled, the information about the TCP and UDP connections established on the “Active” floating IP group will be updated to all “Standby” floating IP groups in real time. Once any failover action is taken, all the existing TCP and UDP connections will not be interrupted because the connection information has been updated to the new “Active” unit by the SSF function. However, if the SSF function is disabled, the existing TCP and UDP connections will be interrupted.

The SSF function supports TCP, UDP, FTP and IP types of SLB applications as well as NAT applications. It can be enabled or disabled per virtual service.

Note:

- To ensure the SSF function works well, please make sure that the HA-related configurations on all the units in one HA domain are the same. It is recommended to use Runtime Synconfig while the SSF function is enabled.
- The SSF function uses a stable network link between two HA units to transmit SSF session information. If the network link used for SSF goes down, session information cannot be exchanged between two units. If a group failover occurs subsequently, the existing connections might be reset.

5.7 HA Logging

The HA function provide logging function. By default, the HA logging function is disabled. If the HA function is enabled, the logging function will be enabled too. If the HA function is disabled, the logging function will be disabled too.

The HA logging function allows administrators to set the level of the HA logs that the system generates. Eight levels HA logs are supported: emerg, alert, crit, err, warning, notice, info, and debug. Once the level of HA logs is specified, the log messages lower than this level will be ignored, that is will not be recorded in the system. The default log level is info.

5.8 Configuration Examples

The HA function can be deployed in the following typical scenarios:

- Scenario 1: Active/Standby
- Scenario 2: Active/Active
- Scenario 3: N+1

The following sections describe the configuration examples for all the three scenarios.

5.8.1 Scenario 1: Active/Standby

5.8.1.1 Configuration Objectives

The Active/Standby deployment scenario can be used to achieve the following configuration objectives:

- The HA domain contains two HA units, each of which is enabled with the same floating IP group.
- The Floating IP group contains the VIP addresses of two application services.
- Unit 1 provides application services, while unit 2 provides backup for such services.
- Fast failover is carried out through the FFO link.

The following figure shows the network topology for the preceding configuration objectives.

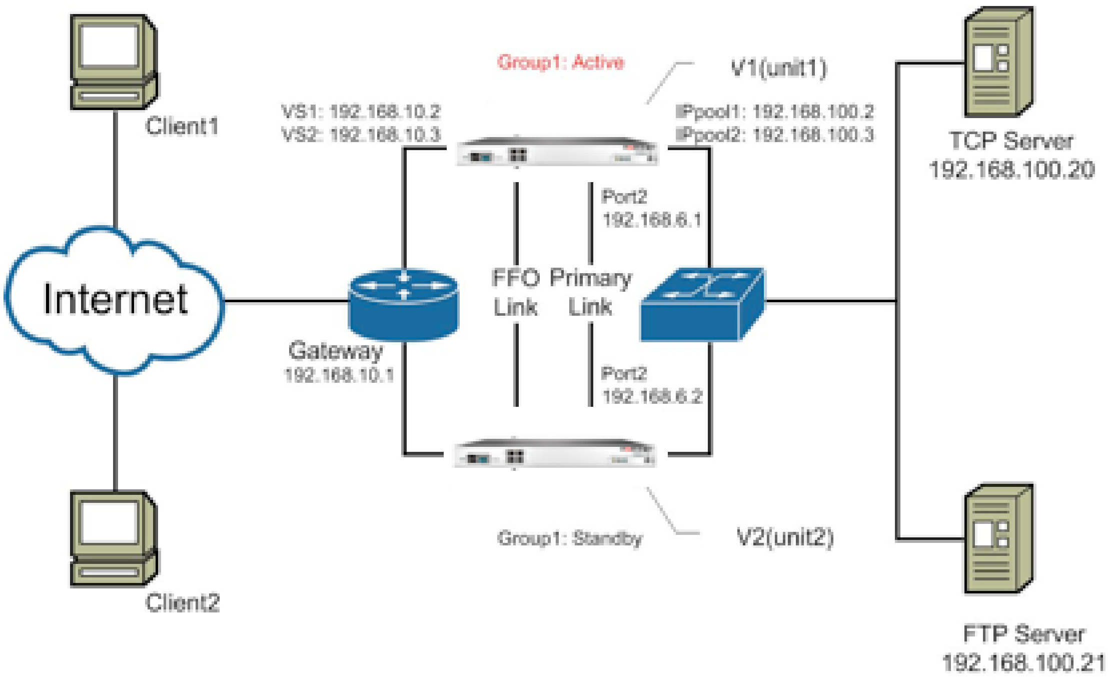


Figure 5-1 Active/Standby Deployment Scenario

5.8.1.2 Configuration Examples

➢ **FortiBalancer1:**

1. Execute the following commands to complete SLB configurations:

```
FortiBalancer(config)#slb real tcp "r1" 192.168.100.20 80 65535 tcp 3 3
FortiBalancer(config)#slb real ftp "r2" 192.168.100.21 21 65535 tcp 3 3
FortiBalancer(config)#slb group method "slb_g1" rr
FortiBalancer(config)#slb group method "slb_g2" rr
FortiBalancer(config)#slb group member "slb_g1" "r1" 1 0
FortiBalancer(config)#slb group member "slb_g2" "r2" 1 0
FortiBalancer(config)#slb virtual tcp "v1" 192.168.10.2 80 arp 0
FortiBalancer(config)#slb virtual ftp "v2" 192.168.10.3 21 0
FortiBalancer(config)#slb policy default "v1" "slb_g1"
FortiBalancer(config)#slb policy default "v2" "slb_g2"
FortiBalancer(config)#ip pool p1 192.168.100.2 192.168.100.2
FortiBalancer(config)#ip pool p2 192.168.100.3 192.168.100.3
FortiBalancer(config)#slb proxyip group slb_g1 p1
FortiBalancer(config)#slb proxyip group slb_g2 p2
```

2. Execute the following commands to configure HA units and links:

```
FortiBalancer(config)#ha unit "unit1" 192.168.6.1 65521
FortiBalancer(config)#ha unit "unit2" 192.168.6.2 65521
FortiBalancer(config)#ha link network on
FortiBalancer(config)#ha link ffo on
```

3. Execute the following commands to configure floating IP group:

```
FortiBalancer(config)#ha group id 1
FortiBalancer(config)#ha group fip 1 192.168.10.2 port1
FortiBalancer(config)#ha group fip 1 192.168.10.3 port1
FortiBalancer(config)#ha group fip 1 192.168.100.2 port3
FortiBalancer(config)#ha group fip 1 192.168.100.3 port3
FortiBalancer(config)#ha group priority unit1 1 10
FortiBalancer(config)#ha group priority unit2 1 5
FortiBalancer(config)#ha group preempt on 1
FortiBalancer(config)#ha group enable 1
```

4. (Optional) Execute the following commands to configure health check conditions, taking the health check for gateway and CPU utilization as examples.

```
FortiBalancer(config)#ha hc gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc cpu utilization 90 5000 3 3
FortiBalancer(config)#ha hc vcondition name vcondition1 V_1 AND
FortiBalancer(config)#ha hc vcondition member vcondition1 GATEWAY_1
FortiBalancer(config)#ha hc vcondition member vcondition1 CPU_UTIL
```

5. (Optional) Execute the following command to add failover rules:

```
FortiBalancer(config)#ha decision rule vcondition1 Group_Failover 1
```

6. (Optional) Execute the following commands to enable the SSF function:

```
FortiBalancer(config)#ha ssf peer 192.168.6.2
FortiBalancer(config)#ha ssf on
```

7. (Optional) Execute the following commands to set the configuration synchronization mode:

```
FortiBalancer(config)#ha synconfig bootup on
FortiBalancer(config)#ha synconfig runtime on
```

8. (Optional) Execute the following command to enable the HA logging function:

```
FortiBalancer(config)#ha log on
```

9. Execute the following commands to enable the HA function and save the HA configurations to the memory:

```
FortiBalancer(config)#ha on
FortiBalancer(config)#write memory
```

- **FortiBalancer2：**

In the Active/Standby scenario, it is recommended to use the FFO link and primary link to synchronize configuration information from the peer unit.

1. Execute the following commands to enable the FFO function and the Bootup Synconfig mode:

```
FortiBalancer(config)#ha link ffo on
FortiBalancer(config)#ha synconfig bootup on
```

2. (Optional) Execute the following commands to enable the SSF function:

```
FortiBalancer(config)#ha ssf peer 192.168.6.1
FortiBalancer(config)#ha ssf on
```

3. Execute the following command to enable the HA function:

```
FortiBalancer(config)#ha on
```

   Once the HA function is enabled, unit2 (FortiBalancer2) will join the HA domain. After unit2 joins the HA domain, it first synchronizes configuration information about HA units, FFO link and network links through the FFO link and then synchronizes other configurations through the primary link from unit1 (FortiBalancer1).

5.8.2 Scenario 2: Active/Active

5.8.2.1 Configuration Objectives

The Active/Active deployment scenario can be used to achieve the following configuration objectives:

- The HA domain contains two HA units and provides two floating IP groups.
- Each floating IP group contains the VIP address of one application service.
- Unit1 provides the application service for group1, while unit2 provides the application service for group2. Unit1 and unit2 provide backup for each other.
- Fast failover is carried out through the FFO link.

The following figure shows the network topology for the preceding configuration objectives.

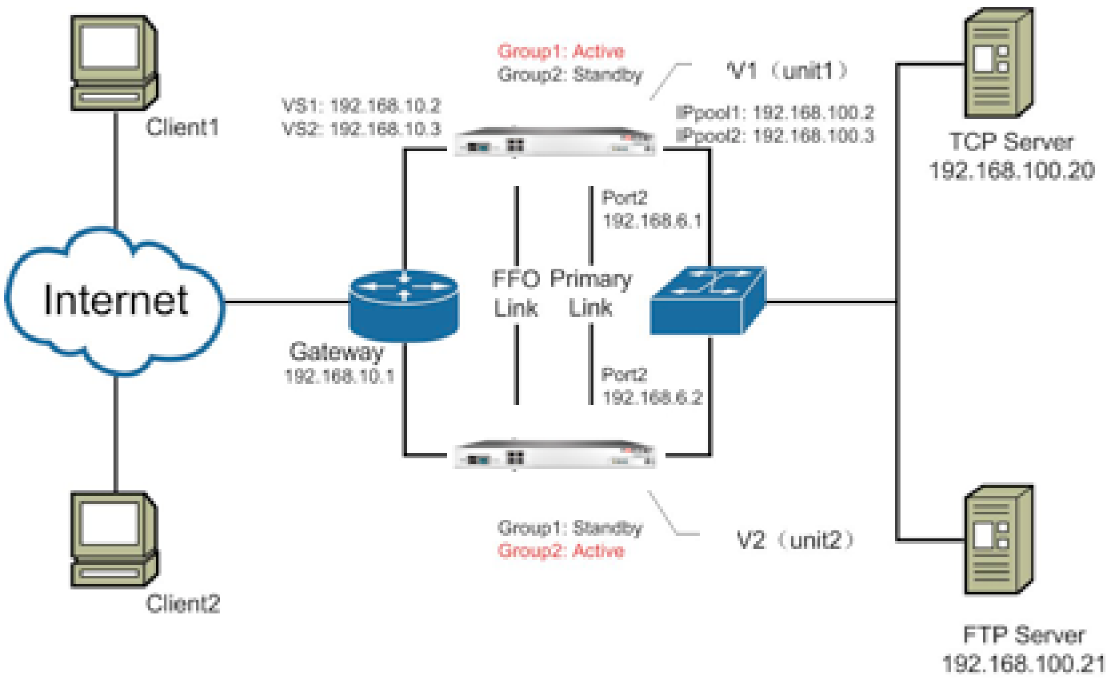


Figure 5-2 Active/Active Deployment Scenario

5.8.2.2 Configuration Examples

➢ **FortiBalancer1:**

1. Execute the following commands to complete SLB configurations:

```
FortiBalancer(config)#slb real tcp "r1" 192.168.100.20 80 65535 tcp 3 3
FortiBalancer(config)#slb real ftp "r2" 192.168.100.21 21 65535 tcp 3 3
FortiBalancer(config)#slb group method "slb_g1" rr
FortiBalancer(config)#slb group method "slb_g2" rr
FortiBalancer(config)#slb group member "slb_g1" "r1" 1 0
FortiBalancer(config)#slb group member "slb_g2" "r2" 1 0
FortiBalancer(config)#slb virtual tcp "v1" 192.168.10.2 80 arp 0
FortiBalancer(config)#slb virtual ftp "v2" 192.168.10.3 21 0
FortiBalancer(config)#slb policy default "v1" "slb_g1"
FortiBalancer(config)#slb policy default "v2" "slb_g2"
FortiBalancer(config)#ip pool p1 192.168.100.2 192.168.100.2
FortiBalancer(config)#ip pool p2 192.168.100.3 192.168.100.3
FortiBalancer(config)#slb proxyip group slb_g1 p1
FortiBalancer(config)#slb proxyip group slb_g2 p2
```

2. Execute the following commands to configure HA units and links:

```
FortiBalancer(config)#ha unit "unit1" 192.168.6.1 65521
FortiBalancer(config)#ha unit "unit2" 192.168.6.2 65521
FortiBalancer(config)#ha link network on
FortiBalancer(config)#ha link ffo on
```

3. Execute the following commands to configure floating IP groups:

```
FortiBalancer(config)#ha group id 1
FortiBalancer(config)#ha group fip 1 192.168.10.2 port1
FortiBalancer(config)#ha group fip 1 192.168.100.2 port3
FortiBalancer(config)#ha group priority unit1 1 10
FortiBalancer(config)#ha group priority unit2 1 5
FortiBalancer(config)#ha group preempt on 1
FortiBalancer(config)#ha group enable 1
FortiBalancer(config)#ha group id 2
FortiBalancer(config)#ha group fip 2 192.168.10.3 port1
FortiBalancer(config)#ha group fip 2 192.168.100.3 port3
FortiBalancer(config)#ha group priority unit1 2 5
FortiBalancer(config)#ha group priority unit2 2 10
FortiBalancer(config)#ha group preempt on 2
FortiBalancer(config)#ha group enable 2
```

4. (Optional) Execute the following command to configure health check conditions, taking the health check for gateway and CPU utilization as examples.

```
FortiBalancer(config)#ha hc gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc cpu utilization 90 5000 3 3
FortiBalancer(config)#ha hc vcondition name vcondition1 V_1 AND
FortiBalancer(config)#ha hc vcondition member vcondition1 GATEWAY_1
FortiBalancer(config)#ha hc vcondition member vcondition1 CPU_UTIL
```

5. (Optional) Execute the following command to add failover rules:

```
FortiBalancer(config)#ha decision rule vcondition1 Unit_Failover
```

6. (Optional) Execute the following commands to enable the SSF function:

```
FortiBalancer(config)#ha ssf peer 192.168.6.2
FortiBalancer(config)#ha ssf on
```

7. (Optional) Execute the following commands to set the configuration synchronization mode:

```
FortiBalancer(config)#ha synconfig bootup on
FortiBalancer(config)#ha synconfig runtime on
```

8. (Optional) Execute the following command to enable the HA logging function:

```
FortiBalancer(config)#ha log on
```

9. Execute the following commands to enable the HA function and save the HA configurations to the memory:

```
FortiBalancer(config)#ha on
FortiBalancer(config)#write memory
```

- **FortiBalancer2:**

In the Active/Active scenario, it is recommended to use the FFO link and primary link to synchronize configuration information from the peer unit.

1. Execute the following commands to enable the FFO function and the Bootup Synconfig mode:

```
FortiBalancer(config)#ha link ffo on
FortiBalancer(config)#ha synconfig bootup on
```

2. (Optional) Execute the following commands to enable the SSF function:

```
FortiBalancer(config)#ha ssf peer 192.168.6.1
FortiBalancer(config)#ha ssf on
```

3. Execute the following command to enable the HA function:

```
FortiBalancer(config)#ha on
```

Once the HA function is enabled, unit2 (FortiBalancer2) will join the HA domain. After unit2 joins the HA domain, it first synchronizes configuration information about HA units, FFO link and network links through the FFO link and then synchronizes other configurations through the primary link from unit1 (FortiBalancer1).

5.8.3 Scenario 3: N+1

In the N+1 deployment scenario, the HA domain contains N+1 units. On N units, the status of the floating IP groups are all "Active", while on the remaining one unit, the status of the floating IP groups are all "Standby". This section will introduce the configuration objectives and examples for the "3+1" deployment scenario.

5.8.3.1 Configuration Objectives

The "3+1" deployment scenario can be used to achieve the following configuration objectives:

- The HA domain contains four HA units and provides three floating IP groups.
- Each floating IP group contains the VIP address of a virtual service.
- Unit1 to unit3 provide the virtual services of group1 to group3 respectively, while unit4 provides backup for unit1 to unit3.

The following figure shows the network topology for the preceding configuration objectives.

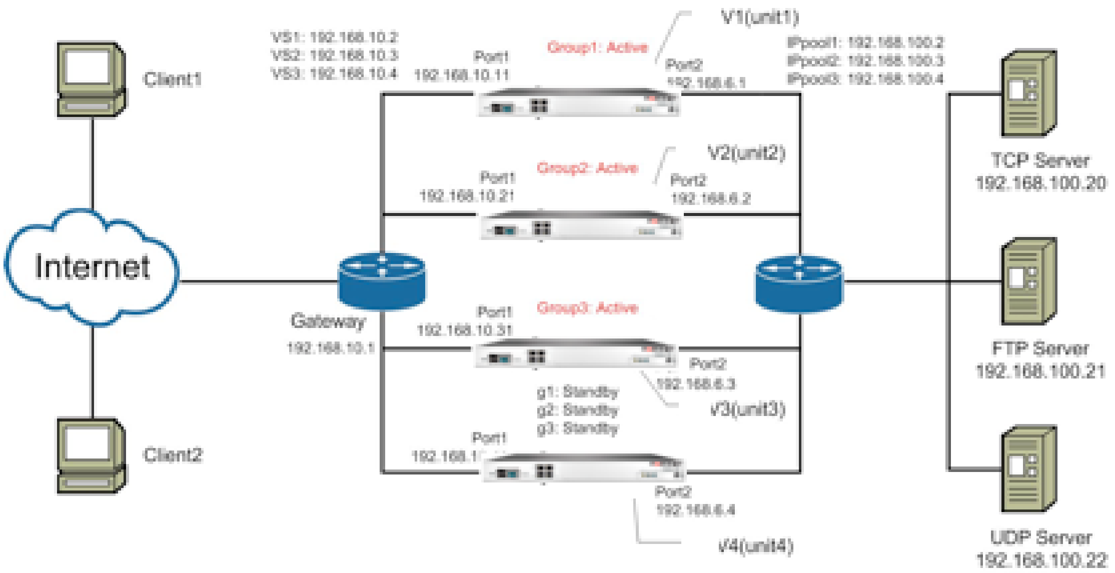


Figure 5-3 N+1 Deployment Scenario

5.8.3.2 Configuration Examples

➢ **FortiBalancer1:**

1. Execute the following commands to complete SLB configurations:

```
FortiBalancer(config)#slb real tcp "r1" 192.168.100.20 80 65535 tcp 3 3
FortiBalancer(config)#slb real ftp "r2" 192.168.100.21 21 65535 tcp 3 3
FortiBalancer(config)#slb real udp "r3" 192.168.100.22 53 65535 3 3 60 icmp
FortiBalancer(config)#slb group method "slb_g1" rr
FortiBalancer(config)#slb group method "slb_g2" rr
FortiBalancer(config)#slb group method "slb_g3" rr
FortiBalancer(config)#slb group member "slb_g1" "r1" 1 0
FortiBalancer(config)#slb group member "slb_g2" "r2" 1 0
FortiBalancer(config)#slb group member "slb_g3" "r3" 1 0
FortiBalancer(config)#slb virtual tcp "v1" 192.168.10.2 80 arp 0
FortiBalancer(config)#slb virtual ftp "v2" 192.168.10.3 21 0
FortiBalancer(config)#slb virtual udp "v3" 192.168.10.4 53 arp 0
FortiBalancer(config)#slb policy default "v1" "slb_g1"
FortiBalancer(config)#slb policy default "v2" "slb_g2"
FortiBalancer(config)#slb policy default "v3" "slb_g3"
FortiBalancer(config)#ip pool p1 192.168.100.2 192.168.100.2
FortiBalancer(config)#ip pool p2 192.168.100.3 192.168.100.3
FortiBalancer(config)#ip pool p3 192.168.100.4 192.168.100.4
FortiBalancer(config)#slb proxyip group slb_g1 p1
FortiBalancer(config)#slb proxyip group slb_g2 p2
FortiBalancer(config)#slb proxyip group slb_g3 p3
```

2. Execute the following commands to configure HA units and links:

```
FortiBalancer(config)#ha unit "unit1" 192.168.6.1 65521
FortiBalancer(config)#ha unit "unit2" 192.168.6.2 65521
FortiBalancer(config)#ha unit "unit3" 192.168.6.3 65521
FortiBalancer(config)#ha unit "unit4" 192.168.6.4 65521
FortiBalancer(config)#ha link network secondary unit1 1 192.168.10.11
FortiBalancer(config)#ha link network secondary unit2 1 192.168.10.21
FortiBalancer(config)#ha link network secondary unit3 1 192.168.10.31
FortiBalancer(config)#ha link network secondary unit4 1 192.168.10.41
FortiBalancer(config)#ha link network on
```

3. Execute the following commands to configure floating IP groups:

```
FortiBalancer(config)#ha group id 1
FortiBalancer(config)#ha group fip 1 192.168.10.2 port1
FortiBalancer(config)#ha group fip 1 192.168.100.2 port3
FortiBalancer(config)#ha group priority unit1 1 200
FortiBalancer(config)#ha group priority unit2 1 100
FortiBalancer(config)#ha group priority unit3 1 50
FortiBalancer(config)#ha group priority unit4 1 150
FortiBalancer(config)#ha group preempt on 1
FortiBalancer(config)#ha group enable 1

FortiBalancer(config)#ha group id 2
FortiBalancer(config)#ha group fip 2 192.168.10.3 port1
FortiBalancer(config)#ha group fip 2 192.168.100.3 port3
FortiBalancer(config)#ha group priority unit1 2 50
FortiBalancer(config)#ha group priority unit2 2 200
FortiBalancer(config)#ha group priority unit3 2 100
FortiBalancer(config)#ha group priority unit4 2 150
FortiBalancer(config)#ha group preempt on 2
FortiBalancer(config)#ha group enable 2

FortiBalancer(config)#ha group id 3
FortiBalancer(config)#ha group fip 3 192.168.10.4 port1
FortiBalancer(config)#ha group fip 3 192.168.100.4 port3
FortiBalancer(config)#ha group priority unit1 3 100
FortiBalancer(config)#ha group priority unit2 3 50
FortiBalancer(config)#ha group priority unit3 3 200
FortiBalancer(config)#ha group priority unit4 3 150
FortiBalancer(config)#ha group preempt on 3
FortiBalancer(config)#ha group enable 3
```

4. (Optional) Execute the following command to configure health check conditions, taking the health check for gateway and CPU utilization as examples.

```
FortiBalancer(config)#ha hc gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc gateway unit3 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc gateway unit4 192.168.10.1 GATEWAY_1 1000 3 3
FortiBalancer(config)#ha hc cpu utilization 90 5000 3 3
FortiBalancer(config)#ha hc vcondition name vcondition1 V_1 AND
FortiBalancer(config)#ha hc vcondition member vcondition1 GATEWAY_1
FortiBalancer(config)#ha hc vcondition member vcondition1 CPU_UTIL
```

5. (Optional) Execute the following command to add failover rules:

```
FortiBalancer(config)#ha decision rule vcondition1 Unit_Failover
```

6. (Optional) Execute the following commands to set the configuration synchronization mode:

```
FortiBalancer(config)#ha synconfig bootup on
FortiBalancer(config)#ha synconfig runtime on
```

7. (Optional) Execute the following command to enable the HA logging function:

```
FortiBalancer(config)#ha log on
```

8. Execute the following commands to enable the HA function and save the HA-related configurations to the memory:

```
FortiBalancer(config)#ha on
FortiBalancer(config)#write memory
```

- **FortiBalancer2:**

1. Execute the following commands to configure HA units and links:

```
FortiBalancer(config)#ha unit "unit1" 192.168.6.1 65521
FortiBalancer(config)#ha unit "unit2" 192.168.6.2 65521
FortiBalancer(config)#ha unit "unit3" 192.168.6.3 65521
FortiBalancer(config)#ha unit "unit4" 192.168.6.4 65521
FortiBalancer(config)#ha link network secondary unit1 1 192.168.10.11
FortiBalancer(config)#ha link network secondary unit2 1 192.168.10.21
FortiBalancer(config)#ha link network secondary unit3 1 192.168.10.31
FortiBalancer(config)#ha link network secondary unit4 1 192.168.10.41
FortiBalancer(config)#ha link network on
```

2. Execute the following command to enable the Bootup Synconfig mode:

```
FortiBalancer(config)#ha synconfig bootup on
```

3. Execute the following command to enable the HA function:

```
FortiBalancer(config)#ha on
```

Once the HA function is enabled, unit2 (FortiBalancer2) will join the HA domain and start to synchronize configuration information from unit1 (FortiBalancer1).

- **FortiBalancer3:**

1. Execute the following commands to configure HA units and links:

```
FortiBalancer(config)#ha unit "unit1" 192.168.6.1 65521
FortiBalancer(config)#ha unit "unit2" 192.168.6.2 65521
FortiBalancer(config)#ha unit "unit3" 192.168.6.3 65521
FortiBalancer(config)#ha unit "unit4" 192.168.6.4 65521
FortiBalancer(config)#ha link network secondary unit1 1 192.168.10.11
FortiBalancer(config)#ha link network secondary unit2 1 192.168.10.21
FortiBalancer(config)#ha link network secondary unit3 1 192.168.10.31
FortiBalancer(config)#ha link network secondary unit4 1 192.168.10.41
FortiBalancer(config)#ha link network on
```

2. Execute the following command to enable the Bootup Synconfig mode:

```
FortiBalancer(config)#ha synconfig bootup on
```

3. Execute the following command to enable the HA function:

```
FortiBalancer(config)#ha on
```

   Once the HA function is enabled, unit3 (FortiBalancer3) will join the HA domain and start to synchronize configuration information from unit1 (FortiBalancer1).

- **FortiBalancer4:**

1. Execute the following commands to configure HA units and links:

```
FortiBalancer(config)#ha unit "unit1" 192.168.6.1 65521
FortiBalancer(config)#ha unit "unit2" 192.168.6.2 65521
FortiBalancer(config)#ha unit "unit3" 192.168.6.3 65521
FortiBalancer(config)#ha unit "unit4" 192.168.6.4 65521
FortiBalancer(config)#ha link network secondary unit1 1 192.168.10.11
FortiBalancer(config)#ha link network secondary unit2 1 192.168.10.21
FortiBalancer(config)#ha link network secondary unit3 1 192.168.10.31
FortiBalancer(config)#ha link network secondary unit4 1 192.168.10.41
FortiBalancer(config)#ha link network on
```

2. Execute the following command to enable the Bootup Synconfig mode:

```
FortiBalancer(config)#ha synconfig bootup on
```

3. Execute the following command to enable the HA function:

```
FortiBalancer(config)#ha on
```

   Once the HA function is enabled, unit4 (FortiBalancer4) will join the HA domain and start to synchronize configuration information from unit1 (FortiBalancer1).

Chapter 6 Server Load Balancing (SLB)

6.1 Overview

SLB (Server Load Balancing) allows you to distribute load and traffic to specific groups of servers or to a specific server. The FortiBalancer appliance supports server load balancing in Layers 2-7 of the OSI network model. Layer 2 SLB is based on network interfaces. Layer 3 SLB works on IP addresses. Layer 4 SLB is mostly concerned with port based load balancing. Layer 7 is used when you want to perform load balancing based on URLs, HTTP headers or Cookies. The basic steps for setting up SLB are:

1. Define the real servers.
2. Define a group load balancing method.
3. Add real servers to the group.
4. Define a Virtual IP to listen for requests.
5. Bind the group balancing method to the Virtual IP.

The real server, the VIP and the virtual service are the fundamental components of SLB deployment.

- The real server is an application server hosting varied applications or services. It processes the requests from the client side.
- The VIP in general is a public IP address that can be accessed from the external clients. As an entrance, it receives and forwards external requests, and sends the processed results from the real servers back to the client side.
- For the Layer 4 and Layer 7 SLB, the virtual service is commonly represented with a VIP/port pair and can be accessed by the external clients to get their target network resources. For example, if a client wants to access some Web resources by a predefined VIP or a Web site name (with DNS), all the requests from this client will go through the VIP and be sent out to different real servers by the FortiBalancer appliance hosting the VIP and real servers. With the virtual service, the internal network architecture and backend real servers are hidden from the external clients by only exposing the VIP address.

The remainder of this chapter will cover these steps and cover some examples of Layer 2, Layer 3, Layer 4 and Layer 7 load balancing strategies.

This following figure is a logical overview of load balancing using the FortiBalancer appliance.

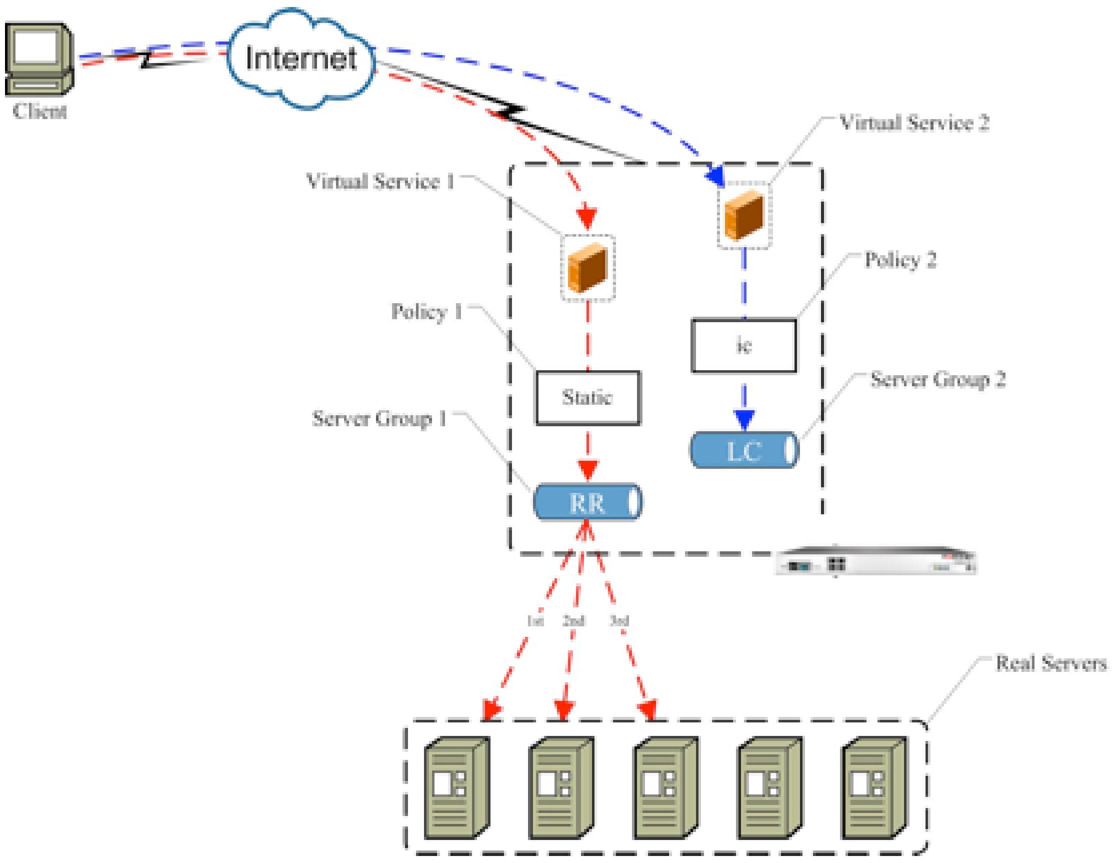


Figure 6-1 SLB Architecture

6.2 Understanding SLB

6.2.1 SLB Methods

The FortiBalancer appliance allows several methods of load balancing. We will briefly discuss certain methods below and situations where you would want to use them. Later in the chapter we will go over and setup each metric in detail.

Table 6-1 SLB Methods

SLB Methods	Description
Round Robin (rr)	If we have three servers in Group 1with two in Group 2, and chose round robin as our metric, each request would follow the real servers in order [1,2,3, 1, 2, 3…] for Group 1 and [4,5, 4, 5…] for Group 2.
Least Connections (lc)	This metric tells SLB to select the real server with the fewest number of active connections.
Shortest Response (sr)	The server with the shortest response time will get the next request. Using this metric you can intermix fast servers with slow servers and the fast servers will get more hits initially. As load increases, and response time increases and the slower servers will start to field more requests.
Persistent IP (pi)	This metric ties the source IP of the request to the real server processing the request. An example application for this metric is when doing e-commerce transactions and a specific server needs to maintain state about the client's transaction. Keep in mind that if a large ISP deploys a Mega-Proxy, one real server could service thousands of requests. (A Mega-Proxy is used to proxy all client requests from a single IP.)
Persistent Cookie (pc)	Persistent cookie is used to associate a cookie name/value pair with a single real server on your backend. When setting up cookie based policies keep in mind that you need a default policy for requests that do not have

SLB Methods	Description
	cookies in the HTTP header.
Insert Cookie (ic)	Dynamically inserts cookies to allow the OS to maintain persistence to a server.
Rewrite Cookie (rc)	Rewrite Cookie rewrites server side cookies on the fly thereby allowing the backend servers to maintain persistence to a client.
Proximity (prox)	This method is based on SDNS proximity info and used by redirect policy only. It directs SLB to select the real server, which has lowest proximity distance with the request IP.
SNMP (snmp)	This method is based on the real server's SNMP (Simple Network Management Protocol) MIB information regarding the server's status and availability, e.g. the server's CPU status and memory usage.
Embed Cookie (ec)	It embeds some information in the server side cookies, allowing the backend servers to maintain persistence to a client.
Hash Query (hq)	It keeps the persistence of the session by hashing the specified tag value in the query of HTTP requests, and must work with persistent url policy.

Additional Load Balancing methods include: Persistent URL (pu), Persistent Hostname (ph), Hash Cookie (hc), Hash Header (hh), SSL SID (sslsid), Hash IP (hi), Consistent Hash IP (chi), Consistent Hash RADIUS User Name (radchu), Consistent Hash RADIUS Session ID (radchs), and persistence (Individual Session Persistence).

For more information on these additional SLB methods, please consult the FortiBalancer CLI Reference.

6.2.2 SLB Policies

Policies are used to tie virtual services to groups. By using policies, administrators may control how load balancing decisions are made by different layer policies rules (Layer 2-7). A virtual service is bound with a group by a policy. A single group can be shared among different virtual services. In the following pages we will cover policies in depth.

SLB supports multiple policy types (see the table below):

Table 6-2 SLB Policies

Basic Policy	Persistent Policy	QoS Policy
Redirect Static Default Backup	Persistent URL Persistent Cookie Rewrite Cookie Insert Cookie Hash URL RADIUS Username RADIUS Session ID	QoS Cookie QoS Hostname QoS URL QoS Network QoS Clientport QoS Body Regular Expression Header

Different types of policies have different priorities. Currently, multiple FortiBalancer appliance SLB policies can be configured for one SLB virtual service and the FortiBalancer appliance will route requests based on the first matched policy with the highest priority. The following is the order of priority that FortiBalancer appliance SLB will follow:

Table 6-3 SLB Policy Priority

Priority	Policy
a	redirect
b	static
c	qos-clientport - qos client port
	qos-network - qos network
	pu - persistent url
	rc - rewrite cookie

Priority	Policy
	ic - insert cookie
	pc - persistent cookie
	qos-cookie - qos cookie
	qos-hostname - qos hostname
	qos-url - qos url
	qos-body- qos body
	regex - regex url
	header - header
	hu - hash url
	raduname - radius username
	radsid - radius session id
d	default
e	backup

For Layer 4 SLB, only two SLB policies (qos network and qos clientport) can be used beside static, default and backup policy.

For backward compatibility, the default policy precedence is the same as before. To configure specific precedence and associate it with specific virtual service, the system administrator can use the following CLI commands:

slb policy order *<order_template_name> <policy_type> <precedence>*

slb virtual order *<virtual_service> <order_template_name>*

6.2.2.1 Policy Nesting

SLB Policy Nesting is the mechanism for different policies to be nested to perform server load balancing.

In traditional SLB policy mechanisms, one policy is defined to bind a virtual service with a group. For example, in the command "**slb policy qos url policy1 vs1 group1 'news' 1**", the virtual service vs1 and group1 are associated by policy1. The requests that match the policy1 will be forwarded to group1.

In SLB Policy Nesting mechanism, a new object "vlink" is defined for policy nesting. A virtual service or a vlink can be associated with a group or another vlink via policies. The requests will be distributed to a group or a vlink according to the policies. For a request sent to a vlink, the system will distribute the request to a group or another vlink associated to the vlink according to the policies. Therefore, the requests will be distributed to a group associated to a vlink only when they match two or more nested policies.

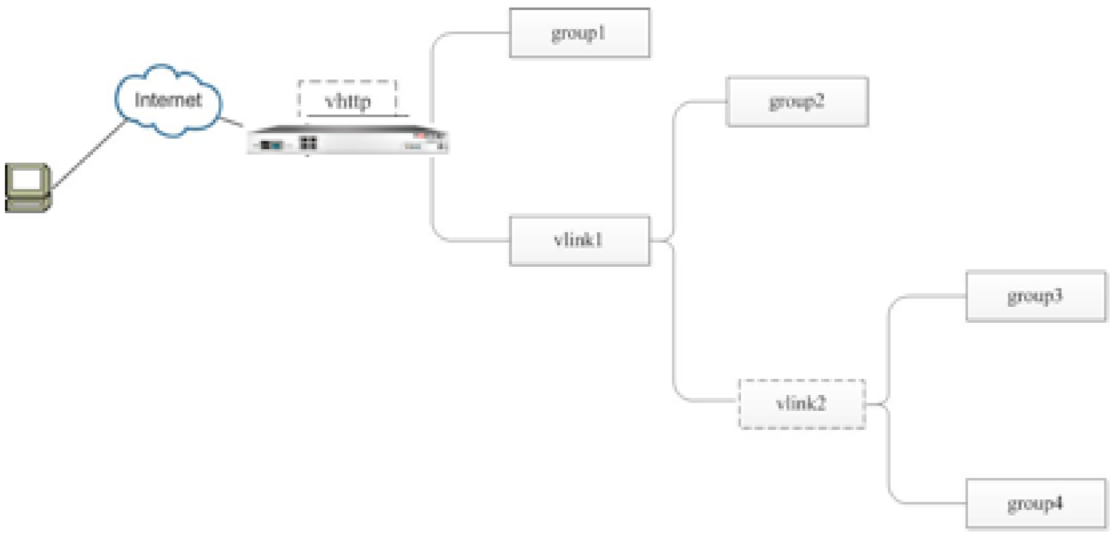


Figure 6-2 Policy Nesting

Limitations of SLB Policy Nesting:

- Some policies cannot be nested, such as static, redirect, filetype and external policies.
- The icookie, rcookie, persistent cookie and persistent URL policies can only associate a virtual service or a vlink with a group, but cannot be associated to a virtual service or a vlink with another vlink.
- To simplify the configuration, the policy nesting layer should be less than or equal to 3. If more than 3 layers are configured, the system will return a 503 error.
- Only HTTP and HTTPS protocols are supported.

6.2.3 SLB Session Persistence

The SLB module of the FortiBalancer appliance introduces a new, generally applied, and session ID-based method for session persistence. This persistence method can either operate independently on Layer 4/7 SLB or work in collaboration with an existing Layer 7 SLB persistence policy. In the latter case, the existing policy extracts the session ID and the persistence method implements the session persistence.

The persistence method maintains a mapping table to keep track of each session ID and its associated real server. Furthermore, the persistence method dynamically processes the timeout information for each session ID to ensure independence of each session.

6.2.3.1 Session ID Types

The persistence method supports the following types of session IDs:

- ip: This session ID type is the client IP address that the persistence method extracts from the client request.
- ip+port: This session ID type is the client IP address and port number that the persistence method extracts from the client request.
- string: This session ID type is a specified string that the persistence method either extracts from the client request (or real server response) itself or obtains through extraction by an existing SLB policy. You may also specify an offset and ID length for more granular control of this session ID type.

6.2.3.2 Obtaining the Session ID

Direct extraction by the Persistence Method

When used independently, the persistence method can directly extract the following session ID types from the client request or real server response:

- Client IP address or IP address and port number from the client request.
- A specified string from the HTTP URL query, HTTP cookie, HTTP header, or HTTP body field in the client request.
- A specified string from the HTTP cookie, HTTP header, or HTTP body field in the server response. In this case, you also need to configure the FortiBalancer appliance to obtain the matching specified string from the HTTP URL query, HTTP cookie, HTTP header, or HTTP body field in the client request.

Note:

When multiple session IDs of the string type are configured, the FortiBalancer appliance will obtain the session ID in priority-descending order from the HTTP URL query, HTTP cookie, HTTP header, and HTTP body.

Indirect extraction through an existing SLB Persistence Policy

In collaboration mode, the persistence method can indirectly obtain the session ID through the following types of SLB persistence policies:

- Header
- Persistent cookie
- Persistent url
- Qos body

Note:

When a client request matches with both the persistence method and an SLB persistence policy, the FortiBalancer appliance uses only the persistence method to obtain the session ID and perform session persistence.

For more information about the collaboration of the persistence method with existing SLB persistence policies, refer to example "Persistence Method Collaborating with an SLB Persistence Policy".

6.2.3.3 Session ID Timeout Management

The persistence method can dynamically processes the timeout information of each session ID using one of the following modes:

- idle: If the FortiBalancer appliance receives no new client requests within the specified "idle" time period, the FortiBalancer appliance will terminate the client's session and clear its associated session ID.
- duration: If the specified "duration" of time has passed since a client's session was created, the FortiBalancer appliance will terminate the client's session and clear its associated session ID regardless of the arrival of any new requests.

Note:

To dynamically manage the timeout information of session IDs, you need to specify the management mode to ensure that the FortiBalancer appliance clears outdated session IDs and records and new ones.

In addition, the persistence method can statically reserve a session ID. Requests matching a statically reserved session ID will be consistently sent to the same real server.

6.2.4 SLB Health Check

The FortiBalancer appliance provides three types of health checks:

Basic health check

Basic health checks determine the application, service or server availability (Up/Down) status using a specified protocol format. FortiBalancer supports basic health check types that include ARP, ICMP (ping), TCP, TCPS, DNS, HTTP and HTTPS. The default health check is assigned with the individual real server (e.g. representing any backend physical or virtual server in the server farm) configuration. FortiBalancer will perform basic health checks on the IP/port pair of the real server to decide the real service availability. This is also called main health check.

Additional health check

Many times application or service infrastructure components are spread across multiple server types that need to be run simultaneously. For example, an application might require Web servers, application servers, and database servers to be run simultaneously. In this case determining the health of one server is not sufficient, and health checks must be performed on the entire suit of servers for determining the health of entire application infrastructure. In addition to the basic health check, multiple diverse health checks can be configured for one real server. The AND or OR relationship for multiple additional health checks can be set. FortiBalancer supports configuring additional health check name, which helps administrators easily configure and identify the additional health check. See the "**health relation**" command.

Script health check

There are many applications and non-standard protocols (in addition to standard TCP/IP based protocols) that follow specific request/response sequence for communication purposes. To support custom application availability check, customers can make use of scripted health check. FortiBalancer offers a generic application aware health check that runs over a TCP or UDP connection. It determines an application's health by exchanging messages with the application in the specified format. Multiple application messages, request/response sequences can be scripted for emulating normal application communication between FortiBalancer and applications.

Two health check types are available, "script-tcp" and "script-udp" for building generic script health checks. Script health checks support the following advanced application health checks: FTP, SMTP, LDAP, RADIUS, POP3, DNS, and TELNET applications. Additional application health check can be built by importing application request and response into the FortiBalancer appliance.

6.2.4.1 Methods of Health Check

Health Check allows the FortiBalancer appliance to perform diagnostic observations on the performance of Web servers and Web server farms associated with each FortiBalancer appliance. These observations on the performance health of the real server will allow the appliance to know how the servers are performing and which backend servers can best handle the incoming client requests.

ICMP Health Check

It is a limited health check method that simply sends an ICMP echo (ping) to the server. If the server responds with an ICMP reply then the server is marked as "up". The server is marked as "down" otherwise. This does NOT check for the running service or the quality of the service.

TCP Health Check

TCP health check simply opens a TCP connection to a specific port of the real server. If that connection fails, the server will be marked as "down". The server will be marked as "up" if the TCP connection succeeds. This health check does not indicate if the service is actually functioning. For checking if a particular service in question is functioning correctly, a specific health check must be used (e.g. HTTP health check for Web applications).

For Layer 2 SLB, the TCP health check method needs to configure a health check reflector on another FortiBalancer appliance. The reflector will open a port and listen upon the port, and responds to health check requests. In this way, the health check process can go through the real server and check the entire link status. If the reflector is able to respond to the health check request, the real server will be marked as "UP", else marked as "DOWN".

TCPS Health Check

TCPS health check provides an SSL health check for SLB real servers. If the SSL handshake fails, the server will be marked as "down". If the SSL handshake succeeds, the server will be marked as "up". This health check function will check for the availability of the real service by opening an SSL connection to a specific port of the real server or defaults to 443.

HTTP Health Check

The basic built-in HTTP health check opens a TCP connection and sends an HTTP request with one of the HTTP methods (such as GET, POST, etc.) pre-defined in the health request table. The FortiBalancer appliance expects the health response as defined in the response table. The default index chosen to reference request/response table is 0. If the response is not satisfied with the conditions configured in the response table, the server will be marked as “down”, otherwise it is marked as “up”.

HTTPS Health Check

HTTPS Health Check provides an SSL health check for real servers. If the SSL handshake succeeds, the FortiBalancer will send a pre-defined HTTP request with proper method format to real servers. If the response from the real server is the same as the expected response, the real server will be marked as “up”; otherwise, it is marked as “down”. When using HTTPS Health Check, users should pre-define HTTP requests/methods and corresponding expected responses for matching purposes. When using client certificates, the imported client certificate must be encoded by DER rules during client authentication.

Script-TCP Health Check & Script-UDP Health Check

Script-TCP and script-UDP are provided for the generic script health check. They run over TCP and UDP connection respectively. Only when the “hc_type” is set to these two types, the health check list can work while doing health check.

Script-TCPS Health Check

Script-TCPS Health Check provides an SSL health check for HTTPS real servers. It works the same way as script-TCP Health Check once the SSL handshake succeeds.

DNS Health Check

DNS health check is one of the built-in application health checks for DNS service that uses scripted health check. DNS health check opens a UDP connection and sends a DNS request to a destination DNS server, and the FortiBalancer expects a special DNS response. The DNS request and response are not configurable because they are unchangeable. If the required conditions are satisfied, then server will be marked as “UP”, otherwise, the server will be marked as “DOWN”.

Radius-Auth Health Check & Radius-Acct Health Check

Radius-Auth health check and Radius-Acct health check are provided for checking the availability of the RADIUS servers.

LDAP Health Check

FortiBalancer supports health check on commonly used LDAP servers like Windows AD, OpenLDAP and SunOne Directory, to better meet the customers’ needs for health check on LDAP binding and search operations.

The LDAP additional health check is only supported for TCP real servers.

RTSP-TCP Health Check

RTSP health check opens a TCP connection and sends an RTSP “OPTIONS” (Get available methods on the streaming server) request to a RTSP real server. If the real server responds with any of the RFC-defined RTSP status codes, then the server will be marked as “up”, otherwise, it will be marked as “down”.

SIP-UDP & SIP-TCP Health Check

SIP health check opens a UDP or TCP connection and sends a SIP “OPTIONS” request to a real server. This request is used to ask the SIP server for the list of SIP methods it supports. The response may contain a set of capabilities (i.e. audio/video codecs) of the responding SIP server. If the real server responds with RFC-defined methods, the server will be marked as “up”, otherwise, it will be marked as “down”.

6.2.4.2 HC Checker and HC Checker List

When the method of health check is set as script_tcp or script_udp, HC checker and HC checker list can work while the FortiBalancer appliance does health check. An HC checker is defined as one transaction of health check. It consists of sending one message and receiving one response. A list of HC checkers can compose an HC checker list, which is identified by the HC checker list name.

Below are the commands used to define the HC checker and HC checker list:

health checker *<checker_name> <request_index ><response_index> [timeout] [flag]*

health list *<list_name>*

health member *<list_name> <checker_name> [place_index]*

health app *{real_name|add_hc_name} <list_name> [frequency] [hc_localip] [hc_localport]*

6.2.4.3 HTTP Requests and Responses

By default the OS defines an HTTP health table of HTTP requests and HTTP responses to be used by the HTTP health check. The default index inside the health table for HTTP requests and responses is “0, 0”. The “**show health request**” command shows the table of requests defined. Likewise “**show health response**” shows the responses.

For our example model:

```
FortiBalancer(config)#health on
FortiBalancer(config)#show health request
Row   Request
0     HEAD / HTTP/1.0
1     HEAD / HTTP/1.0
2     HEAD / HTTP/1.0
3     HEAD / HTTP/1.0
4     HEAD / HTTP/1.0
5     HEAD / HTTP/1.0
6     HEAD / HTTP/1.0
7     HEAD / HTTP/1.0
8     HEAD / HTTP/1.0
9     HEAD / HTTP/1.0
10    HEAD  /HTTP/1.0
11    HEAD / HTTP/1.0
12    HEAD  /HTTP/1.0
13    HEAD  /HTTP/1.0
14    HEAD  /HTTP/1.0
15    HEAD  /HTTP/1.0
FortiBalancer(config)#show health response
Row   Response:
0     200 OK
1     200 OK
2     200 OK
3     200 OK
4     200 OK
5     200 OK
6     200 OK
7     200 OK
8     200 OK
9     200 OK
10    200 OK
11    200 OK
12    200 OK
13    200 OK
14    200 OK
15    200 OK
```

Index 0,0 in the health request table results with the following request being sent to the real server:

```
HEAD / HTTP/1.0
```

The response needed from the real server is:

```
200 OK
```

You may change the response header to accommodate your network. Refer to your Web server's documentation on what valid HTTP responses you may use.

By default, we use a HEAD request as the health check. A HEAD request will only ask for the HEADER information for the object being requested. If we were to use the GET request, you will get the entire page back as a response. If you have a large site or content that is fairly large you should choose the HTTP response that will cause the least amount of overhead.

Changing the HTTP Health Request/Response

You can define your own HTTP requests and the responses to be used by the HTTP health check. For example, you may simply change the request to get a CGI script that returns an HTTP status 200 OK when the database server is up and a 404 NOT FOUND when the database server is "down". Below are the commands to use to perform this task as well as an example from our network model.

health request *<request_index> <request_string>*

health response *<response_index> <response_string>*

health server *{real_name|add_hc_name} <req_index> <res_index>*

For our network example:

➢ Step 1 Define your own HTTP request

```
FortiBalancer(config)#health request 1 "GET /cgi-bin/dbstatus.pl"
```

➢ Step 2 Set the request to index 1, the response to index 0 for server2http

```
FortiBalancer(config)#health server server2http 1 0
```

Note: Index 0 in health response means returning 200 OK.

The health check provided by the OS starts with simple TCP health checks and allows you to perform advanced health checks by utilizing different health requests and responses.

Keyword Health Check for Web Page

HTTP health check supports keyword matching for the specific real server response Web page. The HTTP health check daemon will check the real server's health by searching a keyword in the server's response content. If the keyword is found, this health check is successful. Otherwise, this health check fails.

Let's begin a configured example for this enhancement:

The real server is **10.3.16.188:88**

The Web page is **index.txt**

The key word is **fortibalanceradmin**

➢ Step 1 Add a real service with HTTP health check

```
FortiBalancer(config)#slb real http rs 10.3.16.188 88 1000 http 3 3
```

➢ Step 2 Configure HTTP health check

```
FortiBalancer(config)#health request 1 "GET /index.txt HTTP/1.0\r\n\r\n"
FortiBalancer(config)#health response 1 "fortibalanceradmin"
```

- Step 3 Associate the configured HTTP health check with the real service

```
FortiBalancer(config)#health server rs 1 1
```

Note: Keyword HTTP health check can only support ASCII string search in the Web page, but not support double-byte keyword (e.g. simplified Chinese, traditional Chinese).

6.2.4.4 TCP-based SLB Virtual Service Health Check

The TCP-based SLB virtual service health check supports external devices to detect the availability of TCP-based virtual services defined in FortiBalancer appliances.

How it works

If all real services corresponding to TCP-based SLB virtual service go down, the status of the virtual service will be marked as "DOWN" and the TCP connection attempting from outside will NOT be responded to so that the other devices will know the unavailability of the detected virtual service.

Note: If WebWall is turned on for the interface ("port2" for most cases) that SLB real services are using, you will need to add an access list rule to allow traffic between the FortiBalancer appliance and the real backend servers. If you also have health check turned on for SLB real services, you will need to add corresponding access list rules to allow the health check replies. For example, if the health check type is ICMP, it needs to add the corresponding access list rules to allow ICMP echo reply messages. Otherwise, the SLB real services will fail.

6.2.4.5 Health Failover

If all real servers configured in an FortiBalancer appliance are marked as DOWN by Health Check, other FortiBalancer appliances will take over the traffic. As long as at least one real server configured in an FortiBalancer appliance is marked as UP by Health Check, the FortiBalancer appliance will take over the traffic again if its mode is preemptive.

6.2.5 Transparent, Reverse Proxy and Triangle Transmission

6.2.5.1 Transparent Transmission

For transparent transmission, the FortiBalancer appliance will direct the connections toward specified real servers directly when forwarding requests from clients. That is, the server is aware of the client's IP address and therefore knows which client accesses the server.

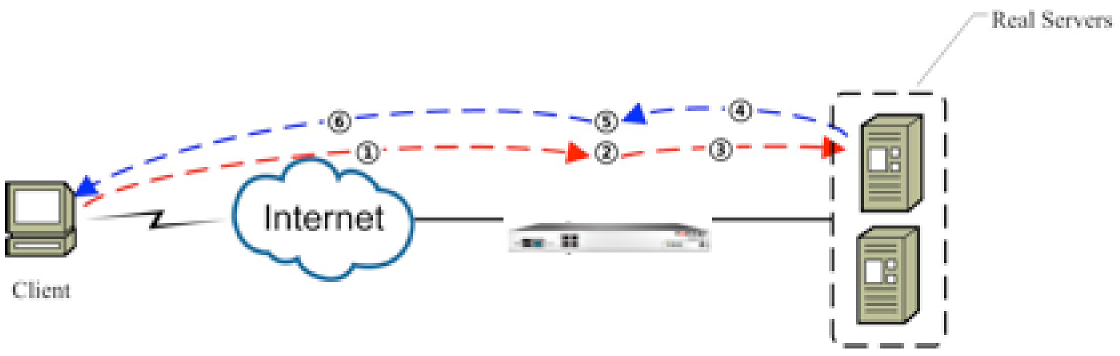


Figure 6-3 Transparent Transmission

1. The client sends a request to a virtual IP on the FortiBalancer appliance.

2. The FortiBalancer appliance switches the destination address of the request into real server IP.
3. The FortiBalancer appliance locates the optimal server available based on the policy and health check conditions, and then forwards the request to the server.
4. After receiving the request, the server sends the response to the FortiBalancer appliance.
5. The FortiBalancer appliance switches the source address into virtual IP;
6. The FortiBalancer appliance sends the response to the client.

➢ Advantages of transparent transmission:

  The server can record the IP addresses of the clients.

➢ Limitations of transparent transmission:

  Structure/router design requires responses from source servers to pass through the FortiBalancer appliance;

  In the one-armed structure, transparent transmission through FortiBalancer appliance is not applicable when the client is at the same network segment with the real server.

  Because requests have different IPs, the system performance cannot be enhanced through the connection pool.

6.2.5.2 Reverse Proxy Transmission

Reverse proxy transmission is used to forward requests to internal servers. The FortiBalancer appliance can distribute the requests evenly among internal servers to ensure server load balancing. As the name of reverse proxy mode indicates, the reverse proxy mode enables clients to access internal servers, whereas the normal proxy transmission enables clients to access external servers. The following figure shows how reverse proxy transmission works:

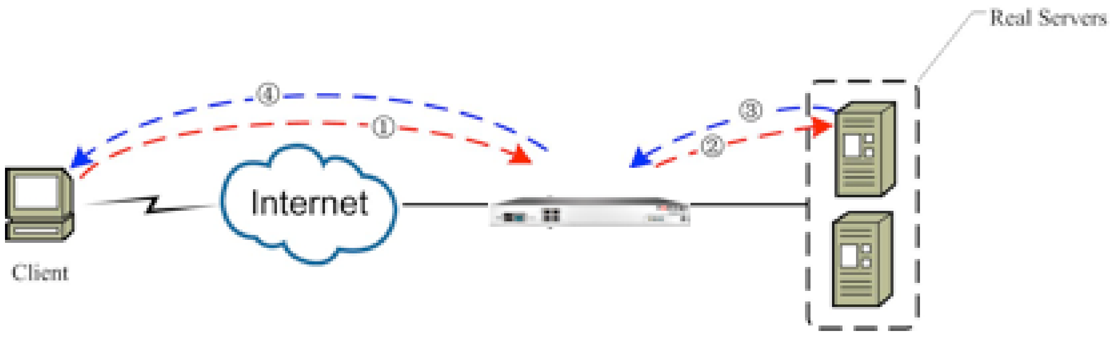


Figure 6-4 Reverse Proxy Transmission

1. The client sends a request to a virtual IP on the FortiBalancer appliance.
2. The FortiBalancer appliance locates the optimal server based on the policy and health check conditions, and then forwards the request to the server.
3. The server sends the response to the FortiBalancer appliance.
4. The FortiBalancer appliance sends the response to the client.

➢ Advantages of reverse proxy transmission:

  One-armed deployment is applicable.

  Connection pool technology can be used to enhance system performance.

➢ Limitations of reverse proxy transmission:

  The IPs of the client that have accessed the server cannot be recorded.

➢ Solution: The FortiBalancer appliance can insert a field “X-Forwarded-For” into the header of the HTTP request of the client to trace the client IP.

6.2.5.3 Triangle Transmission

Triangle Transmission is specially designed for low-inbound/high-outbound applications such as Video On Demand (VOD), and to accommodate requests in the quickest and most efficient manner. In addition to “reverse” and “transparent” modes, a new system mode “triangle” is added for this new feature.

For Triangle Transmisison, when selecting a proper real server from a group, administrators can use Round Robin (rr), Persistent IP (pi), Hash IP (hi), Consistent Hash IP (chi), Least connections (lc) and SNMP (snmp) group methods. In triangle mode, only TCP, UDP and IP virtual services are supported.

The following shows how the Triangle Transmisison works.

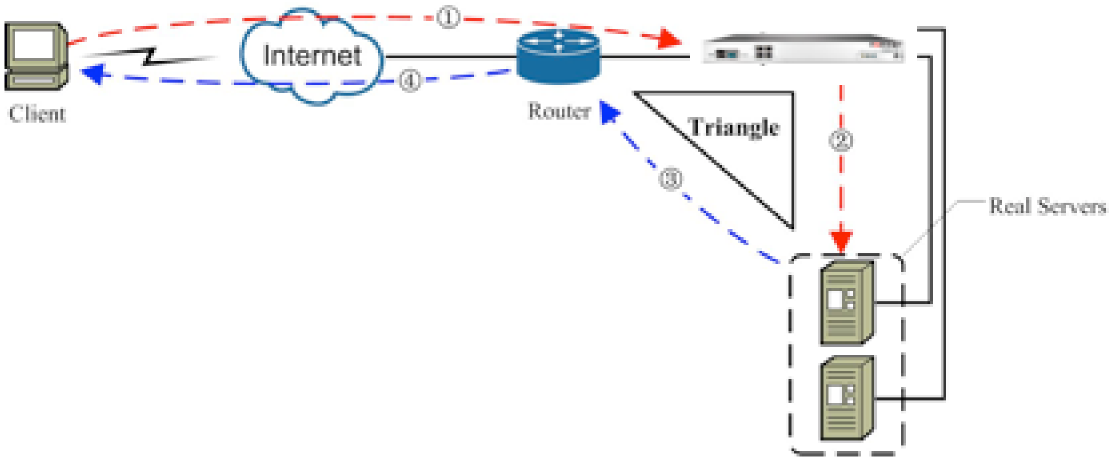


Figure 6-3 Triangle Transmission

1. Client sends a request to a virtual IP on the FortiBalancer appliance via the router.
2. The FortiBalancer appliance forwards the request to a real service.
3. The real service returns a response to the router directly. Since the default route IP on the real service is set to be port2 interface IP of the router, the response will be sent to the router directly.
4. The router forwards the response to the client.

In this packet flow, the request will pass through the FortiBalancer appliance, but the response will be sent from the real server to the client directly without hitting the FortiBalancer appliance.

Note: Triangle transmission SLB health check is based on the system IP addresses of the real servers, not the loopback IP addresses. This means when health check is up, the real service might not be available.

6.2.6 Packet based UDP Load Balancing

In FortiBalancer, UDP packets are distributed according to the four-tuple information. This means the UDP packets with the same source IP and port will be regarded as the same UDP flow and then be distributed to the same real server. However, under some circumstances (e.g. RADIUS applications), this method might cause uneven load balancing. To solve this problem, packet based UDP SLB can be used to support single UDP packet based application and more evenly spread load to different real servers.

6.2.7 SIP Load Balancing

What is SIP (Session Initiation Protocol) Load Balancing

FortiBalancer SIP Load Balancing intelligently distributes and balances SIP traffic among multiple SIP servers and provides application persistence based on the unique SIP caller ID to ensure application and transaction integrity. Additionally, to ensure reliability and availability of SIP services, SIP load balancing is able to perform advanced health checks on SIP devices, routing SIP clients away from unstable or unreliable devices. SIP load balancing is critically important for successful, scalable VoIP telephony environments.

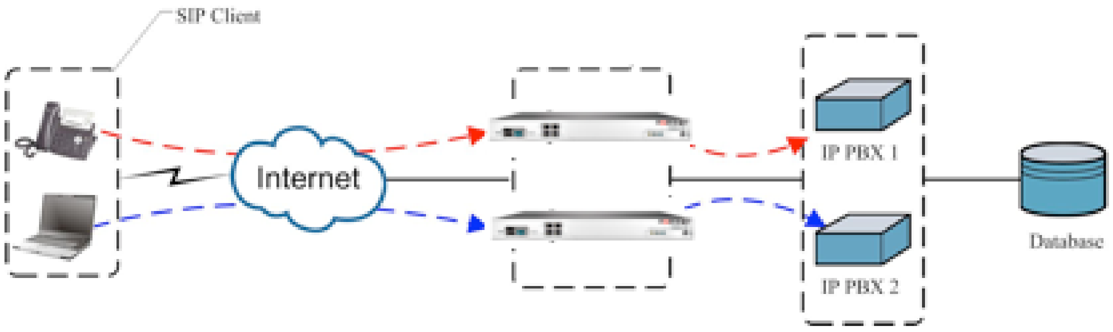


Figure 6-4 SIP Load Balancing

SIP Load Balancing performs stateful inspection of SIP messages to scan and hash calls based on a SIP Call-ID header destined for a SIP server. Stateful inspection means that a packet is inspected not only for its source and destination information found in the header, but also packet contents found at Layer 7 (the Application Layer). Once the FortiBalancer appliance has identified the Call-ID which identifies a specific SIP session, it sends future messages from the same Call-ID to the same SIP server.

A SIP proxy server is implemented to NAT the session packets originated from inside real servers. By SIP NAT, real servers can reside in a private network and do not have to own global IP addresses.

Note: A SIP proxy server is the server controlling the management of connections and IP addresses in a SIP-enabled network.

To synchronize SIP registration information, SIP SLB supports the broadcasting of SIP registration requests to all the SIP register servers in the same SLB group.

6.2.8 RTSP Load Balancing

What is RTSP (Real Time Streaming Protocol)

RTSP (Real Time Streaming Protocol) is an application layer protocol which is used for real-time media traffic. Normally, an RTSP session includes two channels: control channel for control signals and data channel for media stream. The control channel is a TCP connection while the data channel can be either TCP or UDP connection.

What is RTSP Load Balancing

The use of streaming audio and video is growing among enterprises for applications such as eLearning and corporate communications. RTSP streaming delivers higher performance, and is more secure and easier to manage than HTTP streaming implementations. FortiBalancer appliance enables companies to optimize streaming media resources by intelligently and transparently switching requests to RTSP media servers or caches.

RTSP Load Balancing only supports filetype, default, backup and static policy. In filetype policy, the real service is selected based on the file extension names. For example, client request for “rtsp://mp3.xyz.com/test.mp3” will hit the RTSP SLB group corresponding to “mp3” filetype.

Particularly, RTSP Load Balancing supports two working modes: "REDIRECT" and "Dynamic NAT".

➢ **Redirect mode**

The media stream will not pass through FortiBalancer appliance. After FortiBalancer appliance retrieves Request-URL from the client request, it will select a real service according to the file type, and then send a "REDIRECT" response to inform the client to access the selected real service.

➢ **Dynamic NAT mode**

Media stream will pass through FortiBalancer appliance. After FortiBalancer appliance selects a real service according to the policy, it will retrieve the media transport information from the negotiation between the client and the real service. And then implement dynamic NAT for media streaming packets according to their requirement. All connections (TCP and UDP) of one RTSP session will be closed when the control connection tears down.

6.2.9 Layer 2 IP/MAC-based Load Balancing

Layer 2 IP/MAC-based SLB allows you to balance network traffic without changing the network packet's source IP/MAC address and destination IP/MAC address. It is widely used in virus or mail scanner, content filter and triangle transmission.

Different from higher layer SLB (Layer 4 and Layer 7 SLB), the traffic to be balanced in Layer 2 does not need to access a particular IP address or "IP address + Port" pair defined in the interface (normally the port1 interface) on FortiBalancer appliance for balancing purpose. As long as the incoming traffic can be routed to FortiBalancer appliance's interface with defined Layer 2 virtual services (e.g. virtual services are defined on FortiBalancer appliance's port1 interface and FortiBalancer appliance port1 interface's system IP address is set as the gateway of client machines), it will be balanced among a pool of Layer 2 real services according to configured load balance algorithms.

Layer 2 SLB feature has its own definitions for virtual service, real service, group method, health check and policy:

- Virtual service is defined by IP address. Internally, the associated input interface will be found;
- Real service can be defined by either IP or MAC address. Internally, the associated output interfaces will be discovered;
- Layer 2 SLB only supports default policy;
- Layer 2 SLB Health Check: Layer 2 SLB real services support all the health check methods available in Layer 4/Layer 7 SLB health check. Only the additional health check can be configured to the Layer 2 SLB real services.
- The supported groups methods are rr (Round Robin) and hi (Hash IP).

6.2.10 Layer 3 IP-based Load Balancing

Layer 3 IP based SLB balances all the TCP and UDP traffic from one global virtual IP address to multiple real servers. Unlike Layer 4 SLB, both the virtual service and real service in Layer 3 SLB are defined by single IP address without port number. Layer 3 SLB real services only support ICMP health check. Without specifying port number, Layer 3 SLB works for a much wider range of administrators, especially in the following scenarios:

- Cross-port applications: some session protocols bind multiple connections together, one is the initial connection, others could be generated from dynamic ports

- Cross-protocol applications: most streaming protocols use UDP connection for data transferring and TCP connection to transfer control information between the same client and server.

6.2.11 Port Range Load Balancing

Port range SLB allows us to define a virtual service with a range of ports so that theOS will listen for connections on all the ports in the range and distribute the requests to a group of real services that can be configured with either a port range or a static port.

Port range virtual service has lower priority than Layer 4 and Layer 7 virtual services. That means, if an input request hits both a Layer 4/Layer 7 virtual service and a port range virtual service, the Layer 4/Layer 7 virtual service will be matched first.

Port range real service and static port real service can not be configured in one group. Port range real service and port range group can only be associated with port range virtual service. However, port range virtual service can associate with static port real service. Port range SLB does not work for “FTP” type real services and virtual services.

The health check type of a port range real service can only be “none” or “ICMP” because its port is 0. If other health checks are needed, additional health checks can be used:

```
FortiBalancer(config)#slb real http rs1 10.3.0.20 0 1000 none
FortiBalancer(config)#slb real health a1 rs1 10.3.0.20 80 http
```

6.2.12 Terminal Server Load Balancing

FortiBalancer provides session persistency to balance RDP (Remote Desktop Protocol) traffic load among a terminal server farm by using Terminal Services Session Directory Service. This enables a user to disconnect a session with running applications, whether intentional or because of a network failure, and then reconnect at a later time to the same session, with the same running application.

Terminal Services Session Directory Service is a database that keeps track of sessions on terminal servers in a load-balanced farm. The database maintains a list of the user names that are associated with the session IDs that are connected to the servers in a load-balanced terminal server farm. It can either reside on a server that is separate from the terminal servers in the farm, or be hosted on a member of the terminal server farm.

When a client sends an authentication request to FortiBalancer appliance, FortiBalancer appliance will forward the request to one of the terminal servers in the farm, for example TS1. TS1 prompts the client with a login screen. After the client enters the user name and password, TS1 validates the user name and password, and queries Session Directory database with the user name. If a session with the same user name already exists on one of the terminal servers in the farm, for example TS2, Session Directory will send the information (including TS2’s IP address and port number) to TS1, and TS1 will send a routing token with the TS2’s IP address and port to the client and drop the connection with the client. After that, the client sends FortiBalancer appliance another authentication request with the routing token. FortiBalancer appliance will reconnect the client to TS2 according to the IP address and port in the routing token.

6.2.13 RADIUS Server Load Balancing

In RADIUS (Remote Authentication Dial-In User Service) Load Balancing solution, when the RADIUS request packets go through the FortiBalancer appliance before they are forwarded to the backend RADIUS servers, the FortiBalancer appliance will first check the username or session ID field in the RADIUS request packets and employ relevant load balancing methods (“radchu” method for the “username” field and “radchs” method for the “session ID” field) to create a mapping between the RADIUS request packets and the hit RADIUS server. And then, FortiBalancer will check if a connection with the RADIUS servers exists. If the connection exists, the RADIUS packets will be sent to that backend RADIUS server; otherwise, the connection will be created and saved (the connection is time limited and will be removed as soon as it times out) and then the request will be sent to the RADIUS server via the connection. In this way, the FortiBalancer appliance can implement even load balancing among the multiple RADIUS servers.

6.2.14 DirectFWD

DirectFWD is a new Layer 4 SLB function by utilizing a multi-thread and non-lock architecture based on a multi-core system. This new architecture has maximized the advantage of the multi-core system. Compared with the traditional Layer 4 SLB, DirectFWD provides remarkably better Layer 4 SLB performance. This function is controlled by the command "**slb direcfwd {on|off}**".

DirectFWD supports both the IPv4 and IPv6-based TCP packets.

Since only limited functions are implemented in the new architecture now, the DirectFWD function still has some limitations as follows:

1. It is only used for TCP SLB, and temporarily only supports static, default and backup policies. All the Layer 4 SLB methods, except Shortest Response Time, are supported. (Please refer to the policy-method supporting matrix to check the details about the Layer 4 SLB methods).
2. It cannot be used in different MTU environments. In other words, all the interfaces must have the same MTU; otherwise, the connection may fail to handle the big packets.
3. It cannot handle IP fragments.

Note: The more the interfaces used for DirectFWD, the better the performance.

DirectFWD Syncache

DirectFWD syncache effectively and efficiently protects the real servers from SYN flood DOS attacks. When DirectFWD syncache function is on, all the SYN packets from a client will not be forwarded to a real server directly. The FortiBalancer appliance will hold the useful information in those SYN packets firstly, and then send a SYN-ACK packet to the client. After that, the FortiBalancer appliance will receive an ACK packet from the client and setup a TCP connection with the real sever.

Note: When the DirectFWD syncache function is on, clients cannot negotiate MTU with real servers.

6.2.15 Real Service Graceful Shutdown and Warm-up Activation

6.2.15.1 Real Service Graceful Shutdown

By default, when a real service is disabled or deleted, the FortiBalancer appliance SLB shall not send session requests to the real service that has been disabled. However, for the real services using cookie-based group method and load balancing polices, such as pc (Persistent Cookie), ic (Insert Cookie), rc (Rewrite Cookie), SLB will still send the existing session requests that match the cookie to the disabled real service to ensure service persistence. While the new session requests will be sent to other working real services. This function is called "Graceful Shutdown", as shown below.

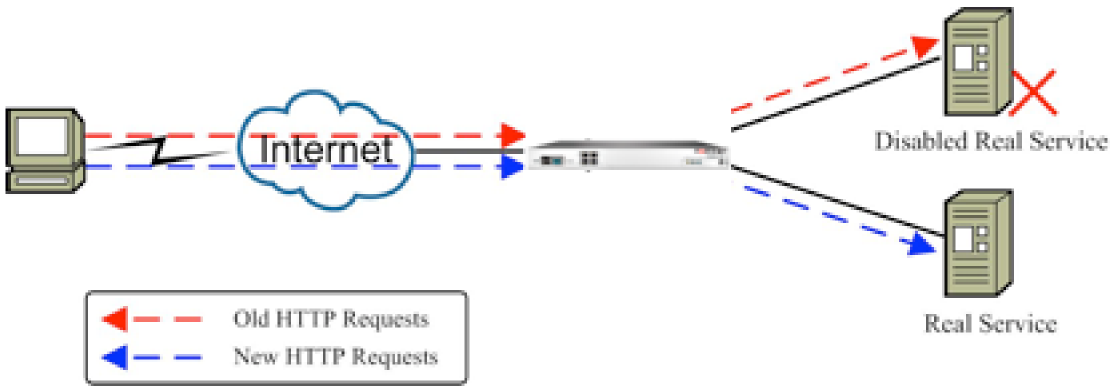


Figure 6-5 Graceful Shutdown of Real Services

The following gives an example of Graceful Shutdown:

```
FortiBalancer(config)#slb real disable service
```

After disabling the real service named "service", users can check the status of the real service by using the command "**show statistics slb real**".

```
FortiBalancer(config)#show statistics slb real http service
Real service service 10.8.6.42 80 DOWN INACTIVE(waiting)
            Main health check: 10.8.6.42 80 tcp DOWN
            Max  Conn Count:                1000
            Current Connection Count:    4572
            Outstanding Request Count: 4215
            Total Hits:                         311
            Total  Bytes In:                 39431
            Total Bytes Out:                 53466
            Total Packets In:               7541
            Total Packets Out:              3252
            Average Response time:         32.000  ms
```

As shown in the above output information, the status of "service" is displayed as "INACTIVE(waiting)", which means the real service is still processing connection requests, i.e. it is in the process of "Gracefully Shutdown". During this process, the session requests that match the cookie will still be forwarded to this real service, while the connection requests from new clients will be forwarded to other working real services.

After a while, users can run the command "**show statistics slb real**" again to check the status of the real service.

```
FortiBalancer(config)#show statistics slb real http service
Real service service 192.168.10.10 80 DOWN INACTIVE(suspend)
            Main health check: 192.168.10.10 80 tcp DOWN
            Max Conn Count:                 1000
            Current Connection Count:    0
            Outstanding Request Count: 0
            Total Hits:                         0
            Total Bytes In:                  0
            Total Bytes Out:                  0
            Total Packets In:               0
            Total Packets Out:              0
            Average Response time:         0.000  ms
```

As shown in the above output information, the status of "service" now is displayed as "INACTIVE(suspend)", which means it is shut down completely.

6.2.15.2 Real Service Warm-up Activation

After a real service is enabled, it might receive a large number of requests immediately before getting fully prepared to work. The sudden increase of traffic on the real service might lead to failure of the service. To solve this problem, the FortiBalancer appliance supports warm-up activation of real services. Administrators can set recovery time and warm-up time for real services.

Right after a real service is enabled, no connection requests will be sent to the real service for a period of time. This period is called “recovery time”. When the recovery time ends, the real service enters the “warm-up time”. In this period of time, firstly only a small amount of connection requests are forwarded to the real service for processing. The number of the requests will be increased gradually. When it reaches the maximum number allowed for the real service, it means the real service works normally.

Administrators can use the command “**show statistics slb real**” to check the status of a real service. As shown in the following output information, the status of the real service in “recovery time” is displayed as “UP(softup)”. No connection requests will be sent to a real service in “softup” status.

```
FortiBalancer(config)#show statistics slb real http service
Real service service 192.168.10.10 80 UP (softup) ACTIVE
        Main health check: 192.168.10.10 80 tcp ACTIVE
        Max  Conn Count:             1000
        Current Connection Count:    0
        Outstanding Request Count: 0
        Total Hits:                  0
        Total Bytes In:              0
        Total Bytes Out:             0
        Total Packets In:            0
        Total Packets Out:           0
        Average Response time:       0.000  ms
```

6.3 SLB Configuration

6.3.1 HTTP/TCP/FTP/UDP/HTTPS/TCPS/DNS Load Balancing

This section covers more than one SLB configuration example. At first, we will give an example with basic SLB configuration. Then more configuration examples are given based on the different policies. Herein we use HTTP protocol as an example. The configurations of other protocols are similar.

6.3.1.1 Configuration Guidelines

Before you start to configure SLB, you need to get familiar with the following SLB network architecture.

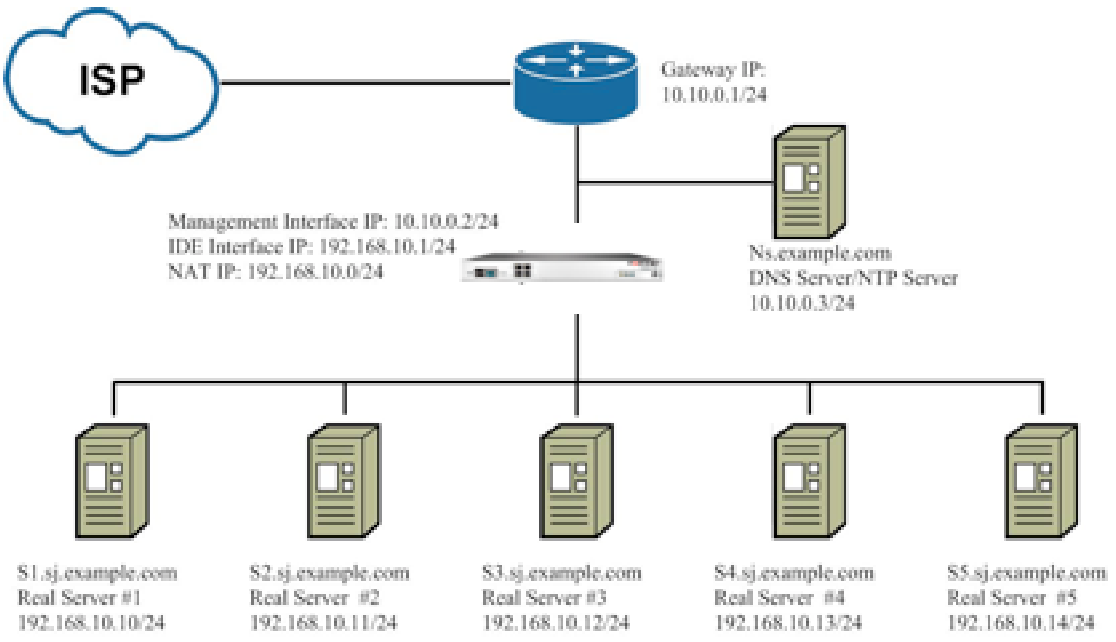


Figure 6-6 SLB Netwok Architetcure

Table 6-4 General Settings of SLB

Operation	Command
Configure real services	**slb real tcp** *<real_name> <ip> <port> [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|ldap] [hc_up] [hc_down]* **slb real ftp** *<real_name> <ip> [port] [max_conn] [tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real http** *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real udp** *<real_name> <ip> <port> [max_conn] [hc_up] [hc_down] [timeout] [icmp|script-tcp|script-udp|radius-auth|radius-acct|sip-tcp|sip-udp|dns]* **slb real https** *<real_name> <ip> [port] [max_conn] [https|tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real tcps** *<real_name> <ip> <port> [max_conn] [tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real dns** *<real_name> <ip> <port> [max_conn] [dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down] [timeout]* **slb real health** *<add_hc_name> <real_name> <ip> <port> [http|https|tcp|icmp|dns|ldap|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|rtsp-tcp] [hc_up] [hc_down]*
Define group methods	**slb group method** *<group_name>* **[rr|pu|sr]** **slb group method** *<group_name>* **hc** *[rr|sr|lc] [weight|threshold]* **slb group method** *<group_name>* **ic** *[cookie_name] [add_path] [rr|sr|lc] [threshold]* **slb group method** *<group_name>* **rc** *[cookie_name] [offset] [rr|sr|lc] [threshold]* **slb group method** *<group_name>* **lc** *[threshold] [yes|no]* **slb group method** *<group_name>* **hh** *<header_name> [rr|sr|lc] [threshold] [prefix] [delimiter]* **slb group method** *<group_name>* **prox** *[rr|sr|lc] [threshold]*

Operation	Command
	slb group method *<group_name>* **ec** *<cookie_name> [rr|sr|lc] [threshold]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb virtual {tcp|tcps|udp}** *<virtual_name> <vip> <vport> [arp|noarp] [max_conn]* **slb virtual rtsp** *<virtual_name> <vip> [vport] [mode] [arp|noarp] [max_conn]* **slb virtual {ftp|ftps}** *<virtual_name> <vip> [vport] [max_conn]* **slb virtual ip** *<virtual_name> <vip>* **slb virtual l2ip** *<virtual_name> <vip> [gateway_ip]*
Bind the group (or a real service) to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_name> <real_name>* **slb policy qos url** *<policy_name> {virtual_name|vlink_name} {group_name|vlink_name} <qos_string> <precedence>* **slb policy qos cookie** *<policy_name> {virtual_name|vlink_name} {group_name|vlink_name} <cookie_name=cookie_value> <precedence>* **slb policy persistent cookie** *<policy_name> {virtual_name|vlink_name} <group_name> <cookie_name> <precedence>* **slb policy qos hostname** *<policy_name> {virtual_name|vlink_name} {group_name|vlink_name} <host_name> <precedence>* **slb policy redirect** *<policy_name> <virtual_name> <group_name> <redirected_from_host>*

6.3.1.2 Configuration Example via CLI

➢ Step 1 Define real services

The first step in setting up your network architecture with the FortiBalancer appliance performing SLB tasks is to create and configure your real services. To define real services for our model, use the following command:

For setting up the first real service:

```
FortiBalancer(config)#slb real http service1http 192.168.10.10
```

When you define an HTTP real service with just the real service name and the IP address, it will use the following default values:

- Port: 80
- Max Con: 1000
- Health Check: tcp
- Consecutive up health check results before server is marked up: 1
- Consecutive down health check results before server is marked down: 1

For service2 we will define what kind of health check to use, HTTP in this case.

```
FortiBalancer(config)#slb real http service2http 192.168.10.11 80 1000 http 3 3
```

For other services, we will rely on the defaults for now.

```
FortiBalancer(config)#slb real http service3http 192.168.10.12
FortiBalancer(config)#slb real http service4http 192.168.10.13
FortiBalancer(config)#slb real http service5http 192.168.10.15
```

Now all five real services are defined for the FortiBalancer appliance. To verify the configuration, you may use the "**show slb real http**" command. You may either specify a service by supplying the real name, or leave this field blank and view all of your configured real services.

Additional Health Check for Real Services

For setting additional health check for the first real service:

```
FortiBalancer(config)#slb real health a1service1http 192.168.10.10 80 http
FortiBalancer(config)#slb real health a1service1http 192.168.10.10 8080 http
```

So the real server "service1http" will be healthy only when the main health check (192.168.10.10, 80, tcp), and the two additional health check servers (192.168.10.10 80 http and 192.168.10.10 8080 http), are all reported as healthy.

Maintaining the Real Services

Sometimes it becomes necessary to disable a real service for maintenance or for temporary shifts in resource allocation. To disable a real service, use the "**slb real disable**" command:

```
FortiBalancer(config)#slb real disable service1http
```

Now verify our configuration change:

```
FortiBalancer(config)#show slb real all
```

To re-enable the real server just use the "**slb enable**" command.

```
FortiBalancer(config)#slb real enable service1http
```

By default, when a real service is disabled or deleted, the FortiBalancer appliance SLB shall not send session requests to the real services that have been disabled. However, for cookie-based group method and load balancing polices: PC (Persistent Cookie), IC (Insert Cookie), RC (Rewrite Cookie), SLB supports the "graceful shutdown" of real services.

Graceful Shutdown

If a real service is disabled when it is already in a cookie-based group, SLB will still send the existing session requests which match the cookie to the disabled real service. The new requests will be sent to other working real services.

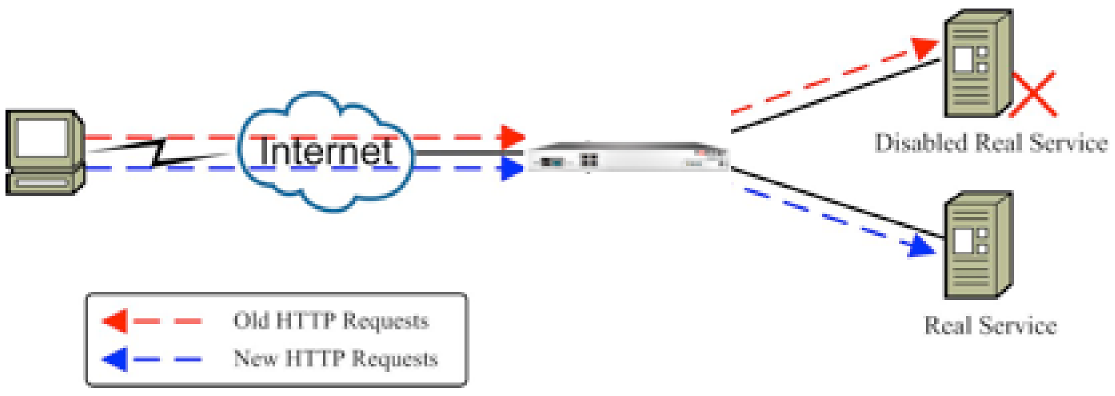


Figure 6-7 Graceful Shutdown of Real Services

For example, you can configure the graceful shutdown function as follows:

```
FortiBalancer(config)#slb real http r1 10.3.0.20 80
FortiBalancer(config)#slb group method g1 pc
FortiBalancer(config)#slb group member g1 r1 "server1"
FortiBalancer(config)#slb real disable r1
```

After the above configurations, the existing session requests will still be sent to r1.

- Step 2 Define groups

Now that the real services have been defined, it is time to assign them to groups. A group is first defined by using the "**slb group method**" command. Below is an example from our model.

Once the group method is defined, the method changes from rr, sr and lc methods switching to rr, sr, lc, pi, hi, and chi methods by employing the "**slb group method**" command can work without removing the configured method. For other method changes, the old group method has to be removed firstly, and then a new one can be added.

So for our purposes here we will employ the following command:

```
FortiBalancer(config)#slb group method rrgroup rr
```

We have just defined a group with the name of "rrgroup" and a metric of Round Robin.

- Step 3 Add the real services into the defined group

Once we have defined a group we simply add the real service by using the "**slb group member**" command:

```
FortiBalancer(config)#slb group member rrgroup server1http
FortiBalancer(config)#slb group member rrgroup server2http
FortiBalancer(config)#slb group member rrgroup server3http
FortiBalancer(config)#slb group member rrgroup server4http
FortiBalancer(config)#slb group member rrgroup server5http
```

- Step 4 Define virtual services

A virtual service is an access point that will service requests for the content which a group is designed for. For Layer 4 and Layer 7 SLB, a virtual service is just a (VIP, port) pair. A VIP (virtual IP) is mostly a public IP address which can be accessed from external clients. For example, if group1 is a set of image servers, then we could define a VIP of 10.10.0.10 that is tied to group1. Any requests made to this Virtual IP will be passed to either the Cache or SLB subsystem depending on your cache and SLB settings. In essence you are hiding your internal architecture by only exposing one IP and not many. Sometimes, the word "VIP" in this chapter is used for "virtual service".

Notes:

1. The VIP address cannot be the same IP as any management IP address.

2. The VIP address configured must be within the same subnet with any system interface on the appliance (except the 0.0.0.0 and noarp cases). For the VIP address that does not match the subnet of any interface, the FortiBalancer appliance will not allow it to be configured.

3. If a VIP address is not tied to a policy the client will get a 503 Service Unavailable response from the FortiBalancer appliance. 503 will also be returned when all real servers are down in a group.

To establish virtual services:

```
FortiBalancer(config)#slb virtual http virtual1http 10.10.0.10 80
```

We now have the IP address 10.10.0.10, listening for requests on port 80. Next we must define a policy so incoming requests will be directed to the proper group.

- Step 5 Define policies

The final step is to define a **default policy** to bind a virtual service to a Layer 4 "group".

```
FortiBalancer(config)#slb policy default virtual1http rrgroup
```

This command sets the default policy for a request on virtual1http to be serviced from rrgroup. We now have Layer 4 load balancing using the round robin metric, live on the FortiBalancer appliance. You should be able to test the configuration by typing the HTTP URL: “http://10.10.0.10/” in a Web browser.

6.3.1.3 Policies-based SLB Configuration Example

QoS URL

Using QoS URL, we can setup policies that will look into the URL string and make a decision based upon the information housed within that string. For our next example, we will setup two groups. Group 1 will service all requests with ‘.jpg’ in the URL while Group 2 will service all requests with ‘english’ in the URL.

Group 1: URL: .jpg

Members: S1.sj.example.com

S2.sj.example.com

Group 2: URL: ‘english’

Members: S3.sj.example.com

S4.sj.example.com

We do not have to redefine the real servers and virtual service so we will leave them as they were originally defined in the basic SLB configuration example above.

- Step 1 Define the two groups and their members, much as before

```
FortiBalancer(config)#slb group method group1 rr
FortiBalancer(config)#slb group method group2 rr
```

- Step 2 Add the real services into the groups (the real services are defined in the last example)

```
FortiBalancer(config)#slb group member group1 service1http
FortiBalancer(config)#slb group member group1 service2http
FortiBalancer(config)#slb group member group2 service3http
FortiBalancer(config)#slb group member group2 service4http
```

- Step 3 Set the policies and the URL associated with each group (the virtual service is defined in the last example)

```
FortiBalancer(config)#slb policy qos url url_pol_1 virtual1http group1 “.jpg ”1
FortiBalancer(config)#slb policy qos url url_pol_2 virtual1http group2 english 2
```

- Step 4 Define a default policy

If we receive a request that has neither .jpg or ‘english’ in the URL, then the FortiBalancer appliance will returne a 503 Service Unavailable since it does not know how to handle the request. This is fine if it is guaranteed every URL will always contain one of the strings. If not, it is best to define the default policy. In our case we are just going to direct any request that does not contain one of the strings to go to the “english” servers belonging to group 2.

```
FortiBalancer(config)#slb policy default virtual1http group2
```

QoS Cookie

QoS cookie allows you to add policies to the Layer 7 rules to load balance requests to a group based on a cookie name and value pair. The following figure shows the QoS cookie in action. We are going to define our two groups, define our cookies, and then setup the policies to make cookie-based load balancing work within our model network.

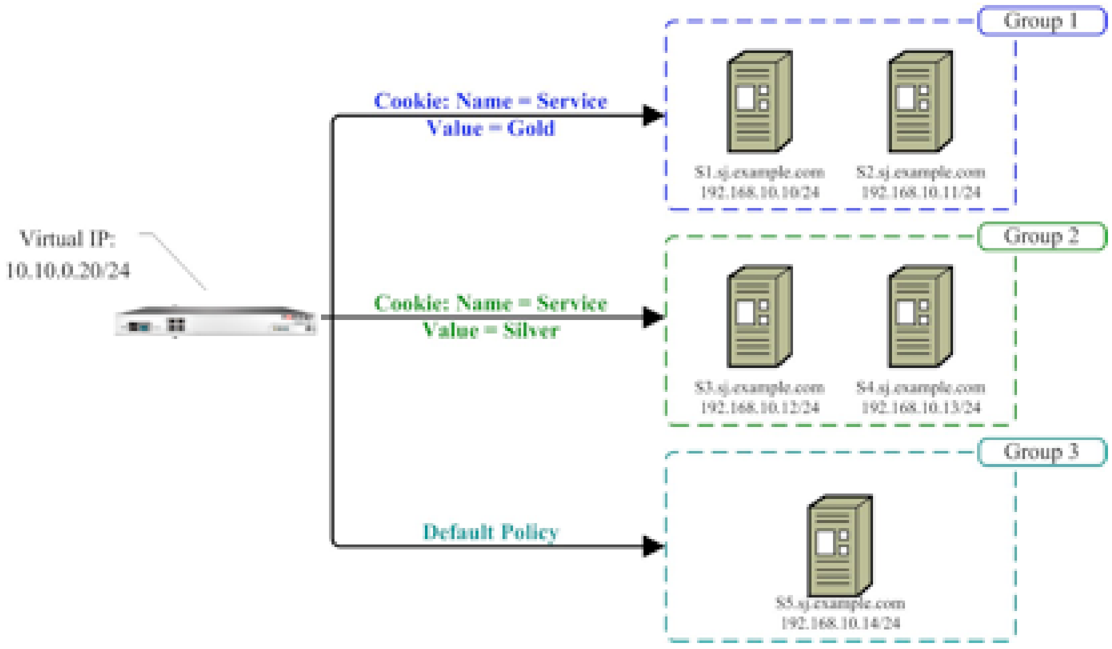


Figure 6-8 QoS Cookie Policy

Since we are using QoS cookie, which works on top of the existing groups, we define the groups as would with Layer 4 rules.

Group1: Round Robin

Members:	S1.sj.example.com
	S2.sj.example.com

Group2: Round Robin

Members:	S3.sj.example.com
	S4.sj.example.com

Group3: Round Robin

Members:	S5.sj.example.com

The path that a packet goes through in this policy is as follows:

1. If there is a Cookie sent with the request that has the name “Service” and value “Gold”, then go to Group 1.
2. If there is a Cookie sent with the request that has a name “Service” and value “Silver”, then go to Group 2.
3. If there are no cookies in the request, go to Group 3.

Group3 will be our cookie setters. If a client has never been to a site then the request will not contain a cookie since it is the server that sets the cookie for the first time. These types of requests will not match Group1 or Group2 and will be served through Group3, the default. After the client has been to the Web site through Group3 and the cookie has been set by the server, the next time the client accesses the Web site the browser will send the cookies in the HTTP request headers. These cookies will ensure we pick the appropriate service from Group1 or Group2.

Note: If we do not have the default policy, the rule will “drop through” and the FortiBalancer appliance will return a 503 service unavailable.

Now that we have defined what needs to happen, let’s configure the appliance. We do not have to redefine the real services so we will leave them as they have been originally defined in the basic SLB configuration example.

➢ Step1 Define three groups

```
FortiBalancer(config)#slb group method gold_group rr
FortiBalancer(config)#slb group method silver_group rr
FortiBalancer(config)#slb group method cookie_set_group rr
```

➢ Step2 Add the real services into the groups

```
FortiBalancer(config)#slb group member gold_group service1http
FortiBalancer(config)#slb group member gold_group service2http

FortiBalancer(config)#slb group member silver_group service3http
FortiBalancer(config)#slb group member silver_group service4http

FortiBalancer(config)#slb group member cookie_set_group service5http
```

➢ Step 3 Set the default policy for group “*cookie_set_group*”, where cookies will be set

```
FortiBalancer(config)#slb policy default virtual1http cookie_set_group
```

➢ Step 4 Set the QoS Cookie policy for group “*gold_group*” and “*silver_group*”

```
FortiBalancer(config)#slb policy qos cookie gold_policy virtual1http gold_group
“service=gold” 1
FortiBalancer(config)#slb policy qos cookie silver_policy virtual1http silver_group
“service=silver” 2
```

Persistent Cookie

Using persistent cookies, you can set cookie names and values and tie them to specific real servers. This is different from QoS cookie in which the requests go directly to a server. And it so happens that the configuration of persistent cookie is completely different.

Group 1

Server1: Cookie Name: Service

Cookie Value: GOLD

Server2: Cookie Name: Service

Cookie Value: SILVER

Group 2

Server3: Default cookie setter

Definition of the real services has been already completed. Now let’s proceed to setting up the cookies, groups and then the policies.

➢ Step 1 Define two groups

```
FortiBalancer(config)#slb group method group1 pc
FortiBalancer(config)#slb group method group2 rr
```

Note: Persistent Cookie must be used with either Hash Cookie or Persistent Cookie group balancing methods.

➢ Step 2 Add the real services into the groups

```
FortiBalancer(config)#slb group member group1 service1http gold
FortiBalancer(config)#slb group member group1 service2http silver
FortiBalancer(config)#slb group member group2 service3http
```

- Step 3 Set the default policy for “group2”, where cookies will be set

```
FortiBalancer(config)#slb policy default virtual1http group2
```

- Step 4 Set the persistent cookie policy for “group1”

```
FortiBalancer(config)#slb policy persistent cookie perst_pol virtual1http group1 Service 1
```

Note: All members of a PC group must have a cookie name association.

QoS Hostname

QoS hostname based load balancing makes decisions on the host name within the URL. This is also known as Virtual Hosting. This setup is similar to QoS cookie, but we are going to use **hostname** policies instead of **qos cookie** policies.

In the following figure we have three groups. In our example, c-one.example.com refers to “customer one” while c-two.example.com refers to “customer two”. Any other host name that is used to access the VIP will go to the default policy.

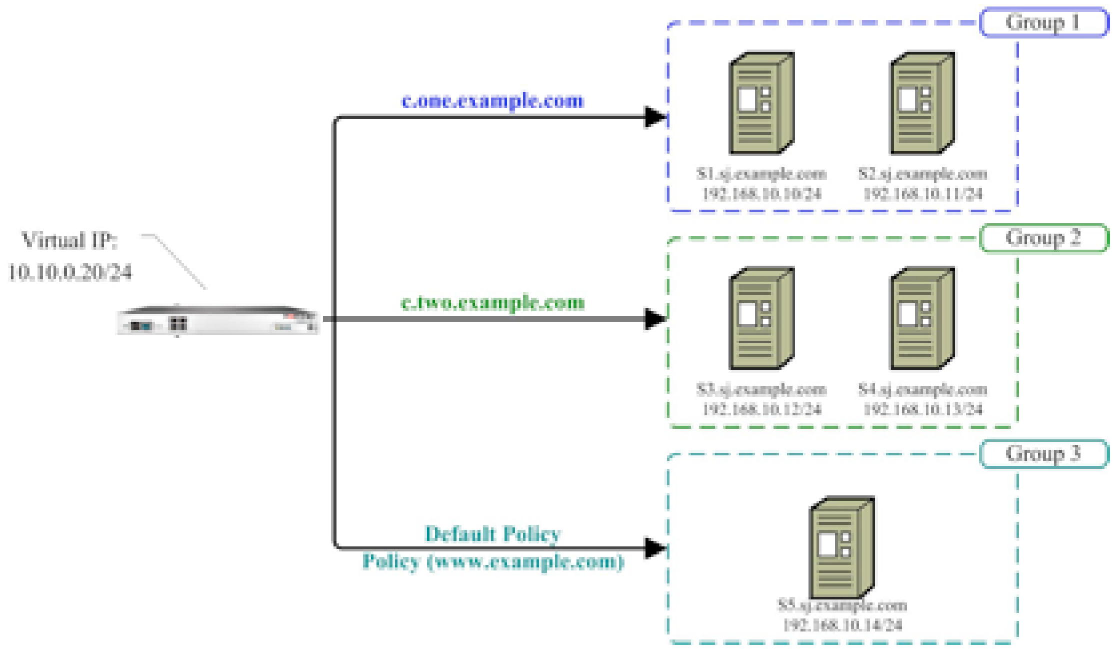


Figure 6-9 QoS Hostname Policy

Group 1: host name: c-one.example.com (Round Robin)

Members:	S1.sj.example.com
	S2.sj.example.com

Group 2: host name: c-two.example.com (Round Robin)

Members:	S3.sj.example.com
	S4.sj.example.com

Group 3: any other host name (Default Policy)

Members:	S5.sj.example.com

➢ Step 1 Define the groups

```
FortiBalancer(config)#slb group method c_one_group rr
FortiBalancer(config)#slb group method c_two_group rr
FortiBalancer(config)#slb group method www_group rr
```

Note: QoS hostname policy can be used with any balancing method except Persistent Cookie and Persistent URL.

➢ Step 2 Add the real services into the groups

```
FortiBalancer(config)#slb group member c_one_group service1http
FortiBalancer(config)#slb group member c_one_group service2http
FortiBalancer(config)#slb group member c_two_group service3http
FortiBalancer(config)#slb group member c_two_group service4http
FortiBalancer(config)#slb group member www_group service5http
```

➢ Step 3 Define the policy for "www_group"

By default, we want "www_group" to service any requests that do not have "c-one.example.com" or "c-two.example.com" as their host names.

```
FortiBalancer(config)#slb policy default virtual1http www_group
```

➢ Step 4 Define the QoS Hostname policy for "c_one_group" and "c_two_group"

```
FortiBalancer(config)#slb policy QoS hostname c_one_pol virtual1http c_one_group
c_one.example.com 1
FortiBalancer(config)#slb policy QoS hostname c_two_pol virtual1http c_two_group
c_two.example.com 2
```

Persistent URL

Persistent URL works by looking for a string which has the format: tag=value. This is typically used within a URL when the browser is submitting a form to the server. This way you can set persistence to the backend server by the value of a parameter being passed to the CGI script. For example a URL of the form:

```
http://www.example.com/find_user.pl?username=bob
```

This will match our Layer 7 policy and direct the request to server1. The following configuration example shows you how to set up a persistent URL policy.

➢ Step 1 Create the group for Persistent URL policy

```
FortiBalancer(config)#slb group method pu_group pu
FortiBalancer(config)#slb group method regular_group rr
```

Note: Persistent URL policy must be used with a persistent URL (pu) and hash query (hq) group.

➢ Step 2 Add the real services into the group

```
FortiBalancer(config)#slb group member pu_group service1http bob
FortiBalancer(config)#slb group member pu_group service2http janet
FortiBalancer(config)#slb group member pu_group service3http steve

FortiBalancer(config)#slb group member regular_group service4http
FortiBalancer(config)#slb group member regular_group service5http
```

➢ Step 3 Setup the policies

```
FortiBalancer(config)#slb policy default virtual1http regular_group
FortiBalancer(config)#slb policy persistent url pol1 virtual1http pu_group username 1
```

Insert Cookie

Insert Cookie (ic) is a method that allows users to insert a cookie into the server response in order to maintain the persistency between the clients and servers.

- Step 1 Create an SLB group and configure Insert Cookie algorithm for it

```
FortiBalancer(config)#slb group method ic_group ic
```

Note: Insert Cookie policy must be used with an Insert Cookie (ic) group.

- Step 2 Define the Insert Cookie properties for the group

```
FortiBalancer(config)#slb group option ic ic_group “expires=300 secure=yes”
```

- Step 3 Add real services into the Insert Cookie group

```
FortiBalancer(config)#slb group member ic_group service1http
FortiBalancer(config)#slb group member ic_group service2http
FortiBalancer(config)#slb group member ic_group service3http
FortiBalancer(config)#slb group member ic_group service4http
FortiBalancer(config)#slb group member ic_group service5http
```

- Step 4 Set up the SLB policies to associate the group with virtual services

```
FortiBalancer(config)#slb policy icookie pol1 virtual1http ic_group 1
FortiBalancer(config)#slb policy default virtual1http ic_group
```

By default you do not need to define a cookie name. The system will generate a random cookie name and save it in the configuration file. You can view the cookie name by running the “**show slb group method**” command, as follows:

```
FortiBalancer(config)#show slb group method
slb group method ic_group “yqv” 1 rr
```

If you want to set the name of the cookie, just add the name by using the command “**slb group method** *<group_name>* **ic** *[cookie_name] [add_path] [rr|sr|lc] [threshold]*”. For example:

```
FortiBalancer(config)#slb group method ic_group ic CookieExampleName
```

Now the cookie “CookieExampleName” will be inserted into the responses from the real services. The value will be used by the FortiBalancer appliance to determine which real service to keep persistent to. For example:

CookieExampleName=server1http

Rewrite Cookie

Rewrite cookie allows us to rewrite a section of a cookie value to make sure that the cookie gets sent back to the same server. We will not strip out the modifications that the OS has made. So the backend server will see the modified cookie.

Note: For insert cookie, we will not strip out the modifications that the OS has made too. So the backend server will see the inserted cookie now.

- Step 1 Create the group for Rewrite Cookie policy

```
FortiBalancer(config)#slb group method rc_group rc CookieExampleName 4
```

Note: Rewrite Cookie policy must be used with Rewrite Cookie group method and Embed Cookie group method.

- Step 2 Add the real services into the group

```
FortiBalancer(config)#slb group member rc_group service1http
FortiBalancer(config)#slb group member rc_group service2http
FortiBalancer(config)#slb group member rc_group service3http
FortiBalancer(config)#slb group member rc_group service4http
FortiBalancer(config)#slb group member rc_group service5http
```

- Step 3 Setup the policies

Then we use the rcookie policy to bind the virtual service to the group:

```
FortiBalancer(config)#slb policy rcookie pol1 virtual1http rc_group 1
FortiBalancer(config)#slb policy default virtual1http rc_group
```

Now the cookie, CookieExampleName, will be rewritten with the service name. For example the name will look like the following if the response comes from service1http:

CookieExaampleName=valueabcdefghijk

New cookie will be:

```
CookieExampleName= service1http!?ijk
```

Note: The length of cookie value has to be not less than the length of real service name +2. Otherwise the rewrite operation will not be performed.

Redirect

A Redirect policy is applied to the URL host in incoming HTTP requests. Redirect policy allows users to redirect the client's HTTP request from one host to another host. By doing this, all the HTTP requests are balanced to different real services and the persistency is kept by the new host name at the same time.

In our example, the requests for the homepage "http://www.abc.com/index.htm" from the client will be redirected to be "http://www1.abc.com/index.htm", "http://www2.abc.com/index.htm" or "http://www3.abc.com/index.htm" depending on the configured group method (rr).

- Step 1 Define the real services that HTTP requests will be redirected to

```
FortiBalancer(config)#slb real http www1.abc.com    192.168.10.10
FortiBalancer(config)#slb real http www2.abc.com    192.168.10.11
FortiBalancer(config)#slb real http www3.abc.com    192.168.10.12
```

- Step 2 Create a group

```
FortiBalancer(config)#slb group method group1 rr
```

- Step 3 Add the real services into the group

```
FortiBalancer(config)#slb group member group1 www1.abc.com
FortiBalancer(config)#slb group member group1 www2.abc.com
FortiBalancer(config)#slb group member group1 www3.abc.com
```

- Step 4 Use the "**slb policy redirect**" command to associate an existing virtual service with the group

```
FortiBalancer(config)#slb policy redirect pol1 virtual1http group1 www.abc.com
```

Currently, the Redirect policy supports rr (Round Robin), lc (Least Connection), sr (Shortest Response Time) and prox (Proximity) load balancing methods.

6.3.2 SIP Load Balancing

This section gives a configuration example on basic SIP load balancing.

6.3.2.1 Configuration Guidelines

In this section, the example is a one-arm case. The default Gateway of two servers is FortiBalancer appliance (i.e. 172.16.30.170). The server subnet (VLAN 30) and client subnet (VLAN 10) are connected by router 172.16.30.1.

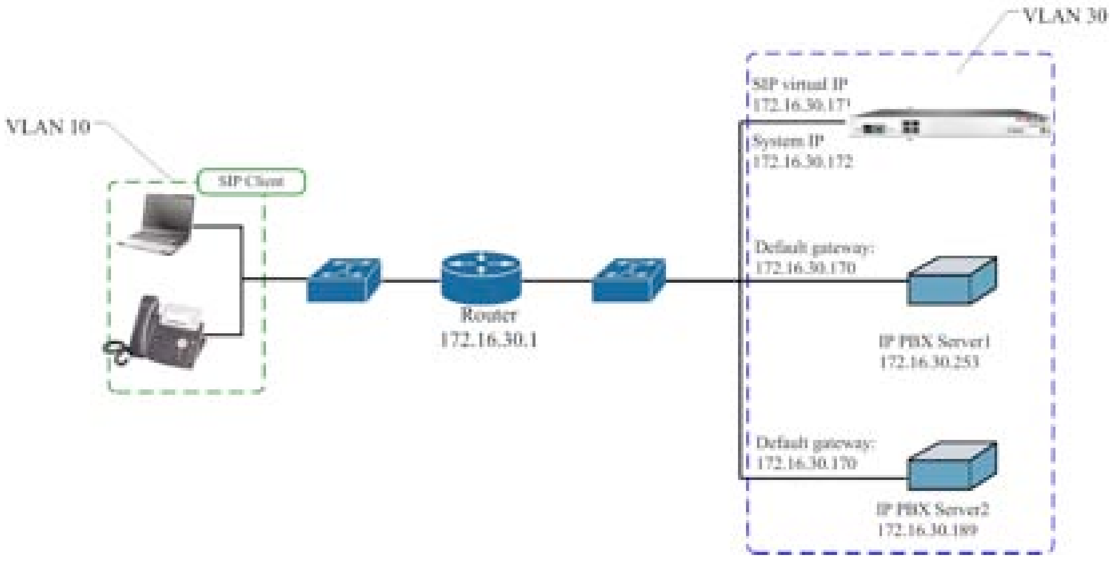


Figure 6-10 SIP Load Balancing

Table 6-5 General Settings of SIP Load Balancing

Operation	Command
Configure real services	**slb real siptcp** *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real sipudp** *<real_name> <ip> [port] [max_conn] [icmp|script-tcp|script-udp|radius-auth|radius-acct|sip-tcp|sip-udp|dns|none] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name>* **{sipcid|sipuid}** *[rr|sr|lc] [threshold]* **slb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** *[threshold] [{yes|no}]* **slb group method** *<group_name>* **pi** *[hash_bits] [rr|sr|lc] [threshold]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **prox** *[rr|sr|lc] [threshold]* **slb group method** *<group_name>* **snmp** *[weight|cpu] [community] [oidcount] [oid1] [oidweight1] [oid2] [oidweight2] [check_interval]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb virtual {tcp|tcps|udp}** *<virtual_name> <vip> <vport> [arp|noarp] [max_conn]* **slb virtual rtsp** *<virtual_name> <vip> [vport] [mode] [arp|noarp] [max_conn]*

Operation	Command
	slb virtual {ftp|ftps} *<virtual_name> <vip> [vport] [max_conn]*

Bind the group to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}*
	slb policy static *<virtual_name> <real_name>*
	slb policy backup *{virtual_name|vlink_name} {group_name|vlink_name}*

6.3.2.2 Configuration Example via CLI

➢ Step 1 Define SIPUDP real services

```
FortiBalancer(config)#slb real sipudp "r1" 172.16.32.253 5060 1000 sip-udp 3 3 60
FortiBalancer(config)#slb real sipudp "r2" 172.16.32.189 5060 1000 sip-udp 3 3 60
```

➢ Step 2 Create a group for SIP load balancing by using the “**slb group method**” command

```
FortiBalancer(config)#slb group method "g1" sipuid rr
```

➢ Step 3 Add SIPUDP real services into the group

```
FortiBalancer(config)#slb group member "g1" "r1" 1
FortiBalancer(config)#slb group member "g1" "r2" 1
```

➢ Step 4 Create virtual services

Then you can define the SIPUDP virtual services by using the “**slb virtual siptcp**” or “**slb virtual sipudp**” command.

```
FortiBalancer(config)#slb virtual sipudp "v1" 172.16.30.171 5060
```

➢ Step 5 Associate the group to the virtual service for SIP SLB

```
FortiBalancer(config)#slb policy default "v1" "g1"
```

➢ Step 6 Configure SIP Multi-register

If the backend servers do not share database, turn on the multi-register function.

```
FortiBalancer(config)#sip multireg on
```

➢ Step 7 Configure SIP NAT

To handle network traffic originated from real servers, you need to set the SIP NAT rules for the defined SIP real services.

```
FortiBalancer(config)#sip nat 172.16.30.171 5060 172.16.32.253 5060 udp 60 callid
FortiBalancer(config)#sip nat 172.16.30.171 5060 172.16.32.189 5060 udp 60 callid
```

6.3.3 RTSP Load Balancing

6.3.3.1 Configuration in Redirect mode

Configuration Guidelines

In our example, the client sends a request “rtsp://10.5.1.80/test.mp3” to virtual service “vs_rtsp1” (10.5.1.80). FortiBalancer appliance chooses a real service according to some policy and method. In redirect mode, FortiBalancer appliance responds the client with the chosen real server’s URL “rtsp://audio2.example.com:554/test.mp3”. The FortiBalancer appliance and the client get disconnected, and the client begins to communicate with the real server “audio2.example.com:554” In this mode, all the real servers should have public IP addresses which can be accessible from Internet clients.

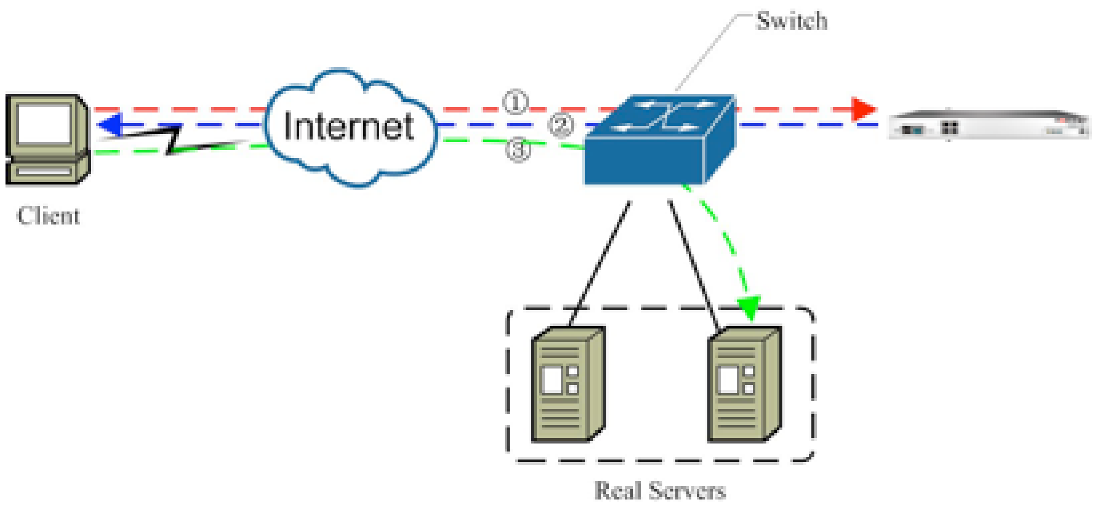


Figure 6-11 RTSP Load Balancing - Redirect Mode

Table 6-6 General Settings of RTSP Load Balancing

Operation	Command
Configure real services	**slb real rtsp** *<real_name> <ip> [port] [max_conn] [rtsp-tcp|tcp|icmp|script-tcp|script-udp|dns] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** *[threshold] [yes|no]* **slb group method** *<group_name>* **pi** *[hash_bits] [rr|sr|lc] [threshold]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **prox** *[rr|sr|lc] [threshold]* **slb group method** *<group_name>* **snmp** *[weight|cpu] [community] [oidcount] [oid1] [oidweight1] [oid2] [oidweight2] [check_interval]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb virtual {tcp|tcps|udp}** *<virtual_name> <vip> <vport> [arp|noarp] [max_conn]* **slb virtual rtsp** *<virtual_name> <vip> [vport] [mode] [arp|noarp] [max_conn]* **slb virtual {ftp|ftps}** *<virtual_name> <vip> [vport] [max_conn]*

Bind the group to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}*
	slb policy static *<virtual_name> <real_name>*
	slb policy backup *{virtual_name|vlink_name} {group_name|vlink_name}*
	slb policy filetype *<policy_name> <vs_name> <group> <filetype>*

Configuration Example for Redirect Mode via CLI

➢ Step 1 Define RTSP real services by using the command "**slb real rtsp**"

When the virtual service mode is "Redirect", the real service name should be "real IP[:port]" or "domainname[:port]".

```
FortiBalancer(config)#slb real rtsp "10.5.1.90" 10.5.1.90 554 1000 rtsp-tcp 3 3
FortiBalancer(config)#slb real rtsp "10.5.1.91:554" 10.5.1.91 554
FortiBalancer(config)#slb real rtsp "audio1.example.com" 10.5.1.92 554
FortiBalancer(config)#slb real rtsp "audio2.example.com:554" 10.5.1.93 554
```

➢ Step 2 Define RTSP real service groups

We can use rr (Round Robin), pi (Persistent IP), hi (Hash IP), chi (Consistent Hash IP), snmp method to choose RTSP real service in one group.

```
FortiBalancer(config)#slb group method "mp3_group" rr
FortiBalancer(config)#slb group member "mp3_group" "10.5.1.90"
FortiBalancer(config)#slb group member "mp3_group" "10.5.1.91:554"

FortiBalancer(config)#slb group method "song" rr
FortiBalancer(config)#slb group member "song" "audio1.example.com"
FortiBalancer(config)#slb group member "song" "audio2.example.com:554"
```

➢ Step 3 Add the real services into the groups

```
FortiBalancer(config)#slb group member "mp3_group" "10.5.1.90"
FortiBalancer(config)#slb group member "mp3_group" "10.5.1.91:554"

FortiBalancer(config)#slb group member "song" "audio1.example.com"
FortiBalancer(config)#slb group member "song" "audio2.example.com:554"
```

➢ Step 4 Define an RTSP virtual service

The default mode of the RTSP virtual service is “Redirect”, and the port is 554.

```
FortiBalancer(config)#slb virtual rtsp "vs_rtsp1" 10.5.1.80
FortiBalancer(config)#slb virtual rtsp "vs_rtsp2" 10.5.1.81 554 "redirect"
```

➢ Step 5 Define a filetype policy to choose a group by file extension

If you add default policy, you will choose that group when you can not find available real services by filetype policy.

```
FortiBalancer(config)#slb policy filetype "p1" "vs_rtsp1" "mp3_group" "mp3"
FortiBalancer(config)#slb policy default "vs_rtsp1" "song"
```

6.3.3.2 Configuration in Dynamic NAT Mode

Configuration Guidelines

In NAT mode, all the RTSP control messages will be balanced to multiple backend media servers across the FortiBalancer appliance. Packets originated from backend media servers (normally the media data) will be NATTed to outside clients. Different from redirect mode, the real servers do not have to use public IP addresses. The internal private IP addresses will be translated into global IP address on FortiBalancer appliance.

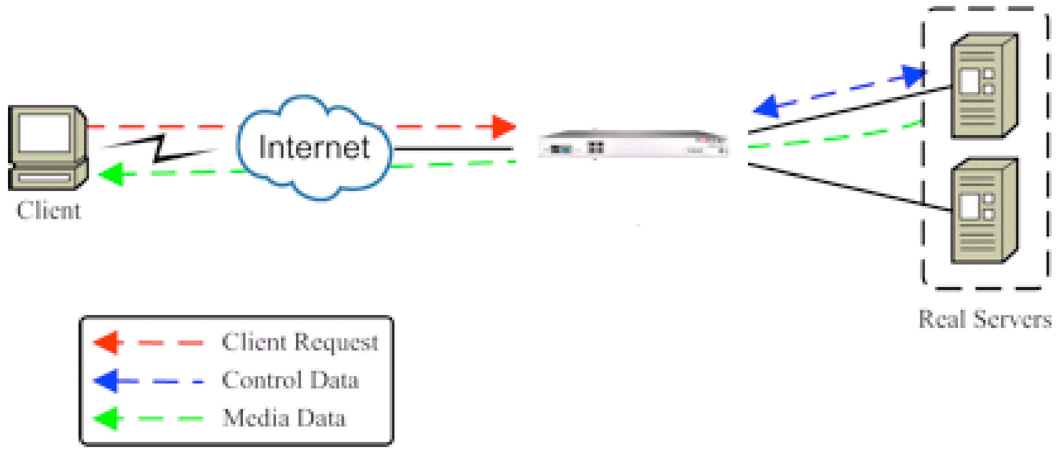


Figure 6-12 RTSP Load Balancing - Dynamic NAT Mode

Table 6-7 General Settings of RTSP Load Balancing

Operation	Command
Configure real services	**slb real rtsp** *<real_name> <ip> [port] [max_conn] [rtsp-tcp|tcp|icmp|script-tcp|script-udp|dns] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** *[threshold] [yes|no]* **slb group method** *<group_name>* **pi** *[hash_bits] [rr|sr|lc] [threshold]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **prox** *[rr|sr|lc] [threshold]* **slb group method** *<group_name>* **snmp** *[weight|cpu] [community] [oidcount] [oid1] [oidweight1] [oid2] [oidweight2] [check_interval]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb virtual {tcp|tcps|udp}** *<virtual_name> <vip> <vport> [arp|noarp] [max_conn]* **slb virtual rtsp** *<virtual_name> <vip> [vport] [mode] [arp|noarp] [max_conn]* **slb virtual {ftp|ftps}** *<virtual_name> <vip> [vport] [max_conn]*

Bind the group to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}*
	slb policy static *<virtual_name> <real_name>*
	slb policy backup *{virtual_name|vlink_name} {group_name|vlink_name}*
	slb policy filetype *<policy_name> <vs_name> <group> <filetype>*

Configuration Example via CLI

➢ Step 1 Define RTSP real services by using the command "**slb real rtsp**"

```
FortiBalancer(config)#slb real rtsp "rs_rtsp1" 10.5.14.90 554 1000 rtsp-tcp 3 3 60
FortiBalancer(config)#slb real rtsp "rs_rtsp2" 10.5.14.91 554 1000 rtsp-tcp 3 3 60
```

➢ Step 2 Define RTSP real service groups

We can use rr (Round Robin), pi (Persistent IP), hi (Hash IP), chi (Consistent Hash IP), and snmp method to choose RTSP real service in one group.

```
FortiBalancer(config)#slb group method "grt1" rr
FortiBalancer(config)#slb group member "grt1" "rs_rtsp1" 1
FortiBalancer(config)#slb group member "grt1" "rs_rtsp2" 1
```

➢ Step 3 Add the real services into the group

```
FortiBalancer(config)#slb group member "grt1" "rs_rtsp1" 1
FortiBalancer(config)#slb group member "grt1" "rs_rtsp2" 1
```

➢ Step 4 Define RTSP virtual services

```
FortiBalancer(config)#slb virtual rtsp "vs_rtsp1" 10.3.14.90 554 nat
FortiBalancer(config)#slb virtual rtsp "vs_rtsp2" 10.3.14.91 7070 nat
```

➢ Step 5 Set the policy

Static policy has higher priority than the default policy.

```
FortiBalancer(config)#slb policy default "vs_rtsp1" "grt1"
FortiBalancer(config)#slb policy static "vs_rtsp2" "rs_rtsp1"
```

6.3.4 Layer 2 IP/MAC-based Load Balancing

6.3.4.1 Two-arm Network Topology

Configuration Guidelines

Before we show how to set this up, we should describe the relative concepts in our system.

Let’s begin to setup the environment for firewall load balance. We will describe several different cases.

To make Layer 2 SLB work, the clients, servers and firewalls should have the default gateway or some static route gateway configured as one of the FortiBalancer appliance’s IP addresses so that the traffic can be forwarded to the FortiBalancer appliance.

For example, the following routes can be added for the clients, servers and firewalls respectively:

Client route:

```
route add –net 172.16.167.0 172.16.162.70 -netmask 255.255.255.0
```

Server route:

```
route add –net 172.16.162.0 172.16.167.73 -netmask 255.255.255.0
```

Firewall1 routes:

```
route add -net 172.16.167.0 172.16.165.73 -netmask 255.255.255.0
route add -net 172.16.162.0 172.16.163.70 -netmask 255.255.255.0
```

Firewall2 routes:

```
route add -net 172.16.167.0 172.16.166.73 -netmask 255.255.255.0
route add -net 172.16.162.0 172.16.164.70 -netmask 255.255.255.0
```

Note: We assume all the systems are Unix-alike. For Windows, different versions of route commands may need to be applied.

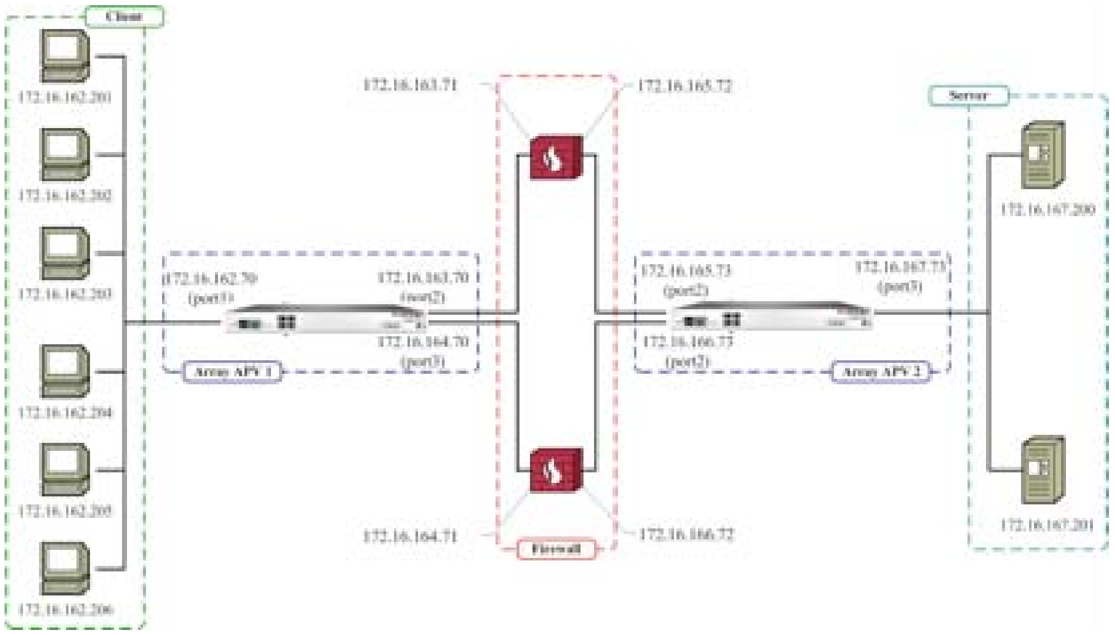


Figure 6-13 Layer 2 IP/MAC-based Load Balancing - Two-arm Network

Note: One real service can only be included by one real service group. Layer 3 real service and Layer 2 real service can not be on the same interface.

Table 6-8 General Settings of Layer 2 IP/MAC-based Load Balancing

Operation	Command
Configure real services	**slb real l2ip** *<real_name> <real_ip>* **slb real l2mac** *<real_name> <real_mac> <output_interface>*
Define group methods	**slb group method** *<group_name>* **{hi|rr|chi}** *[route|direct]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual l2ip** *<virtual_name> <vip> [gateway_ip]*
Bind the group to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}*
Define the additional health check	**slb real health** *<add_hc_name> <real_name> <ip> <port> [http|https|tcp|icmp|dns|ldap|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|rtsp-tcp] [hc_up] [hc_down]*
Configure reflector	**health ipreflect** *<reflector_name> <ip_address> <port> [protocol]*

Configuration Example via CLI

Then we begin to configure the left box FortiBalancer1 according to the above figure.

➢ Step 1 Assign IP address to relative interfaces:

```
FortiBalancer(config)#ip address port1 172.16.162.70 255.255.255.0
FortiBalancer(config)#ip address port2 172.16.163.70 255.255.255.0
FortiBalancer(config)#ip address port3 172.16.164.70 255.255.255.0
```

➢ Step 2 Define Layer 2 real services

We can use IP address to define a real service.

```
FortiBalancer(config)#slb real l2ip rs1 172.16.163.71
FortiBalancer(config)#slb real l2ip rs2 172.16.164.71
```

On the other hand, we can use MAC address to define a real service as well.

```
FortiBalancer(config)#slb real l2mac rs1 00:e0:81:03:36:e4 port2
FortiBalancer(config)#slb real l2mac rs2 00:30:48:81:54:9c port3
```

Note: To get the MAC address, please use relative IP address command for your specific system. For example, if you use Linux, then you can use the command "ifconfig -u" to get the MAC address of NIC.

➢ Step 3 Define the group for the real service and add its members to the group

```
FortiBalancer(config)#slb group method g1 rr direct
```

Note: Layer 2 SLB supports three methods for the group: rr, hi and chi.

When the "**slb group method**" command is used to define a Layer 2 SLB group, a new parameter is introduced as the last argument: "route mode". This parameter is used to route a data packet coming from a Layer 2 real service. Possible values for route mode are: direct and route. "direct" the data packet from a Layer 2 real service will be sent out from the related Layer 2 virtual service's interface directly without bothering any route settings. On the contrary, if the route mode is valued "route", route settings will be used to send the data packet.

➢ Step 4 Add the real services to the group

```
FortiBalancer(config)#slb group member g1 rs1 1
FortiBalancer(config)#slb group member g1 rs2 1
Or
FortiBalancer(config)#slb group member g1 rs1
FortiBalancer(config)#slb group member g1 rs2
```

- Step 5 Define Layer 2 virtual service

```
FortiBalancer(config)#slb virtual l2ip vs1 172.16.162.70
```

- Step 6 Define the SLB policy

```
FortiBalancer(config)#slb policy default vs1 g1
```

Note: Layer 2 SLB only supports default policy.

- Step 7 Add the additional health check on the backend server

```
FortiBalancer(config)#slb real health a1 rs1 172.16.165.73 80 tcp
FortiBalancer(config)#slb real health a1 rs2 172.16.166.73 80 tcp
```

Here, we have finished the configuration on FortiBalancer1 for the Layer 2 IP/MAC based SLB. Now we will begin to configure FortiBalancer2:

- Step 1 Assign IP address to relative interfaces

```
FortiBalancer(config)#ip address port1 172.16.165.73 255.255.255.0
FortiBalancer(config)#ip address port2 172.16.166.73 255.255.255.0
FortiBalancer(config)#ip address port3 172.16.164.73 255.255.255.0
```

- Step 2 Define Layer 2 real services

```
FortiBalancer(config)#slb real l2ip rs1 172.16.165.72
FortiBalancer(config)#slb real l2ip rs2 172.16.166.72
Or
FortiBalancer(config)#slb real l2mac rs1 00:e0:81:03:36:e5 port1
FortiBalancer(config)#slb real l2mac rs2 00:30:48:81:54:9d port2
```

- Step 3 Define the group for the real service

```
FortiBalancer(config)#slb group method g1 rr direct
FortiBalancer(config)#slb group member g1 rs1 1
FortiBalancer(config)#slb group member g1 rs2 1
Or
FortiBalancer(config)#slb group method g1 hi direct
FortiBalancer(config)#slb group member g1 rs1
FortiBalancer(config)#slb group member g1 rs2
```

- Step 4 Add its members to the group

```
FortiBalancer(config)#slb group member g1 rs1 1
FortiBalancer(config)#slb group member g1 rs2 1
Or
FortiBalancer(config)#slb group member g1 rs1
FortiBalancer(config)#slb group member g1 rs2
```

- Step 5 Define Layer 2 virtual service

```
FortiBalancer(config)#slb virtual l2ip vs1 172.16.167.73
```

- Step 6 Define the SLB policy

```
FortiBalancer(config)#slb policy default vs1 g1
```

- Step 7 Add the additional health check on the backend server

```
FortiBalancer(config)#slb real health a1 rs1 172.16.163.70 80 tcp
FortiBalancer(config)#slb real health a1 rs2 172.16.164.70 80 tcp
```

- Step 8 Configure reflector for Layer 2 SLB TCP health check

```
FortiBalancer(config)#health ipreflect aa 0.0.0.0 80 tcp
```

6.3.4.2 Single-arm Network Topology

Configuration Guidelines

Same as two-arm scenario, new routes need to be configured in the client, server and firewalls for packet forwarding:

Client route:

```
route add -net 172.16.167.0 172.16.162.70 -netmask 255.255.255.0
```

Server routes:

```
route ADD 172.16.162.0 MASK 255.255.255.0 172.16.167.73
route ADD 172.16.163.0 MASK 255.255.255.0 172.16.167.73
route ADD 172.16.164.0 MASK 255.255.255.0 172.16.167.73
```

Firewall1 route:

```
route add default 172.16.163.70
```

Firewall2 route:

```
route add default 172.16.164.70
```

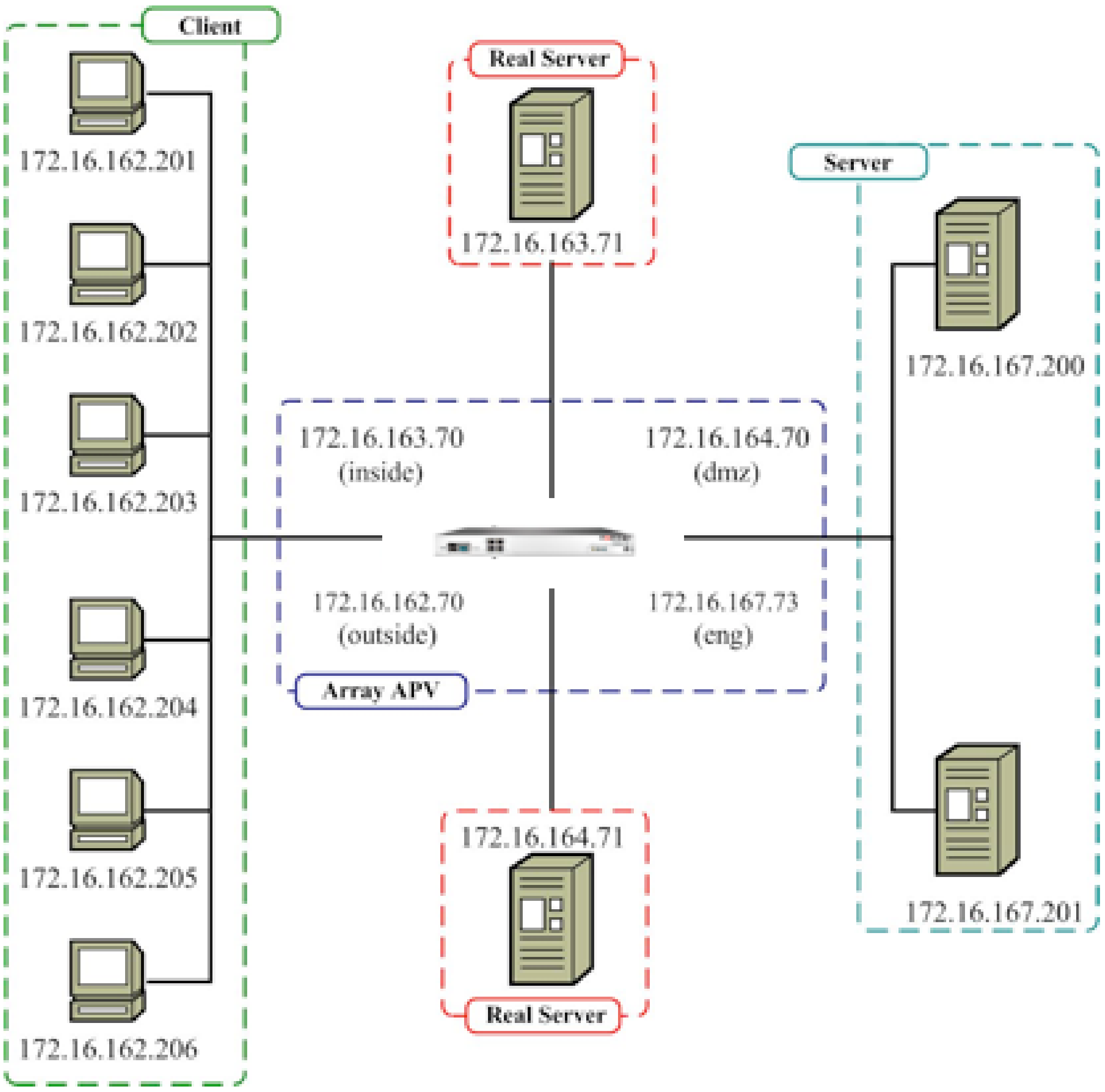


Figure 6-14 IP/MAC-based Load Balancing - Single-arm Network

The general settings for single-arm network are the same as the settings for two-arm network.

Configuration Example via CLI

Then we begin to configure FortiBalancer according to the above figure.

➢ Step 1 Assign IP address to relative interfaces

```
FortiBalancer(config)#ip address "port1" 172.16.162.70 255.255.255.0
FortiBalancer(config)#ip address "port2" 172.16.163.70 255.255.255.0
FortiBalancer(config)#ip address "port3" 172.16.164.70 255.255.255.0
FortiBalancer(config)#ip address "port4" 172.16.167.73 255.255.255.0
```

➢ Step 2 Define Layer 2 real services

```
FortiBalancer(config)#slb real l2ip "r1" 172.16.163.71
FortiBalancer(config)#slb real l2ip "r2" 172.16.164.71
or
FortiBalancer(config)#slb real l2mac r1 00:e0:81:03:36:e4 port2
FortiBalancer(config)#slb real l2mac r2 00:30:48:81:54:9c port3
```

➢ Step 3 Define the group for the real service

```
FortiBalancer(config)#slb group method "g1" "rr" "route"
```

➢ Step 4 Add the real services into the group

```
FortiBalancer(config)#slb group member "g1" "r1" 1
FortiBalancer(config)#slb group member "g1" "r2" 1
or
FortiBalancer(config)#slb group member "g1" "r1"
FortiBalancer(config)#slb group member "g1" "r2"
```

➢ Step 5 Define Layer 2 virtual service

```
FortiBalancer(config)#slb virtual l2ip "v1" 172.16.162.70
FortiBalancer(config)#slb virtual l2ip "v2" 172.16.167.73
```

➢ Step 6 Define the SLB policy

```
FortiBalancer(config)#slb policy default "v1" "g1"
FortiBalancer(config)#slb policy default "v2" "g1"
```

➢ Step 7 Add the additional health check on the backend server

```
FortiBalancer(config)#slb real health a1 r1 172.16.163.71 0 icmp
FortiBalancer(config)#slb real health a1 r2 172.16.164.71 0 icmp
or
FortiBalancer(config)#slb real health a1 r1 172.16.163.70 0 icmp
FortiBalancer(config)#slb real health a1 r2 172.16.164.70 0 icmp
```

6.3.5 Layer 3 IP-based Load Balancing

6.3.5.1 Configuration Guidelines

The commands used to configure Layer 3 SLB are summarized in the following table:

Table 6-9 General Settings of Layer 3 IP-based Load Balancing

Operation	Command
Configure real services	**slb real ip** *<real_name> <ip> [max_conn] [icmp|none] [hc_up] [hc_down] [udp_timeout]*
Define group methods	**lb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** *[threshold] [yes|no]* **slb group method** *<group_name>* **pi** *[hash_bits] [rr|sr|lc] [threshold]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **prox** *[rr|sr|lc] [threshold]* **slb group method** *<group_name> snmp [weight|cpu] [community] [oidcount] [oid1] [oidweight1] [oid2] [oidweight2] [check_interval]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual ip** *<virtual_name> <vip>*

Bind the group (or the real service) to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}*
	slb policy static *<virtual_name> <real_name>*
	slb policy backup *{virtual_name|vlink_name} {group_name|vlink_name}*

6.3.5.2 Configuration Example via CLI

➢ Step 1 Create real services

```
FortiBalancer(config)#slb real ip rip0 10.3.14.10
FortiBalancer(config)#slb real ip rip1 10.3.14.20 1000 none
```

The health check of rip0 defaults to “icmp”.

- Step 2 Define a group for Layer 3 load balancing by using the “**slb group method**” command

```
FortiBalancer(config)#slb group method gip0
```

- Step 3 Add the real services into the group

```
FortiBalancer(config)#slb group member “gip0” “rip0” 1
FortiBalancer(config)#slb group member “gip0” “rip1” 1
```

- Step 4 Create a virtual service

```
FortiBalancer(config)#slb virtual ip vip0 10.3.14.56
```

- Step 5 Associate a group with a virtual service

You may associate the group or the real service to the virtual service of Layer 3 load balancing by using the command “**slb policy default**” or associate the real service to the virtual service by the command “**slb policy static**”.

```
FortiBalancer(config)#slb policy default vip0 gip0
Or
FortiBalancer(config)#slb policy static vip0 rip0
```

6.3.6 Port Range Load Balancing

6.3.6.1 Configuration Guidelines

The commands used to configure Port Range SLB are summarized in the following table:

Table 6-10 General Settings of Port Range Load Balancing

Operation	Command
Configure real services	**slb real tcp** *<real_name> <ip> <port> [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|ldap] [hc_up] [hc_down]* **slb real http** *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real udp** *<real_name> <ip> <port> [max_conn] [hc_up] [hc_down] [timeout] [icmp|script-tcp|script-udp|radius-auth|radius-acct|sip-tcp|sip-udp|dns]* **slb real https** *<real_name> <ip> [port] [max_conn] [https|tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real tcps** *<real_name> <ip> <port> [max_conn] [tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real dns** *<real_name> <ip> <port> [max_conn] [dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name> [algorithm]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb virtual {tcp|tcps|udp}** *<virtual_name> <vip> <vport> [arp|noarp] [max_conn]*

Operation	Command
	slb virtual rtsp *<virtual_name> <vip> [vport] [mode] [arp|noarp] [max_conn]* **slb virtual {ftp|ftps}** *<virtual_name> <vip> [vport] [max_conn]*
Define the port range of a virtual service	**slb virtual portrange** *<virtual_name> <min_port> <max_port> [protocol] [dst|src]*

Bind the group (or the real service) to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}*
	slb policy static *<virtual_name> <real_name>*
	slb policy backup *{virtual_name|vlink_name} {group_name|vlink_name}*

6.3.6.2 Configuration Example via CLI

Herein we use HTTP protocol as an example. The configurations of other protocols are similar.

➢ Step 1 Define static or port range real services

When the real services are static:

```
FortiBalancer(config)#slb real http rhttp0 10.3.14.10
```

The port of HTTP type real service defaults to 80.

```
FortiBalancer(config)#slb real http rhttp1 10.3.14.11 90
```

The port of “rhttp1” is specified to 90.

When the real services are port range services, the health check can only be “icmp” or “none”.

```
FortiBalancer(config)#slb real http rhttp0 10.3.14.10 0 1000 icmp
FortiBalancer(config)#slb real http rhttp1 10.3.14.11 0 1000 none
```

➢ Step 2 Define a group

```
FortiBalancer(config)#slb group method ghttp1
```

➢ Step 3 Add the real services into the group

```
FortiBalancer(config)#slb group member ghttp1 rhttp0
FortiBalancer(config)#slb group member ghttp1 rhttp1
```

➢ Step 4 Create a virtual service

```
FortiBalancer(config)#slb virtual http vhttp0 10.3.14.50 0
```

➢ Step 5 Define the port range for the virtual service

At most three port ranges can be defined for an SLB virtual service.

```
FortiBalancer(config)#slb virtual portrange vhttp0 80 90
FortiBalancer(config)#slb virtual portrange vhttp0 8000 9000
```

In the above example, all data packets with destination IP address to be “10.3.14.50” and port falling into the range 80-90 or 8000-9000 will be handled by the port range virtual service “vhttp0”.

Note: Port range real services and static real services can not be added into one group.

➢ Step 6 Associate a group or a real service with a virtual service

Associate the group to the port-range virtual service with the command “**slb policy default**” and associate the real service to the port-range virtual service with the command “**slb policy static**”.

```
FortiBalancer(config)#slb policy default vhttp0 ghttp1
Or
FortiBalancer(config)#slb policy static vhttp0 rhttp1
```

6.3.7 Terminal Server Load Balancing

6.3.7.1 Configuration Guidelines

The commands used to configure Terminal Server Load Balancing are summarized in the following table:

Table 6-11 General Settings of Terminal Server Load Balancing

Operation	Command
Create RDP real services	**slb real rdp** *<real_name> <ip> [port] [maxconn] [tcp|icmp] [hc_up] [hc_down]*
Create RDP groups	**slb group method** *<group_name>* **rdprt** *[rr|sr|lc]*
Add the real services into the group	**slb group member** *<group_name> <real_name>*
Create RDP virtual services	**slb virtual rdp** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]*
Associate the real server group with virtual services	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}*

6.3.7.2 Configuration Example via CLI

➢ Step 1 Create RDP real services

The default port number for RDP real services is 3389.

```
FortiBalancer(config)#slb real rdp rs1 172.16.69.191 3389 1000 icmp 3 3
FortiBalancer(config)#slb real rdp rs2 172.16.69.192 3389 1000 icmp 3 3
```

Note: For the RDP real services, only the “icmp” and “tcp” types of health check can be used.

➢ Step 2 Create RDP groups

```
FortiBalancer(config)#slb group method g1 rdprt rr
```

➢ Step 3 Add the real service into the group

```
FortiBalancer(config)#slb group member g1 rs1
FortiBalancer(config)#slb group member g1 rs2
```

➢ Step 4 Create RDP virtual services

The default port number for RDP virtual services is 3389.

```
FortiBalancer(config)#slb virtual rdp vs1 172.16.69.171 3389 arp 0
```

➢ Step 5 Associate the RDP group with the virtual services

```
FortiBalancer(config)#slb policy default vs1 g1
```

Note: RDP only supports the Default group policy.

6.3.8 Policy Nesting

6.3.8.1 Configuration Guidelines

The commands used to configure Policy Nesting are summarized in the following table:

Table 6-12 General Settings of Policy Nesting

Operation	Command
Configure real services	**slb real http** *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real https** *<real_name> <ip> [port] [max_conn] [https|tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|dns] [hc_up] [hc_down]*
Define group methods	**slb group method** *<group_name> [algorithm]*
Add the real servers into the group	**slb group member** *<group_name> <real_name> [weight]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb virtual {tcp|tcps|udp}** *<virtual_name> <vip> <vport> [arp|noarp] [max_conn]* **slb virtual rtsp** *<virtual_name> <vip> [vport] [mode] [arp|noarp] [max_conn]* **slb virtual {ftp|ftps}** *<virtual_name> <vip> [vport] [max_conn]*
Create the vlink	**slb vlink** *<vlink_name>*
Bind the group to the virtual service or the vlink	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_name> <real_name>* **slb policy icookie** *<policy_name> {virtual_name|vlink_name} <group_name> <precedence>* **slb policy rcookie** *<policy_name> {virtual_name|vlink_name} <group_name> <precedence>* **slb policy persistent cookie** *<policy_name> {virtual_name|vlink_name} <group_name> <cookie_name> <precedence>* **slb policy persistent url** *<policy_name> {virtual_name|vlink_name} <group_name> <url_tag> <precedence>*

6.3.8.2 Configuration Example via CLI

To help you better understand the example, the following graphic shows the logical relationship among VIPs, vlinks and groups.

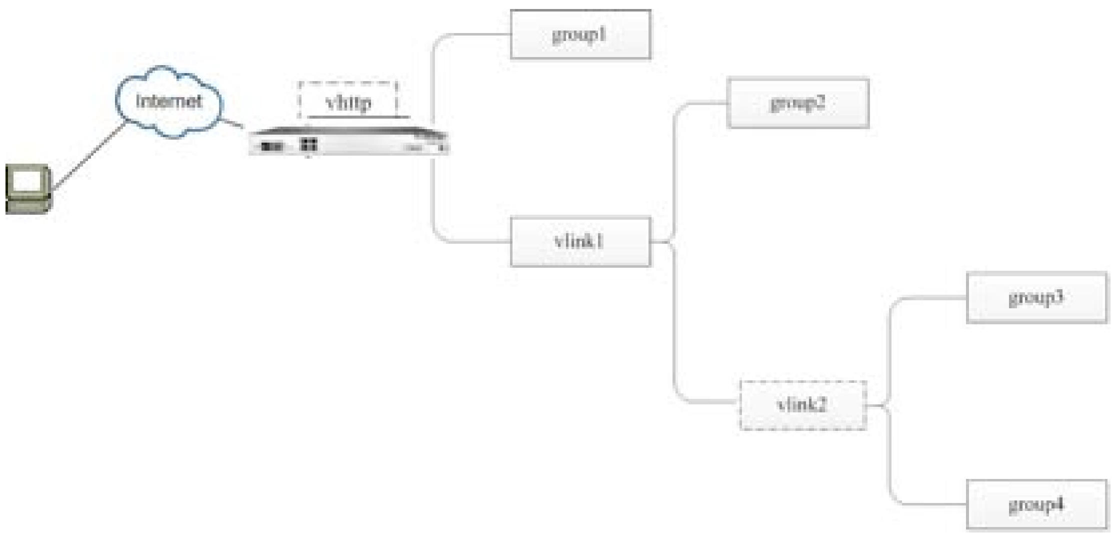


Figure 6-15 Policy Nesting

➢ Step 1 Define the real services

Six HTTP real services are configured in this example:

```
FortiBalancer(config)#slb real http "rhttp1" 172.16.85.109 80 1000 http 3 3
FortiBalancer(config)#slb real http "rhttp2" 172.16.85.110 8081 1000 http 3 3
FortiBalancer(config)#slb real http "rhttp3" 172.16.85.111 8080 1000 http 3 3
FortiBalancer(config)#slb real http "rhttp4" 172.16.85.112 112 1000 tcp 3 3
FortiBalancer(config)#slb real http "rhttp5" 172.16.85.113 113 1000 tcp 3 3
FortiBalancer(config)#slb real http "rhttp6" 172.16.85.114 114 1000 tcp 3 3
```

- Step 2 Create the groups

```
FortiBalancer(config)#slb group method group1 rr
FortiBalancer(config)#slb group method group2 rr
FortiBalancer(config)#slb group method group3 rr
FortiBalancer(config)#slb group method group4 rr
```

- Step 3 Add the real services into the groups

```
FortiBalancer(config)#slb group member group1 rhttp1 1
FortiBalancer(config)#slb group member group1 rhttp2 2
FortiBalancer(config)#slb group member group2 rhttp3 1
FortiBalancer(config)#slb group member group2 rhttp4 2
FortiBalancer(config)#slb group member group3 rhttp5 1
FortiBalancer(config)#slb group member group4 rhttp6 1
```

- Step 4 Create a virtual service

```
FortiBalancer(config)#slb virtual http vhttp 172.16.63.109 80 arp 0
```

- Step 5 Create the vlinks

```
FortiBalancer(config)#slb vlink vlink1
FortiBalancer(config)#slb vlink vlink2
```

- Step 6 Associate a group to a virtual service or a vlink

```
FortiBalancer(config)#slb policy qos network policy1 vhttp vlink1 192.168.2.3 255.255.255.255
1
FortiBalancer(config)#slb policy default vhttp group1
FortiBalancer(config)#slb policy qos url policy2 vlink1 vlink2 “news” 1
FortiBalancer(config)#slb policy default vlink1 group2
FortiBalancer(config)#slb policy qos url policy3 vlink2 group3 “sport” 1
FortiBalancer(config)#slb policy default vlink2 group4
```

In this example:

- If the source IP address of a request is within the sub network, it will match policy1 and go to vlink1. Otherwise, the request will be distributed to group1.
- If the request goes to vlink1 and there is “news” in the URL, it will match policy2 and go to vlink2. Otherwise, the request will be forwarded to group2.
- If the request goes to vlink2 and there is “sports” in the URL, it will match policy3 and go to group3. Otherwise, the request will be sent to group 4.

6.3.9 SLB Session Persistence Configuration

6.3.9.1 Independently Applying the Persistence Method for Session Persistence (ip)

- **Configuration purpose:**

To implement session persistence by independently using the persistence method with the client IP address as the session ID. Once configured, the FortiBalancer appliance will:

- Send the first access request from a client to the real server selected by the first choice method (rr in this example).
- Obtain and record the client IP address as the unique session ID.
- Send subsequent requests from the client to the same real server. If the client remains idle (for example, no new client requests are detected) for 5 minutes, the FortiBalancer appliance will terminate the client's session and clear the session ID.

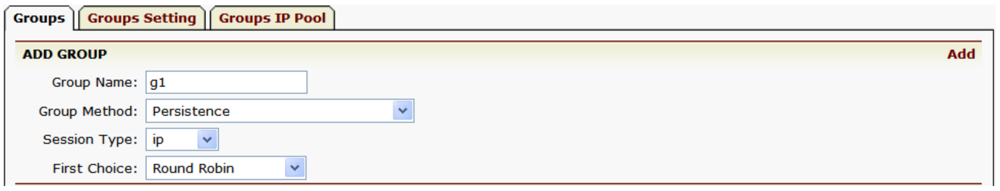
The detailed configuration procedure is as follows:

➢ **Prerequisites:**

- Layer 4 or 7 real services r1 and r2 are already defined. In this example, real services are of the HTTP type.
- A Layer 4 or 7 virtual service v1 is already defined. In this example, the virtual service is of the HTTP type.

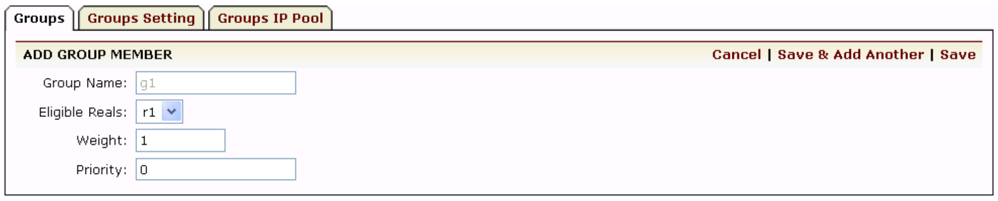
➢ **web UI:**

1. Select **Server Load Balance > Groups > Groups**. In the **Add Group** area, select "Persistence" from the **Group Method** drop-down list and "ip" from the **Session Type** drop-down list. Specify other parameters as required. Click the **Add** action link.



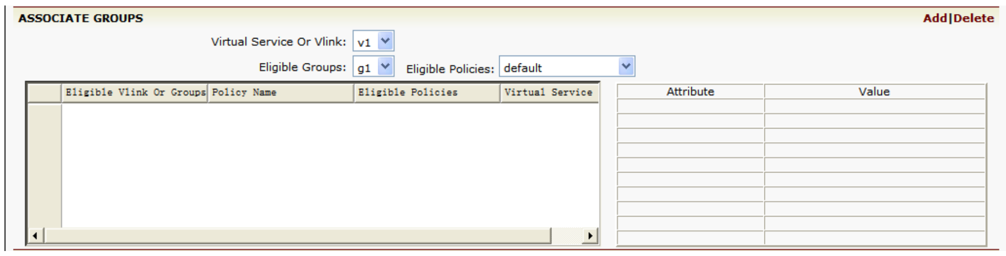
Group Name:	g1
Group Method:	Persistence
Session Type:	ip
First Choice:	Round Robin

2. In the **Group List** area, double-click the group. In the **Group Members** area of the displayed page, click the **Add** action link. In the **Add Group Member** area of the displayed page, specify the required parameters and click the **Save** action link to save the configuration.



Group Name:	g1
Eligible Reals:	r1
Weight:	1
Priority:	0

3. Select **Server Load Balance > Virtual Services > Virtual Services**. In the **Virtual Service List** area, double-click the virtual service. In the **Associate Groups** area of the displayed page, specify the required parameters and click the **Add** action link.

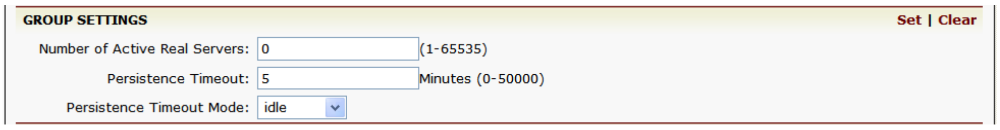


Virtual Service Or Vlink: v1
Eligible Groups: g1 Eligible Policies: default

Eligible Vlink Or Groups	Policy Name	Eligible Policies	Virtual Service

Attribute	Value

4. Select **Server Load Balance > Groups > Groups**. In the **Group List** area, double-click the group. In the **Group Settings** area of the displayed page, specify the **Persistence Timeout** and **Persistence Timeout Mode** parameters. Click the **Set** action link.



Number of Active Real Servers:	0 (1-65535)
Persistence Timeout:	5 Minutes (0-50000)
Persistence Timeout Mode:	idle

➢ **CLI**:

1. Execute the following command to configure the service group and persistence method:

   slb group method *<group_name>* **persistence** *<session_id_type> [rr|sr|lc] [threshold]*

   For example:

   ```
   FortiBalancer(config)#slb group method g1 persistence ip rr
   ```

2. Execute the following command to add real services to the service group:

   slb group member *<group_name> <real_name> [weight]*

   For example:

   ```
   FortiBalancer(config)#slb group member g1 r1 1 0
   FortiBalancer(config)#slb group member g1 r2 1 0
   ```

3. Execute the following command to bind the service group and virtual service with the default policy:

   slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

   For example:

   ```
   FortiBalancer(config)#slb policy default v1 g1
   ```

4. Execute the following command to configure the time out mode:

   slb persistence timeout *<timeout _ minutes> [group_name] [idle|timeout]*

   For example:

   ```
   FortiBalancer(config)#slb persistence timeout 5 g1 idle
   ```

6.3.9.2 Independently Applying the Persistence Method for Session Persistence (string)

➢ **Configuration purpose:**

To implement session persistence by independently using the persistence method with a specified string (obtained from the HTTP response cookie) as the session ID. Once configured, the FortiBalancer appliance will:

- Send the first access request from a client to the real server selected by the first choice method (rr in this example).
- Obtain the “mycookie” value from the real server response and record it as the session ID.
- When receiving subsequent requests from the client, check whether the value of “telnum” in the request URL Query matches with the session ID. If yes, send the request to the same real server. If the client remains idle (for example, no new client requests are detected) for 5 minutes, the FortiBalancer appliance will terminate the client’s session and clear the session ID.

The detailed configuration procedure is as follows:

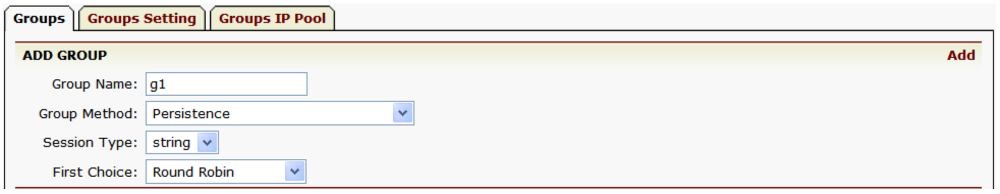
➢ **Prerequisites:**

- Layer 4 or 7 real services r1 and r2 are already defined. In this example, the virtual service is of the HTTP type.

- A Layer 4 or 7 virtual service v1 is already defined. In this example, the virtual service is of the HTTP type.

➢ **web UI:**

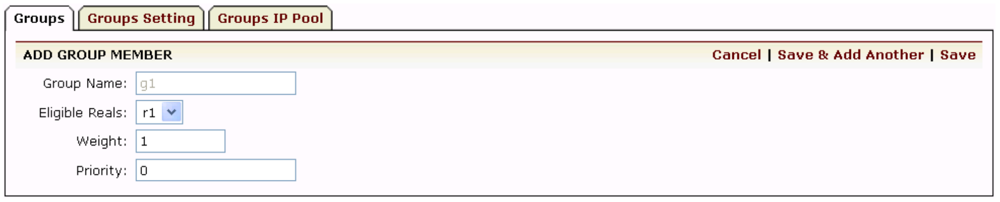
1. Select **Server Load Balance > Groups > Groups**. In the Add Group area, select "Persistence" from the **Group Method** drop-down list and "string" from the **Session Type** drop-down list. Specify other parameters as required. Click the **Add** action link.



ADD GROUP

Group Name:	g1
Group Method:	Persistence
Session Type:	string
First Choice:	Round Robin

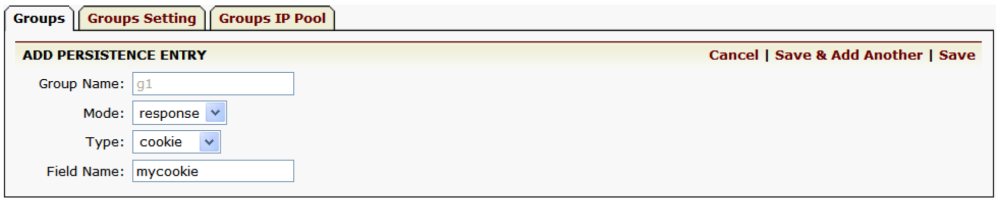
2. In the **Group List** area, double-click the group. In the **Group Members** area of the displayed page, click the **Add** action link. In the **Add Group Member** area of the displayed page, specify the required parameters and click **Save** to save the configuration.



ADD GROUP MEMBER

Group Name:	g1
Eligible Reals:	r1
Weight:	1
Priority:	0

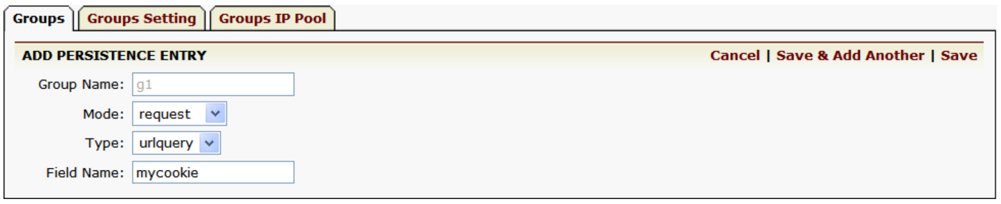
3. In the **Persistence List** area, click the **Add** action link. In the **Add Persistence Entry** area of the displayed page, specify the response cookie-related parameters and click **Save** to save the configuration.



ADD PERSISTENCE ENTRY

Group Name:	g1
Mode:	response
Type:	cookie
Field Name:	mycookie

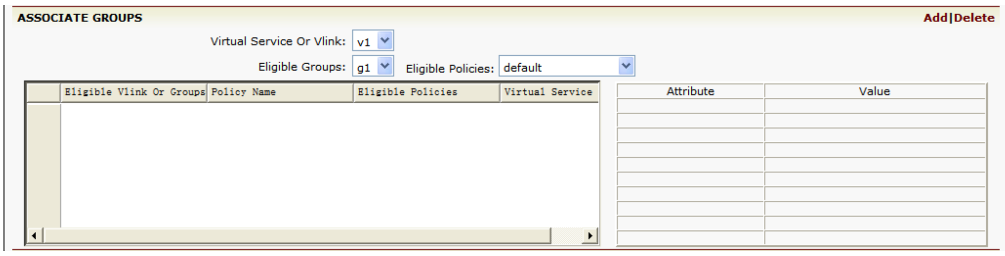
4. In the **Persistence List** area, click the **Add** action link. In the **Add Persistence Entry** area of the displayed page, specify the request-related parameters and click **Save** to save the configuration.



ADD PERSISTENCE ENTRY

Group Name:	g1
Mode:	request
Type:	urlquery
Field Name:	mycookie

5. Select **Server Load Balance > Virtual Services > Virtual Services**. In the **Virtual Service List** area, double-click the virtual service. In the **Associate Groups** area of the displayed page, specify the required parameters and click the **Add** action link.

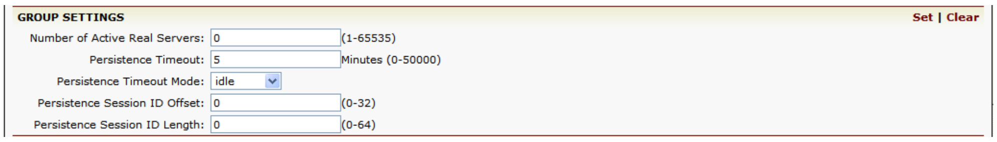


ASSOCIATE GROUPS

Virtual Service Or Vlink: v1

Eligible Groups: g1 Eligible Policies: default

6. Select **Server Load Balance > Groups > Groups**. In the **Group List** area, double-click the group. In the **Group Settings** area of the displayed page, specify the **Persistence Timeout** and **Persistence Timeout Mode** parameters. Click the **Set** action link.



GROUP SETTINGS		Set | Clear
Number of Active Real Servers:	0	(1-65535)
Persistence Timeout:	5	Minutes (0-50000)
Persistence Timeout Mode:	idle	
Persistence Session ID Offset:	0	(0-32)
Persistence Session ID Length:	0	(0-64)

➢ **CLI:**

1. Execute the following command to configure the service group and persistence method:

slb group method *<group_name>* **persistence** *<session_id_type> [rr|sr|lc] [threshold]*

For example:

```
FortiBalancer(config)#slb group method g1 persistence string rr
```

2. Execute the following command to add real services to the service group:

slb group member *<group_name> <real_name> [weight]*

For example:

```
FortiBalancer(config)#slb group member g1 r1 1 0
FortiBalancer(config)#slb group member g1 r2 1 0
```

3. Execute the following commands to obtain the value of "mycookie" in the response from the real server as the session ID:

slb group persistence request urlquery *<group_name> <query_name>*
slb group persistence response cookie *<group_name> <cookie_name>*

For example:

```
FortiBalancer(config)#slb group persistence request urlquery g1 telnum
FortiBalancer(config)#slb group persistence response cookie g1 mycookie
```

4. Execute the following command to bind the service group and virtual service with the default policy:

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

For example:

```
FortiBalancer(config)#slb policy default v1 g1
```

5. Execute the following command to configure the time out mode:

slb persistence timeout *<timeout_minutes> [group_name] [idle|timeout]*

For example:

```
FortiBalancer(config)#slb persistence timeout 5 g1 idle
```

Note:

Before configuring the FortiBalancer to implement session persistence based on the HTTP cookie, you need to specify the cookie in the real service configurations to ensure that the HTTP response can carry the corresponding cookie.

6.3.9.3 Persistence Method Collaborating with an SLB Persistence Policy

➢ **Configuration purpose:**

To implement session persistence by applying both the persistence method and a Layer 7 SLB persistence policy. In this case, the session ID is obtained through the policy.

The following table describes the configuration for the method and policy:

Table 6-13 Configurations of the Method and Policy

Item	Configurations
Policy	• The header policy is configured to obtain the content of "x-up-calling-line-id" in the HTTP request header as the target string. • The default policy is configured.
Persistence method	• The first choice method is round robin (rr). • The session ID is of the string type. • Whether to obtain the session ID from the client request or server response is not specified. • The offset and ID length for obtaining the session ID are specified.

Once configured, the FortiBalancer appliance will:

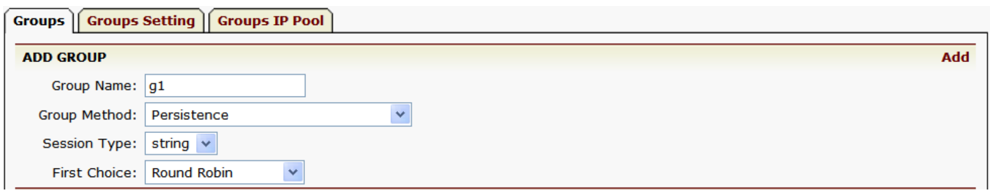
- If the request from the client matches with the header policy, the FortiBalancer appliance will obtain the session ID based on the offset and ID length from the content of "x-up-calling-line-id".
- If the request from the client matches with the default policy, the FortiBalancer appliance will use the first choice method to direct the request to a real server and not implement session persistence.

➢ **Prerequisites:**

- Layer 4 or 7 real services r1 and r2 are already defined. In this example, the real services are of the HTTP type.
- A Layer 4 or 7 virtual service v1 is already defined. In this example, the virtual service is of the HTTP type.

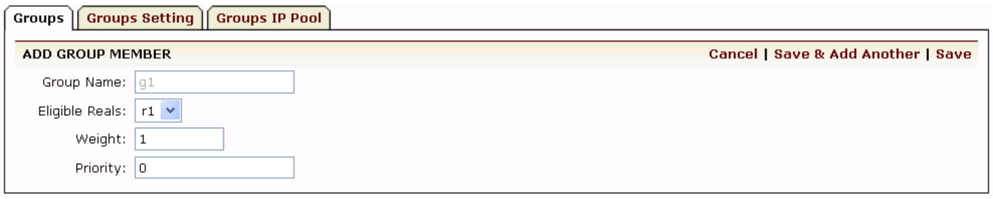
➢ **web UI:**

1. Select **Server Load Balance > Groups > Groups**. In the **Add Group** area, select "Persistence" from the **Group Method** drop-down list and "string" from the **Session Type** drop-down list. Specify other parameters as required. Click the **Add** action link.



Group Name: g1
Group Method: Persistence
Session Type: string
First Choice: Round Robin

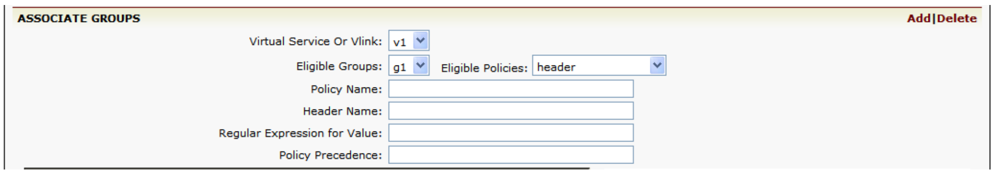
2. In the **Group List** area, double-click the group. In the **Group Members** area of the displayed page, click the **Add** action link. In the **Add Group Member** area of the displayed page, specify the required parameters and click **Save** to save the configuration.


Groups | Groups Setting | Groups IP Pool

ADD GROUP MEMBER — Cancel | Save & Add Another | Save

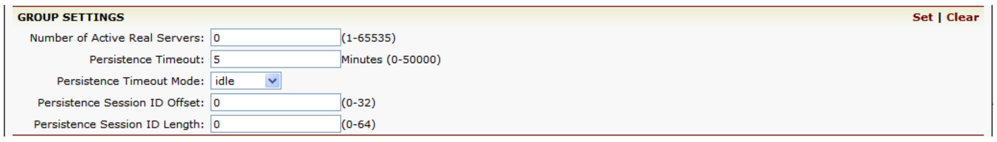
Group Name:	g1
Eligible Reals:	r1
Weight:	1
Priority:	0

3. Select **Server Load Balance > Virtual Services > Virtual Services**. In the **Virtual Service List** area, double-click the virtual service. In the **Associate Groups** area of the displayed page, specify the required parameters and click the **Add** action link.


ASSOCIATE GROUPS — Add|Delete

Virtual Service Or Vlink:	v1
Eligible Groups:	g1
Eligible Policies:	header
Policy Name:	
Header Name:	
Regular Expression for Value:	
Policy Precedence:	

4. Select **Server Load Balance > Groups > Groups**. In the **Group List** area, double-click the group. In the **Group Settings** area of the displayed page, specify the **Persistence Timeout**, **Persistence Timeout Mode**, **Persistence Session ID Offset,** and **Persistence Session ID Length** parameters. Click the **Set** action link.


GROUP SETTINGS — Set | Clear

Number of Active Real Servers:	0	(1-65535)
Persistence Timeout:	5	Minutes (0-50000)
Persistence Timeout Mode:	idle	
Persistence Session ID Offset:	0	(0-32)
Persistence Session ID Length:	0	(0-64)

➢ **CLI:**

1. Execute the following command to configure the service group and persistence method:

```
slb group method <group_name> persistence <session_id_type> [rr|sr|lc]
[threshold]
```

For example:

```
FortiBalancer(config)#slb group method g1 persistence string rr
```

2. Execute the following command to add real services to the service group:

```
slb group member <group_name> <real_name> [weight]
```

For example:

```
FortiBalancer(config)#slb group member g1 r1 1 0
FortiBalancer(config)#slb group member g1 r2 1 0
```

3. Execute the following command to bind the service group and virtual service with the header and default policies:

```
slb policy header <policy_name> {virtual_name|vlink_name}
{group_name|vlink_name} <header_name> <header_pattern> <precedence>
slb policy default {virtual_name|vlink_name} {group_name|vlink_name}
```

For example:

```
FortiBalancer(config)#slb policy header p1 v1 g1 "x-up-calling-line-id" "^1" 1
FortiBalancer(config)#slb policy default v1 g1
```

4. Execute the following command to configure the offset and length for the session ID.

```
slb group persistence value <group_name> <offset> [session_id_length]
```

For example:

```
FortiBalancer(config)#slb group persistence request header g1 abc user y 0
FortiBalancer(config)#slb group persistence value g1 4 3
```

5. Execute the following command to configure the time out mode:

```
slb persistence timeout <timeout_minutes> [group_name] [idle|timeout]
```

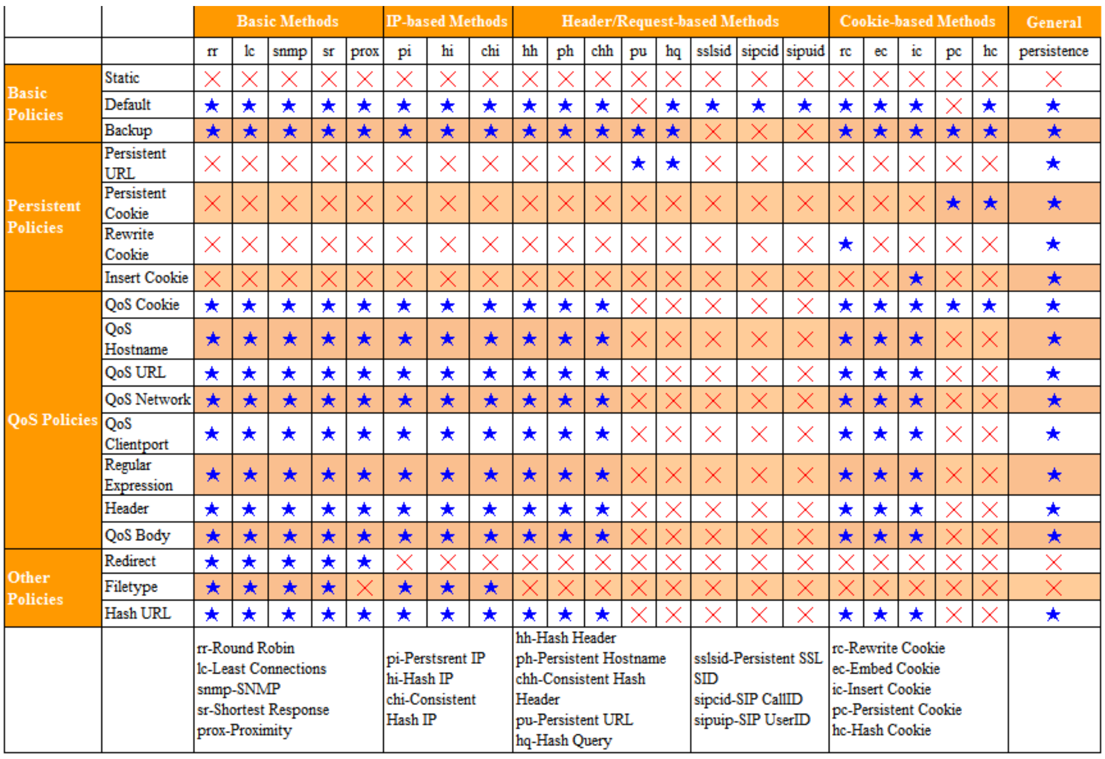
   For example:

```
FortiBalancer(config)#slb persistence timeout 5 g1 idle
```

6.4 SLB Summary

SLB Type	Priority (1 is the highest)	Virtual Service	Real Service	Health check	Scenarios
Layer 7 HTTP/HTTPS	2	IP + Port + proto (HTTP, HTTPS)	IP + Port + proto (HTTP, HTTPS)	None HTTP HTTPS TCP TCPS ICMP Additional Script	1. Balance traffic according to application protocol headers. e.g. HTTP headers 2. Cache feature is needed
Layer 7 DNS	2	IP + Port + proto (DNS)	IP + Port + proto (DNS)	None DNS ICMP Additional Script	DNS requests DNS cache feature can be applied for better performance
Layer 7 FTP	2	IP + Port + proto (FTP)	IP + Port + proto (FTP)	None TCP ICMP Additional Script	FTP traffic
Layer 7 SIP	2	IP + Port + proto (SIP-TCP, SIP-UDP)	IP + Port + proto (SIP-TCP, SIP-UDP)	None TCP TCPS ICMP Additional Script SIP-TCP SIP-UDP	Balance VOIP traffic
Layer 7 RTSP	2	IP + Port + proto (RTSP)	IP + Port + proto (RTSP)	None TCP ICMP Additional Script RTSP-TCP	Balance real time media traffic
Layer 4	2	IP + port	IP + Port	None TCP TCPS ICMP Additional Script	1. Balance traffic according to TCP/UDP headers. 2.TCP port or UDP port is specified to determine a particular service
Port range (for Layer 7)	3	Layer 7 VS + Port range	Layer 7 RS Layer 7 RS (0 port)	Non-zero port RS: Layer 7 health check Zero port RS: ICMP Additional	In addition to Layer 7 SLB, cross-port and dynamic port application traffic balance is supported
Port range (for Layer 4)	3	Layer 4 VS + Port range	Layer 4 RS Layer 4 RS (0 port)	Non-zero port RS: Layer 4 health check Zero port	In addition to Layer 4 SLB, cross-port and dynamic port application traffic balance is supported

SLB Type	Priority (1 is the highest)	Virtual Service	Real Service	Health check	Scenarios
				RS: ICMP Additional	
Layer 3	4	IP	IP	None ICMP Additional	In addition to port range SLB, cross-protocol application traffic balance is supported. Currently, only TCP and UDP protocol are supported
Layer 2	1	IP + port ranges	IP, MAC	ARP Additional (only ICMP)	1. The backend real services do not have usable IP addresses so that the traffic cannot be balanced according to IP addresses; 2. The backend real services are not the destination of the input traffic (e.g. virus scanners check every packet before forwarding it to the real destination).



		Basic Methods					IP-based Methods			Header/Request-based Methods								Cookie-based Methods					General
		rr	lc	snmp	sr	prox	pi	hi	chi	hh	ph	chh	pu	hq	sslsid	sipcid	sipuid	rc	ec	ic	pc	hc	persistence
Basic Policies	Static	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Default	★	★	★	★	★	★	★	★	★	★	★	✗	★	★	★	★	★	★	★	✗	★	★
	Backup	★	★	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	★	★	★	★	★	★
Persistent Policies	Persistent URL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	★	★	✗	✗	✗	✗	✗	✗	✗	✗	★
	Persistent Cookie	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	★	★	★
	Rewrite Cookie	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	★	✗	✗	✗	✗	★
	Insert Cookie	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	★	✗	✗	★
QoS Policies	QoS Cookie	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	★	★	★
	QoS Hostname	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
	QoS URL	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
	QoS Network	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
	QoS Clientport	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
	Regular Expression	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
	Header	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
	QoS Body	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
Other Policies	Redirect	★	★	★	★	★	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Filetype	★	★	★	★	✗	★	★	★	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Hash URL	★	★	★	★	★	★	★	★	★	★	★	✗	✗	✗	✗	✗	★	★	★	✗	✗	★
		rr-Round Robin lc-Least Connections snmp-SNMP sr-Shortest Response prox-Proximity					pi-Perstsrent IP hi-Hash IP chi-Consistent Hash IP			hh-Hash Header ph-Persistent Hostname chh-Consistent Hash Header pu-Persistent URL hq-Hash Query					sslsid-Persistent SSL SID sipcid-SIP CallID sipuip-SIP UserID			rc-Rewrite Cookie ec-Embed Cookie ic-Insert Cookie pc-Persistent Cookie hc-Hash Cookie					

★ The policy can collaborate with the group method.
✗ The policy cannot collaborate with the group method.

Note: DirectFWD does not support Shortest Response (sr).

Chapter 7 Reverse Proxy Cache

7.1 Overview

This chapter will cover the concepts and strategies of using the Reverse Proxy Cache to better enhance the overall speed and performance of your Web servers. Using the cache function will improve website performance and throughput, and will also reduce server load by caching heavily requested data in the temporary memory of the FortiBalancer appliance.

7.1 Understanding Reverse Proxy Cache

7.1.1 How Reverse Proxy Cache Works

The Reverse Proxy Cache is located right in front of Web servers. It receives the requests from clients all over the Internet and responds to these requests by working with the Web servers.

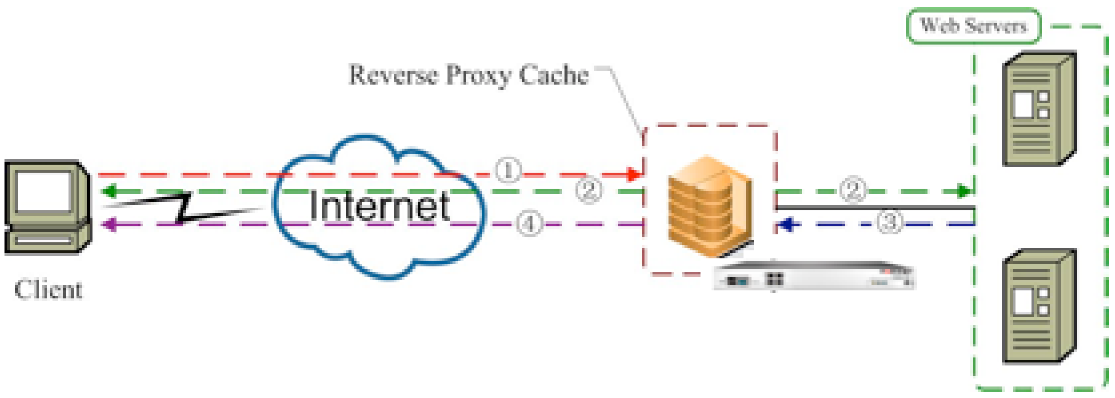


Figure 7-1 Reverse Proxy Cache Working Mechanism

1. The client sends a request to the FortiBalancer appliance, requesting for a file on the Web server. The FortiBalancer appliance will forward the request to the cache module for processing.
2. If the requested content has been cached on the FortiBalancer appliance and the cache does not expire (cache hit), the FortiBalancer appliance will send the file copy to the client directly without forwarding the request to the Web server. If the requested content does not exist in cache (cache miss), the request will be forwarded to the Web server for processing.
3. The Web server responds to the FortiBalancer appliance with the requested content.
4. The FortiBalancer appliance responds to the client with the requested content, and caches the content. Future requests for the same content will be responded directly from the FortiBalancer appliance cache module.

The default behavior of the cache is to send the cached object to the client while the cache is being filled with new objects.

The maximum size of the cache objects depends on different system memories of the FortiBalancer appliances. See the table below:

Table 7-1 Max Size of Cache Object

System Memory	Max Size of Cache Object
4GB	10240KB (10MB)
8GB	20480KB (20MB)
16GB	40960KB (40MB)

7.1.2 Advantages of Reverse Proxy Cache

Compared with traditional cache functions, the Reverse Proxy Cache, without making any compromise on the overall stability and performance, provides a smarter, high-efficient and personalized configuration platform for administrators to more flexibly adjust the FortiBalancer appliances to addressing the demands of different websites. This helps administrators improves the response ability of the websites, reduces the load of Web servers and delivers perfect user experience of visiting the websites.

The Reverse Proxy Cache function is mainly featured with the following advantages:

➢ **Improved performance**

- The cache function is an independent module in the OS. Turning it off will not impact the functionality of other modules in the system.
- The cache module strictly controls its memory consumption under 25% of the total system memory.
- The ability to cache both the compressed and uncompressed contents allows the FortiBalancer to send compressed contents to appropriate clients without having to involve the compression engine. This greatly enhances compression throughput.

➢ **Outstanding stability**

- If any error occurs to the cache module, administrators can turn off the module immediately, which will not affect the running of other system functions.
- All cache tuning parameters now use the “**cache filter**” mechanism, and the global control parameters are reduced. This new approach gives administrators more flexibility and control and minimizes confusion during configuration.

➢ **Intelligent monitoring**

- The FortiBalancer process monitor also monitors the cache (in addition to the reverse proxy). If it detects any issues (or if the cache process crashes), it will restart the cache after appropriate cleanup.

➢ **Flexible configuration**

- Since the cache is a separate process, it can be updated in place using the “**component update**” mechanism.
- The statistics from “**show statistics cache**” are more detailed and are designed to allow administrators to get data that would help them understand how the FortiBalancer is making caching decisions. This should help the customer tune the FortiBalancer or their website to optimize performance.
- The “**cache filter**” mechanism reduces global control parameters, which increases the precision and flexibility of command control by administrators.
- The cache can now be switched on/off on a per-virtual site basis.

7.1.3 Cacheability of Contents

The HTTP traffic falls into two categories: cacheable contents and non-cacheable contents. The cacheability of contents depends on the information within the HTTP headers. The reverse-proxy cache will check the request and response HTTP headers to make cache decisions. If the response for a request is cached, the subsequent request for the same object will be served from the cache instead of from a backend server.

By default, if there are no HTTP headers that restrict caching for an object, the reverse-proxy cache will cache the content. The following are the more popular cache-control headers that will control if the content will be cached and if so, for how long.

Cache-Control: public

The public keyword indicates that any available and configured cache may store the content.

Cache-Control: private

This directive is intended for a single user and will not be cached by the reverse-proxy cache.

Cache-Control: no-cache

This directive lets the reverse-proxy cache know that it can cache the content and can only use the cached content if the appliance first re-validates the content with the origin server.

Expires: Tue, 30 Oct 2010 14:00:00 GMT

This header tells the cache when the content will expire and when to re-fetch it from an origin server when the request for that object is made. In this example it tells the cache to make the content expire on Tuesday, 30 Oct 2010 14:00:00 GMT.

7.1.3.1 Cacheable Contents

Any content with Cache-Control directives which allow caching of the content will always be cached. If the content does not contain a Cache-Control directive, then it will always be cached and will not be re-validated until it is manually flushed from the cache. If the content does not contain an “Expires” header, after it expires, the FortiBalancer appliance will re-validate the content with the origin server and update the content when it is requested next time.

7.1.3.2 Non-Cacheable Contents

Content that has Cache-Control headers which restrict caching of the content will not be cached. Responses to requests with cookies are not served from cache, unless configured by the user to do so.

7.1.4 Cache Filter

The Cache Filter mechanism helps administrators realize more precise control over the cache module with simple cache configurations, such as whether to cache the contents requested by a specific host or not and how long the cached content will live. The priority of the control parameters in cache filter is higher than the peer parameters defined in the Cache-Control header.

The following gives several application examples of cache filter. For detailed configuration examples, refer to the section Reverse Proxy Cache Configuration and the FortiBalancer CLI Reference.

- Cache all “*.jpg” files from the host “www.xyz.com”, and the TTL of cache contents is 200,000 seconds.
- Only cache the image files in common formats, such as JPG, GIF or BMP.
- For the cache objects from the same host, some can be cached by following the cache filter rules, and others can be cached by following the definitions in the cache control header, such as the TTL of the cache.
- Ignore the specific type of query strings in the request URL when looking up for an object in cache.

7.1.5 Cache Expiration Time

Three types of cache expiration time are involved during the cache process:

- The expiration time defined by the "Expires" field in the HTTP header;
- The global cache expiration time configured via the command **"cache settings expire** *{hh:mm:ss|seconds}*";
- The TTL time specified by the "ttl" parameter in the command "**cache filter rule** *<host_name> <url> {cache|urlquery|ttl}*".

The priorities of the three expiration times are as follows:

1. The expiration time configured in "**cache filter rule**" will be used first.
2. If the "ttl" parameter is not specified, the global expiration time specified by "**cache settings expire**" command will apply.
3. For the cache content that does not match any cache filter rule, the expiration time defined in the HTTP header will be applied.
4. If no "Expires" field is available in the HTTP header to define the expiration time, just follow the configuration of "**cache settings expire**".

7.2 Reverse Proxy Cache Configuration

The Cache configuration commands are designed for the administrators to set vital parameters as to what cacheable elements will be housed in the temporary memory of the FortiBalancer appliance. By caching certain elements, the appliance will be able to deliver commonly requested information more expediently without requesting the server frequently, thus reducing the total-download time and server load, and improving overall network performance.

7.2.1 Configuration Guidelines

Table 7-2 General Settings of Reverse Proxy Cache

Operation	Command
Enable cache	**cache {on|off}** *<virtual_service>*
View cache status	**show cache status**
Configure global cache expire time	**cache settings expire** *{hh:mm:ss|seconds}*
Configure the maximum size for a cache object	**cache settings objectsize** *<size>*
View cache basic settings	**show cache settings**
View cache statistics	**show statistics cache** *[virtual_service]*
Clear cache statistics	**clear statistics cache** *[virtual_service|all]*
View contents of cache objects	**show cache content** *<host_name> <url_regex>*
Remove all cache cache objects by force	**clear cache content**
Enable cache filter	**cache filter {on|off}**
Configure cache filter rule	**cache filter rule** *<host_name> <url> {cache|urlquery|ttl}*
View cache filter configuration	**show cache filter status**
View all cache filters about the specified host name	**show cache filter hostname** *<host_name>*

Operation	Command
View all cache filter rules	**show cache filter all**
View the cache filter rules matching the specified host name and URL	**cache filter match** *<host_name> <url_regex>*
Remove specified cache filter rules	**no cache filter rule** *<host_name> <url>*
Clear cache filter rules matched with the specified host	**clear cache filter hostname** *<host_name>*
Clear all cache filter rules	**clear cache filter all**
View cache filter statistics	**show statistics cachefilter** *<host_name> <url_regex>*
Clear cache filter statistics	**clear statistics cachefilter** *[host_name|all]*

7.2.2 Configuration Example via CLI

The Cache function for each virtual service works independently. By default, the Cache function is turned off. When Cache is turned off, no objects are stored in cache and all requests will go to the backend servers through the server load balancing mechanism.

➢ Step 1 Enable the cache function

To use cache, we need to first enable the Cache function for the specified virtual service.

In this example, we enable the Cache function for the virtual service “virtual_MOSS”.

```
FortiBalancer(config)#cache on virtual_MOSS
```

The current status of cache can be viewed by using the “**show cache status**” command.

```
FortiBalancer(config)#show cache status
reverse proxy cache:                    enable
per-vs status “virtual_MOSS”:           enable
```

➢ Step 2 Configure basic cache settings

We start to define basic cache rules for FortiBalancer appliance to follow. The settings that can be configured include:

- The expiration time of the objects in Cache,
- The maximum size of an object in Cache

The current Cache settings can be viewed by using the “**show cache settings**” command.

```
FortiBalancer#show cache settings
Cache Configuration:
              Cache Default Expiration:          82800 seconds
              Maximum Cacheable Object Size:     5120 KB
```

The above cache settings are the default values, which are the optimal values. If your application has some special requirements, you can make the above cache settings as your needs determine.

To set the global cache expiration time, we can use the “**cache settings expire**” command. The default value is 82800 seconds (23 hours). The global default expiration time will be used as the expiration time for an object in cache only if it is impossible to calculate the expiration time using the Expiration Model specified in Section 13.2 of RFC 2616.

```
FortiBalancer(config)#cache settings expire “43200”
```

To set the maximum size of an object in cache, the “**cache settings objectsize**” command should be used. The command takes the size in kilobytes. The default value is 5120 KB. If the size of an object being sent to the client is greater than the configured maximum object size, the object will not be cached even if it is otherwise cacheable.

```
FortiBalancer(config)#cache settings objectsize 1000
```

Now we use using the “**show cache settings**” command to view current cache settings:

```
FortiBalancer(config)#show cache settings
Cache Configuration:
                Cache Default Expiration:              43200 seconds
                Maximum Cacheable Object Size:    1000 KB
```

- Step 3 Configure cache filter

First, enable the cache filter function by using the command “**cache filter {on|off}** *<virtual_service>*”. By default, the cache filter function is disabled.

```
FortiBalancer(config)#cache filter on
```

Then, define cache filter rules by using the command “**cache filter rule** *<host_name> <url> {cache|urlquery|ttl}*”. Cache filter rules conveniently controls whether to cache an object and how long to cache it.

In our example, cache all “.jpg” objects from the host “www.xyz.com” and set the TTL to be 200,000 seconds:

```
FortiBalancer(config)#cache filter rule www.xyz.com ".*\.jpg" "cache=yes" "urlquery=yes"
ttl=200000
```

To view all cache filter rules we have configured.We can execute the command “**show cache filter all**”.

```
FortiBalancer(config)#show cache filter all
cache filter rule "www.xyz.com" "./*.jpg " "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.xyz.com" ".*\.bmp" "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.xyz.com" ".*\.gif" "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.test.com" "example" "cache=yes" "urlquery=yes" "ttl=150000"
cache filter rule "www.test.com" ".*\.jpg" "cache=yes" "urlquery=yes" "ttl=200000"
```

- Step4 Show cache statistics

Once you’ve configured your cache functions, the OS will allow you to view the running status of the appliance as it pertains to the caching requirements you’ve configured.

```
FortiBalancer(config)#show statis cache
Reverse Proxy Cache Global Statistics:
Basic Statistics:
      Requests received:                                    3601254
      Requests with GET method:                             3601254
      Requests with HEAD method:                              0
      Requests with PURGE method:                             0
      Requests with POST method:                             0
      Number of open client connections:                  115
      Number of open server connections:                  115
```

```
        Requests redirected to HTTPS:                          0
        Requests redirected based on regex match:              0
        Requests forwarded with rewritten url:                 0
        Locations rewritten to HTTPS:                          0
        Locations rewritten based on regex match:              0
        Cache skip, cache off:                                 3601254
        Cache hit, reply using cache:                          0
        Cache hit, reply with "Not Modified":                  0
        Cache hit, reply with "Precondition Failed":           0
        Cache hit, revalidate:                                 0
        Cache miss, noncacheable requests:                     3601254
        Cache miss, create new entry:                          0
        Cache miss, create new entry, resp noncacheable: 0
        Hit ratio:                                             0.00%

(Notice: the real server's time should be in sync with this machine.
                  Otherwise, the time difference could expire the cachable objects
                  resulting in low cache hit ratio.)

Advanced Statistics:
        Number of cache objects:                                         0
        Number of cache frames:                                          0
        Successful cache probes:                                         0

    Why were certain requests sent to the server?
    a) We had to revalidate the cached object due to:
            Request with "no-cache":                                     0
            Requset with "maxage=0":                                     0
            Cached object had "no-cache":                                0
            Cache object expired:                                        0

    b) We had to bypass cache for some requests because:
            Cache was filling when request was made:                 0
            Revalidation failed due to IMS mismatched:               0
            Client has newer copy, cannot send from cache:           0
            Object in cache is chunked, cannot give to 1.0 client: 0
            Network memory utilization was too high:                 0

c) Request cannnot be served from cache because:
            Cache filter denied caching:                             0
            Requests with "no-store":                                0
            Requests with "authorization":                           0
            Requests with cookies:                                   0
            Requests with range:                                     0
            Requests non GET, non HEAD:                              0
            Requests URL too long:                                   0
            Requests host too long:                                  0

    d) Error occured while doing cache lookup
            Network memory shortage when cache hit (200, 304):       0
            Cache was not accessible:                                0
            Fail to send cache lookup to cache:                      0
            Fail to find url and host:                               0
            Fail to parse cache specific http request headers:       0
            Fail to create a new cache object:                       0
            Noncacheble requests due to other errors:                3601254
```

```
Why were certain responses not stored in cache?
a) HTTP directive in response told us not to cache
        HTTP response code not 200, 300 or 301:                    0
        Response had a "no-store":                                  0
        Response had a "private":                                   0
        Response had a "set-cookie":                                0
        Response had a "vary":                                        0

b) The response did not meet our guidelines for cacheability
        Response noncacheable too big:                              0

 c) Error occured when trying to cache response
        Cache storage limit exceeded based on header data:         0
        Cache storage limit exceeded based on payload:             0
        Network memory shortage when storing response body:       0
        Cache object was deleted before response arrived:          0
        Fail to parse cache specific http response headers:      0
        Fail to store response headers in cache:                   0
        Fail to store response body in cache:                        0
        Cache object was aborted due to connection reset:          0
        Noncacheble responses due to other errors:                   0
```

Chapter 8 HTTP Content Rewrite

8.1 Overview

The HTTP Content Rewrite feature allows end users to visit the HTTP contents on the Web servers behind the FortiBalancer appliance. This feature aims to reduce network latency and improve user experience.

This chapter will cover the theories and configurations of the HTTP Content Rewrite feature.

8.2 Understanding HTTP Content Rewrite

8.2.1 How HTTP Content Rewrite Works

When a company places the FortiBalancer appliance in front of the application servers, users can access the applications through FortiBalancer. However, the Web pages that the applications generate contain links with private IP addresses or internal domain names. So if the user fetches the pages with those internal links through FortiBalancer, the browser will not be able to retrieve the images and other objects on the HTML page since the HTML links pointing to them do not point back to the FortiBalancer appliance.

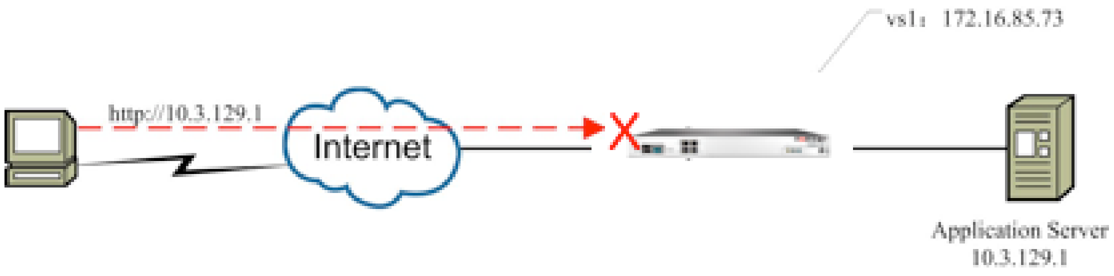


Figure 8-1 Users Failed to Access Internal Web Pages

The solution to the above problems is to rewrite contents of the HTML pages before it is sent to the client. To be more precise, the contents of every HTML page will be processed and all the HTML links will be found and rewritten so that the end user can access the pages via a browser. In addition, the rewritten pages can be cached by the FortiBalancer appliance so that FortiBalancer does not need to rewrite the same pages over and over again.

Upon receiving the responses from the real server, the FortiBalancer appliance will communicate with the uproxy module to read the response data that needs to be rewritten, and send these data to the rewrite module for rewriting. The IP addresses and domain names in the files that need rewriting according to configured rules will be rewritten. Finally the FortiBalancer appliance responds the rewritten data to the end user, and further caches the rewritten data.

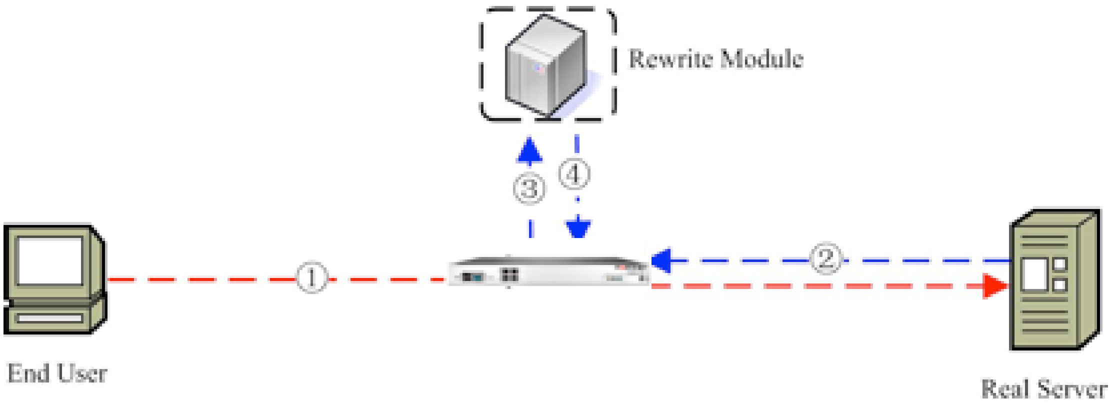


Figure 8-2 HTTP Content Rewrite Working Mechanism

1. The end user sends an HTTP request to the real server via the FortiBalancer appliance.
2. The response data coming from the real server reaches the FortiBalancer appliance.
3. The FortiBalancer sends the data to the rewrite module.
4. The rewrite module reads the data that needs rewriting, and rewrites the data.
5. The FortiBalancer appliance sends the rewritten data to the end user, and caches the data.

8.2.2 Advantages of HTTP Content Rewrite

Improve the user experience

The HTTP Content Rewrite help external end users from visiting the internal real server hidden behind the FortiBalancer appliance. It rewrites the Web page file into the valid format to end users. The FortiBalancer appliance also supports to rewrite the multi-byte character files such as Chinese and Japanese.

Reduce the effect to the performance

Comparing with the URL rewrite method, the HTTP Content Rewrite feature causes less communication with the real server to reduce the effect to the performance.

Decrease the response time

In addition, the rewritten pages can be cached by the FortiBalancer appliance so that FortiBalancer does not need to rewrite the same pages over and over again. When the client requests the real server for the same Web page, the FortiBalancer appliance will respond the client with the cached data to decrease the response time.

Easy to maintain

The HTTP Content Rewrite feature helps enterprise decrease the cost of the human resource. The administrator needs less monitoring because the HTTP Content Rewrite feature rewrites the Web page file by the rewrite driver and the rewrite module automatically.

8.2.3 Working Principles of HTTP Content Rewrite

The following introduces the working principles of HTTP Content Rewrite:

- **Global or per virtual service content rewrite**

The administrator can enable/disable the HTTP content rewrite globally or per virtual service.

Note:

1. By default, the HTTP content rewrite function is disabled globally, while enabled per virtual service.

2. Only with the global HTTP content rewrite enabled, will the per virtual service HTTP content rewrite enabling and configurations take effect.

- **Define the global content rewrite rule**

The HTTP content rewrite function allows administrators to define global rewrite rules to rewrite the IP addresses, domain names or other strings in the Web page files into new strings as pre-defined.

Two kinds of rewrite rules are supported:

1. ProxyHTMLURLMap

Only rewrite the URLs inside the HTML tags. The other strings will not be rewritten. This kind of rewrite rules can only be applied to rewriting of HTML and XHTML files.

For example:

The source Web page file:

```
<p><a href="10.3.129.1">10.3.129.1</a></p>
<p><a href="http://10.3.129.1/">http://10.3.129.1</a></p>
<p><a href="https://10.3.129.1/">https://10.3.129.1</a></p>
```

The rewritten Web page file:

```
<p><a href="172.16.85.74">10.3.129.1</a></p>
<p><a href="http://172.16.85.74/">http://10.3.129.1</a></p>
<p><a href="https://172.16.85.74/">https://10.3.129.1</a></p>
```

The IP address “10.3.129.1” in the URLs inside the HTML tags has been rewritten into “172.16.85.74”, while others remain unchanged.

2. Substitute

Rewrite URLs inside and outside the HTML tags.

For example:

The source Web page file:

```
<p><a href="10.3.129.1">10.3.129.1</a></p>
<p><a href="http://10.3.129.1/">http://10.3.129.1</a></p>
<p><a href="https://10.3.129.1/">https://10.3.129.1</a></p>
```

The rewritten Web page file:

```
<p><a href="172.16.85.74">172.16.85.74</a></p>
<p><a href="http://172.16.85.74/">http://172.16.85.74</a></p>
<p><a href="https://172.16.85.74/">https://172.16.85.74</a></p>
```

All the “10.3.129.1” strings have been rewritten into “172.16.85.74”.

Note:

1. The configuration strings of the parameter “rule” must be framed in double quotes.

2. The configuration strings of “ProxyHTMLURLMap” and “Substitute” are strictly case-sensitive.

3. When both “ProxyHTMLURLMap” and “Substitute” rules are configured, the “ProxyHTMLURLMap” rules will be applied first, and then the “Substitute” rules. If the “ProxyHTMLURLMap” and “Substitute” rules have been configured to map to the same regex, the “Substitute” rules will overwrite the “ProxyHTMLURLMap” rules.

4. To change the rewrite rules, the currently running rewrite operations will fail, and FortiBalancer will reset the related connections. Therefore, it is suggested not change the rewrite rules while the FortiBalancer appliance is processing network traffic.

5. If the HTTP content rewrite function is enabled and content rewrite rules are configured, the length of each line in responses cannot be greater than 1MB; otherwise, the FortiBalancer appliance will send a RST packet to the client to abort the TCP connection.

- **Specify the type of the files to be rewritten**

The HTTP Content Rewrite function allows administrators to specify the type of the files to be rewritten. The following are the supported file types:

- text/html
- text/plain

- text/richtext
- text/xml
- application/xml
- application/xhtml+xml
- text/css
- text/javascript
- application/javascript

By default, only the files in “text/html” type are allowed to be rewritten.

- **Define the URL regex for per virtual service HTTP content rewrite**

The administrator can define the URL regex to permit or deny rewriting of the files that match the URL regex per virtual service.

To specify the URL regex, the administrator should first define a URL list, and then add URL regexes into the URL list. The URL regex can be defined as the extension name, the file content or a part of the file name. Then the administrator need to associate the URL list with a virtual service through a permit/deny rule. After all these are done, the files that match any URL regex in the URL list will be rewritten according to the associated permit/deny rules.

Multiple URL regexes can be added into a URL list, in “OR” relationship. That is to say, the permit/deny rule will work as long as any of the extension name, file name or file contents matches the URL regex.

- **Define the HTTP response status code**

The HTTP Content Rewrite function also supports rewriting the Web page files that contain specific HTTP response status code. The “200” HTTP response status code is supported by default. That is to say, the FortiBalancer appliance will only rewrite the Web page files with the “200” HTTP response status code by default.

8.3 HTTP Content Rewrite Configuration

8.3.1 Configuration Guidelines

Before you start to configure HTTP Content Rewrite, please take a moment to familiarize yourself with the network architecture for HTTP Content Rewrite configuration.

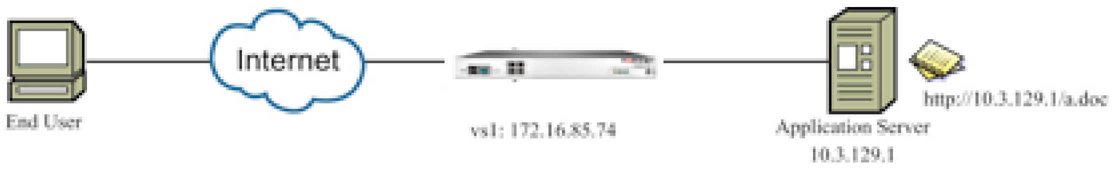


Figure 8-3 HTTP Content Rewrite Topology

As shown above, the real server is hidden behind the FortiBalancer appliance, and the end user can only access the IP address “172.16.85.74” of the virtual service “vs1”. If the end user wants to visit the Web resource “a.xml” on the real server “10.3.129.1”, the source URL address “http://10.3.129/1/a.xml” should be rewritten into “http://172.16.85.74/a.xml”.

Table 8-1 General Settings of HTTP Content Rewrite

Operation	Command
Enable HTTP Content Rewrite	**http rewrite body {on|off}** *[virtual_name]*

Operation	Command
Configure HTTP Content Rewrite rules	**http rewrite body rule** *<rule> [flags]*
Configure the file type to be rewritten	**http rewrite body mimetype** *<mime_type>*
Add a URL regex into a URL list	**http rewrite body url list** *<url_list> <url_regex>*
Configure to permit rewriting the string that matches the URL regex in a URL list	**http rewrite body url permit** *<virtual_service> <url_list>*
Configure to deny rewriting the string that matches the URL regex in a URL list	**http rewrite body url deny** *<virtual_service> <url_list>*
Configure to rewrite the files that contain specific HTTP response status code	**http rewrite body statuscode** *<response_code>*

8.3.2 Configuration Example via CLI

➢ Step 1 Configure the virtual service

```
FortiBalancer(config)#slb virtual http "vs1" 172.16.85.74 80 arp 0
```

➢ Step 2 Enable the HTTP Content Rewrite function

```
FortiBalancer(config)#http rewrite body on
```

➢ Step 3 View the status (enable/disable) of HTTP Content Rewrite

```
FortiBalancer(config)#show http rewrite body status
http  body rewrite:                enable
per-vs  status "vs1":                enable
```

➢ Step 4 Configure an HTTP Content Rewrite rule to rewrite the IP address “10.3.129.1” into “172.16.85.74”

```
FortiBalancer(config)#http rewrite body rule “ProxyHTMLURLMap 10.3.129.1
172.16.85.74” -R
```

➢ Step 5 View the defined HTTP Content Rewrite rules

```
FortiBalancer(config)#show http rewrite body rule
http rewrite body rule “ProxyHTMLURLMap 10.3.129.1 172.16.85.74” “-R”
```

➢ Step 6 Configure the type of files to be rewritten

```
FortiBalancer(config)#http rewrite body mimetype xml
```

Chapter 9 DNS Cache

9.1 Overview

The DNS SLB mechanism used by FortiBalancer appliance supports DNS cache feature. Upon receiving any type "A" or "AAAA" DNS responses, which are mapping of host names to IP addresses, FortiBalancer will save them in SLB DNS cache. Then, when the FortiBalancer appliance receives any DNS requests for the cached "A" or "AAAA" records from clients, the appliance will directly send back the "A" or "AAAA" responses to the clients. If there is no records in cache that hit the requests, the FortiBalancer appliance will communicate with the remote DNS server(s) directly, and then save the server responses in cache for responding to the coming requests.

9.2 DNS Cache Configuration

9.2.1 Configuration Guidelines

Table 9-1 General Settings of DNS Cache

Operation	Command
Define related SLB component	**slb real dns** *<real_name> <ip> <port> [max_conn] [dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down] [timeout]* **slb virtual dns** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb policy static** *<virtual_name> <real_name>*
Enable DNS cache	**dns cache {on|off}**
Configure the DNS cache expiration time	**dns cache expire** *<min_seconds> <max_seconds>*
Establish hosts for the DNS cache	**dns cache host** *<host_name> <ip>*

9.2.2 Configuration Example via CLI

➢ Step 1 Configure necessary SLB component

Since DNS cache is interdependent with SLB configuration strategies, please refer to the chapter Server Load Balancing (SLB). Below is a configuration example for DNS cache deployment. First, the SLB component needs to be established.

```
FortiBalancer(config)#slb real dns "RS_DNS_1" 10.1.1.10 53 1000 icmp 1 1 20
FortiBalancer(config)#slb virtual dns "VS_DNS_1" 10.1.61.100 53
FortiBalancer(config)#slb policy static "VS_DNS_1" "RS_DNS_1"
```

The commands above set up an SLB configuration where the real service is named and bound to a real IP address/port pair. This real service is then, in turn, bound to the configured virtual service via the static policy. These commands are covered in depth in the CLI Reference.

➢ Step 2 Enable DNS cache

To enable DNS cache, the "**dns cache {on|off}**" command should be used. The DNS cache is disabled by default.

```
FortiBalancer(config)#dns cache on
```

➢ Step 3 Configure the DNS cache expiration time

```
FortiBalancer(config)#dns cache expire 1 36000
```

- Step 4 Establish hosts for the DNS cache

```
FortiBalancer(config)#dns cache host "sting" 10.1.61.200
FortiBalancer(config)#dns cache host "gunrose" 10.1.61.100
FortiBalancer(config)#dns cache host "roxxette" 10.1.61.2
FortiBalancer(config)#dns cache host "queens" 10.1.61.47
```

Chapter 10 HTTP Compression

10.1 Overview

The FortiBalancer appliance supports in-line compression of HTTP objects. By employing this licensed feature, administrators may maximize throughput to the desired site while end-users will experience quicker download speeds. This chapter describes the configuration of HTTP Compression capabilities which are part of the FortiBalancer platform. Configuration of HTTP Compression functionality can be divided into two main parts. The first part is the basic configuration and the second part is dedicated to advanced configuration.

10.2 Understanding HTTP Compression

HTTP compression, otherwise known as content encoding, is a publicly defined way to compress textual contents transferred from Web servers to browsers. HTTP compression uses public domain compression algorithms to compress XHTML, JavaScript, CSS, and other text files at the server.

By default, the following MIME types can be compressed by the FortiBalancer appliance for all the browsers:

- Text (text/plain)
- HTML (text/HTML)
- XML (text/XML)

The following MIME types are able to be compressed by the FortiBalancer appliance for certain browsers via "**http compression policy useragen**t" command:

- Java Scripts (application/x-javascript)
- Cascade Style Sheets (text/css, application/x-pointplus)
- PDF documents (application/pdf)
- PPT documents (application/powerpoint)
- XLS documents (application/MSExcel)
- DOC (application/MSWord)

Now, if Administrators do not want to compress some textual contents from Web servers to browsers, they are able to configure a url-exclude HTTP compression rule for the client request via the the "**http compression policy urlexclude**" command. If the URL of the client request matches the rule, the textual content from the Web servers to the browsers will not be compressed even if HTTP compression is on.

Not all files are suitable for compression. For obvious reasons, files that are already compressed, such as JPEGs, GIFs, PNGs, movies, and bundled contents (e.g., Zip, Gzip, and bzip2 files) are not going to be compressed appreciably further with a simple HTTP compression filter. Therefore, you are not going to get much benefit from compressing these files or a site that relies heavily on them.

Note: For the data files in very small size, the size of the compressed data may be larger than the original data size.

10.3 HTTP Compression Configuration

This section covers enabling HTTP Data Compression functionality with default settings and configuring the advanced HTTP compression.

10.3.1 Configuration Guideline

Table 10-1 General Settings of HTTP Compression

Operation	Command
Enable HTTP data compression	**http compression{on|off}** *[virtual_name]*
View HTTP data compression status	**show http compression settings**
Set advanced HTTP compression	**http compression policy useragent** *<user_agent_string> {js|css|pdf|ppt|xls|doc}* **http compression advanced useragent on**
Set the url-exclude compression rule	**http compression policy urlexclude** *<vhost> <wildcard_expression>*

10.3.2 Configuration Example via CLI

➢ Step 1 Enable HTTP Compression

```
FortiBalancer(config)#http compression on
```

This command enables the HTTP Compression functionality with default settings. By default, the FortiBalancer appliance compresses the following MIME types for all the browsers:

- Text (text/plain)
- HTML (text/HTML)
- XML (text/XML)

➢ Step 2 Check the status of HTTP Compression

```
FortiBalancer(config)#show http compression settings
```

➢ Step 3 Configure advanced HTTP Compression

If you want to enable the compression of Java Script for Microsoft IE 5.5, you can enable it by specifying the following parameters:

```
FortiBalancer(config)#http compression policy useragent “MSIE 5.5” JS
```

There are other types of Web contents that are compressible such as:

- Java Scripts (application/x-javascript)
- Cascade Style Sheets (text/css, application/x-pointplus)
- PDF documents (application/pdf)
- PPT documents (application/powerpoint)
- XLS documents (application/MSExcel)
- DOC (application/MSWord)

Not all browsers are able to process the compressed forms of these MIME types. Support for the above MIME types requires detection of appropriate user agents that can deal with the compressed forms of these types, and then apply the compression functionality to only those user agents. To process variations in the handling of these MIME types by browsers, the FortiBalancer appliance provides the administrator with the capability to turn ON compression based on specific user agent and MIME types.

Note: TEXT, XML and HTML of HTTP compression are default values, so they do not need to be configured by the command “http compression policy useragent”.

Fortinet provides a tested list of browsers that can handle the compressed form of additional MIME types. The FortiBalancer appliance provides administrators with a way to enable the compression of additional MIME types for a best-known-working-set of browsers by using the following command:

```
FortiBalancer(config)#http compression advanced useragent on
```

It activates the compression of Java Script and CSS types for IE 6, IE 7, IE 8 and Mozilla 5.0 browsers.

- Step 4 Configure url-exclude HTTP Compression rule

```
FortiBalancer(config)#http compression policy urlexclude “v1” “/abc”
```

If the URL of a client request to the virtual service “v1” matches the string “/abc”, the textual contents in the response will not be compressed even if HTTP compression is turned on.

```
FortiBalancer(config)#http compression policy urlexclude “v1” “^/def”
```

If the URL of a client request to the virtual service “v1” starts from the string “/def”, the textual contents in the response will not be compressed even if HTTP compression is turned on.

```
FortiBalancer(config)#http compression policy urlexclude “v1” “ghi.txt$”
```

If the URL of a client request to the virtual service “v1” ends with the string “ghi.txt”, the textual contents in the response will not be compressed even if HTTP compression is turned on.

```
FortiBalancer(config)#http compression policy urlexclude “v1” “abc*def”
```

If the URL of a client request to the virtual service “v1” matches the string “abc*def”, the textual contents in the response will not be compressed even if HTTP compression is turned on.

Chapter 11 Secure Sockets Layer (SSL)

11.1 Overview

Now that the basic SLB and Caching are setup on the FortiBalancer appliance, we can set up the SSL (Secure Sockets Layer) acceleration functionality to provide secure transactions with your clients. The SSL Accelerator works by decrypting the secure traffic and passing the unencrypted traffic to the original server. In an alternative mode the SSL accelerator can be used to decrypt the secure traffic, apply traffic management (SLB, caching, etc.) processing on decrypted traffic and then encrypt it back before passing it to SSL enabled origin server.

11.2 Understanding SSL

The main role of SSL is to provide security for Web traffic. Security includes confidentiality, message integrity, and authentication. SSL achieves these elements of security through the use of cryptography, digital signatures, and certificates.

11.2.1 Cryptography

SSL protects confidential information through the use of cryptography. Sensitive data is encrypted across public networks to achieve a level of confidentiality. There are two types of data encryption: secret key cryptography and public key cryptography.

Secret key cryptography – known as symmetric cryptography. It uses the same key for encryption and decryption. An example of symmetric cryptography is a decoder ring. Alice has a ring and Bob has the same ring. Alice can encode messages to Bob using her ring as the cipher. Bob can then decode the sent message using his ring. In cryptography, the "decoder ring" is considered a preshared key. The key is agreed upon by both sides and can remain static. Both sides must know each other already and have agreed upon what key to use for the encryption and decryption of messages.

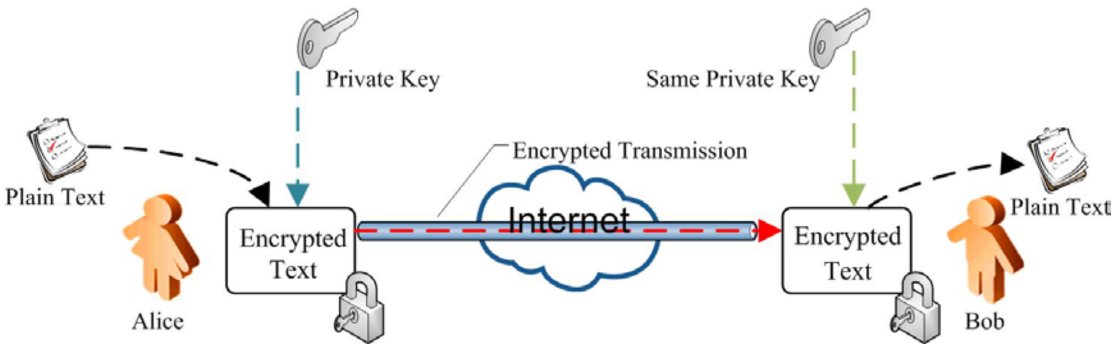


Figure 11-1 Secret Key Encryption/Decryption

Public key cryptography – It uses one key for encryption of data, and then a separate key for decryption. It is more favorable than secret key cryptography because even if the encryption key is learned in one direction, the third party still needs to know the other key in order to decrypt the message in the other direction.

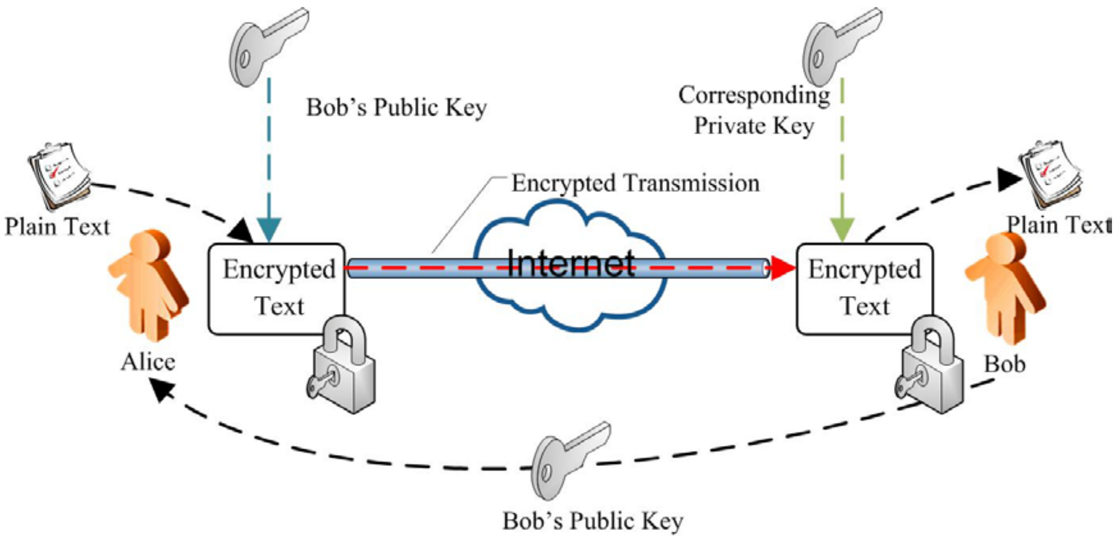


Figure 11-2 Public Key Encryption/Decryption

11.2.2 Digital Signatures

To ensure the integrity of messages transmitted via the Internet, each message exchanged via SSL has a digital signature attached to it. A digital signature is a hashed message digest which is encrypted by hash algorithm and contains public key information. The message digest is generated based on the checksum results on the message. The message digest cannot be reversed by algorithm. Thus, both parties will compute the message digest separately and then compare the hashed results. If their computing results match, it means the message has not been altered during transission on Internet, which minimizes the chances of information leakage.

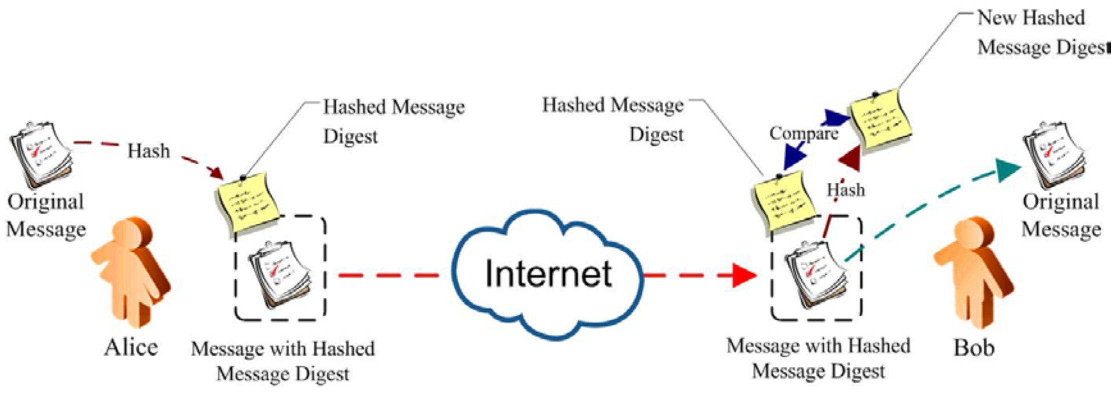


Figure 11-3 Digital Signatures

11.2.3 Certificates

Certificates contain information identifying the user/device. They are digital documents that will attest to the binding of a public key to an individual or other entity. They allow verification of the claim that a specific public key does, in fact, belong to the specified entity. Certificates help prevent someone from impersonating the server with a false key. SSL uses X.509 standard certificates to validate identities. X.509 standard certificates contain information about the entity, including public key and name. A certificate authority then validates this certificate.

Table 11-1 X.509 Certificate

Certificate Information
Algorithm Identifier
Serial Number
Version

Certificate Information
Issuer
Period of Validity
Subject
Subject's Public Key
Issue Unique ID
Subject Unique ID
Extensions
Signature

11.2.3.1 Client Certificate Parse

A backend real service needs information of a client certificate before processing the client requests. But the backend server itself cannot recognize and analyze a complete SSL certificate. FortiBalancer appliance will parse the client certificate into many fields and then transfer them to the backend server through HTTP URL request parameters or HTTP headers.

The FortiBalancer appliance supports using the certificate parser (Fortinet patent) to verify the client certificate in a fast way.

11.3 SSL Acceleration Configuration

11.3.1 Configuration Guidelines

Before we get started, let's explain the terminologies used extensively throughout this chapter.

Virtual Host: An SSL host associated with an SLB virtual. An SSL virtual host acts as an SSL server and is used to communicate by using SSL between browser and FortiBalancer appliance.

Real Host: An SSL host associated with an SLB real. An SSL real host acts as an SSL client and is used to communicate by using SSL between FortiBalancer appliance and backend origin server.

Origin Server: A backend server that will accept clear-text or encrypted requests.

Clear-text: Any traffic that is not encrypted.

Virtual Host Port: The port that SSL virtual host will listen on. Typically port 443 is used.

Key (private): A private key that is stored on the FortiBalancer appliance for PKI (Public Key Infrastructure) authentication purposes. FortiBalancer appliance supports keys up-to 2048 bits in size.

Certificate: This is used for authentication purpose and to help set up secure communications between the appliance and the browser.

Certificate Authority (CA): A certificate authority is an entity that will create a certificate from a CSR (Certificate Signing Request).

Trusted Certificate Authority: Current Web Browsers have a list of known CA's public keys that are used to verify certificates authenticity. If the browser cannot identify the CA it will inform the user as such. In a similar manner the FortiBalancer appliance also maintains a list of Trusted Certificate Authorities to verify certificates.

For our example environment, we have a domain name of "www.example.com". For our SSL purposes we will be using "www.example.com" as our SSL virtual host. This SSL virtual host is associated with an SLB virtual host using IP 10.10.0.10 and port 443.

SSL virtual Host: www.example.com

SLB virtual Host IP: 10.10.0.10

SLB virtual Host Port: 443

There are two methods for setting up SSL acceleration. The first method applies if you have never set up SSL, and you will need to walk through the whole process of setting up the SSL virtual host and generation of a CSR to send to the CA of your choice. The CA will send you a signed certificate that you will then import. The second method applies if you already have a key and certificate, and you can skip the CSR step and import your key and certificate. Let's go ahead and setup SSL as though we have never set it up.

Table 11-2 General Settings of SSL

Operation	Command
Create an SSL virtual host	**slb virtual https** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **ssl host** *{real|virtual} <host_name> <slb_service>*
Import certificate and key for SSL virtual host	**ssl csr** *<virtual_host_name> [key_length]* **ssl import certificate** *<host_name> [cert_index] [tftp_ip] [file_name]* **ssl import key** *<host_name> [cert_index] [tftp_ip] [file_name]* **ssl activate certificate** *<host_name> [cert_index]*
Create an SSL real host	**slb real https** *<real_name> <ip> [port] [max_conn] [https|tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb real tcps** *<real_name> <ip> <port> [max_conn] [tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **ssl host** *{real|virtual} <host_name> <slb_service>*
Advanced configuration for an SSL virtual host	**ssl stop** *<host_name>* **ssl settings ciphersuite** *<host_name> <cipher_string>* **ssl settings protocol** *<host_name> <version>* **ssl settings reuse** *< host_name>* **ssl settings clientauth** *<host_name>* **ssl settings certfilter** *<vhost_name> <filter1> [filter2]* **ssl import rootca** *[virtual_host_name] [tftp_ip] [filename]* **ssl settings crl offline** *<virtual_host_name > <crldp_name> <crldistribution_point> [time_interval] [delay_time]* **ssl settings crl online** *<virtual_host_name>* **ssl settings ocsp** *<virtual_host_name> <ocsp server>* **ssl settings minimum** *<virtual_host_name> <key_size> <url>* **ssl start** *<host_name>* **ssl import error** *<error_code> <url> [virtual_host_name]* **ssl load error** *<error_code> [virtual_host_name]*
Advanced configuration for an SSL real host	**ssl settings ciphersuite** *<host_name> <cipher_string>* **ssl settings protocol** *<host_name> <version>* **ssl settings reuse** *<host_name>* **ssl settings clientauth** *<host_name>* **ssl import rootca** *[virtual_host_name] [tftp_ip] [filename]* **ssl settings servername** *<real_host_name> <ssl_server_common_name>*

11.3.2 Configuration Example via CLI

11.3.2.1 Creating an SSL Virtual Host

To do this, first we will employ an SLB related command. This command will create the SLB virtual service. The second step is to use the "**ssl host virtual** *<host_name> <slb_service>*" command to define our SSL virtual host.

```
FortiBalancer(config)#slb virtual https virtual1https 10.10.0.10 443
FortiBalancer(config)#ssl host virtual www.example.com virtual1https
```

In the above example, please note that "virtual1https" is our newly created SLB virtual service. Now you may move on to importing your certificate.

11.3.2.2 Importing Certificate and Key for the Newly Created SSL Virtual Host

If you do not have a certificate and key pair, FortiBalancer appliance provides with you the facility to create a key pair and CSR for your newly configured SSL virtual host. The FortiBalancer appliance also creates a test certificate that can be used for either testing or evaluation purposes.

➢ Step 1 Use FortiBalancer appliance to create a CSR for the newly created SSL virtual host

The first step is to use the "**ssl csr** *<virtual_host_name> [key_length]*" command to generate a CSR to send to your CA. After this command is employed, the appliance will prompt you for additional information. (The information in **bold** typeface represents answer examples.)

```
FortiBalancer(config)#ssl csr www.example.com
We will now gather some required information about your ssl virtual host,
This information is encoded into your certificate
Two character country code for your organization (e.g. US): US
State or province []: CA
Location or local city []: San Jose
Organization Name: Example.com
Organizational Unit: Example.com
Organizational Unit []:
Organizational Unit []:
Do you want to use the virtual host name "www.example.com" as the Common Name
(recommended)?(Y/N): Y
Email address of administrator []: admin@example.com
Do you want the private key to be exportable [Yes/(No)]:
Enter passphrase for the private key:
Confirm passphrase for the private key:

-----BEGIN CERTIFICATE REQUEST-----
MIIB5TCANU4ANQAwgaQxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDQTERMA8GA1
UEBxMIU2FuIEpvc2UxFDASBgNVBAoTC0V4YW1wbGUuY29tMRQwEgYDVQQLEwtFeG
FtcGxlLmNvbTEnMCUGA1UEAxMec3NsLXRlc3QucHBiLmFycmF5bmV0d29ya3MubmV0M
SAwHgYJKoZIhvcNAQkBFhFhZG1pbkBleGFtcGxlLmNvbTCBnzANBgkqhkiG9w0BAQEFAA
OBjQAwgYkCgYEApk18ozLXGEpJS69BvtfNLcBEjoO82+QWRtH4CtIVJYCEOIAlQXQPWs
NNN2A74AOW2wkm3f7leSEf2fPI/U6ScHYm8dz2OT523XdDZ/yqmQNwRwTz3NC0sNtXKR
g9WD9fPMgr6grdBCEH2eVcRdDK8EIXIFrlhXmz+UTxA9y92gMANwEAAaAAMA0GCSqG
SIb3DQEBBAUAA4GBAJCZiGnJ3AHcpuapkjbr31Qr9+1eHl/V6TOesQS/gOlxbOug00T7HndIo
32dZ9vnyGZqNd4CVg9rfFfQWuk09XfDSXdvEFU9ZzedNEr1d5ujbQv8pCsrNIlkHDPDF4Hs2r
e1ZJeSDpnEEj1EFAFaEyW452C8v4uGjCe2nrgrksgN
-----END CERTIFICATE REQUEST-----
```

This command creates a CSR for the SSL virtual, which will be sent to a CA for signing. This CSR uses the public key from the public-private key pair of the SSL virtual host, which was generated at the time of creating this SSL virtual host.

Note: This command also creates a test certificate for the SSL virtual host. The test certificate generated by the "ssl csr" command should not be used for production systems, rather only for testing purposes. The OS will check the certificate chain for the SSL virtual host when starting the virtual host. A warning message, stating that the certificate chain is incomplete, will be printed on console for the test certificate.

If you would like to use the test certificate for testing/demonstration purposes, this is the point where you may start the SSL subsystem:

```
FortiBalancer(config)#ssl start www.example.com
```

The FortiBalancer appliance is configured to take full advantage of the SSL functionality within the OS. Administrators should be able to connect securely to the site by using a Web browser.

- Step 2 Forward CSR to a CA

To perform this task, simple cut from "-----BEGIN CERTIFICATE REQUEST----" line down to the "----END CERTIFICATE REQUEST-----". Your CA needs these lines in order to expedite your request for a certificate. This process can take up to two days depending on verification. You will typically get an email back that looks like:

```
-----BEGIN CERTIFICATE-----
MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0
NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQ
QDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2Z
WxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgw
MTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEw
NET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIu
MC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQ
ADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIX
gRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6Yin
DTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEB
QADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4
pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnE
aiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cRQLU=
-----END CERTIFICATE-----
```

Warning: It is imperative that you do not delete the SSL virtual host before you import the certificate you received from your CA. If you clear your SSL information you will have to send another CSR to your CA to get another certificate. Fortunately most CAs give you a 30day trial period to get another certificate if something goes wrong. If it is past the 30day mark you might have to pay for another certificate. Be very careful when manipulating any SSL configurations on the FortiBalancer appliance.

Once you have received the certificate you can import it into the SSL subsystem. To perform this task, simple cut from "-----BEGIN CERTIFICATE ----" line down to the "----END CERTIFICATE-----". It is important to follow the instructions as supplied by the appliance to terminate the import.

```
FortiBalancer(config)#ssl import certificate www.example.com 1
You may overwrite an existing certificate file, type "YES" without quotes to continue: YES
Enter the certificate file in PEM format, use "..." on a single line, without quotes to terminate
import
-----BEGIN CERTIFICATE-----
MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0
NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQ
QDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2Z
WxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgw
MTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEw
NET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIu
MC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQ
ADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIX
gRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6Yin
DTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEB
QADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4
pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnE
aiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cRQLU=
-----END CERTIFICATE-----
```

Also you can import a certificate file from a remote machine running the TFTP service. The file name defaults to <Host name>.crt. In our case, the file name is “www.example.com.crt”.

```
FortiBalancer(config)#ssl import certificate www.example.com 1 10.10.13.82
www.example.com.crt
You may overwrite an existing certificate file, type "YES" without quotes to continue: YES
```

After importing the certificate successfully, you will get a response from the CLI prompt. Then you can activate the certificate via the following command “**ssl activate certificate** *<host_name> [cert_index]*”. For example:

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

And then you can start up the SSL:

```
FortiBalancer(config)#ssl start www.example.com
```

Now we have a functioning SSL accelerated site. If this example is a real site configuration, you will be able to connect securely to the site by using your Web browser.

Note: The OS will check the certificate chain for the SSL virtual host when enabling the virtual host. A warning message, stating that the certificate chain is incomplete, will be printed on console for the certificate if any of the certificates for its root CA and intermediate CAs cannot be found in the host’s intermediate CA file or the global trusted CA file. These certificates can be imported by using the “ssl import rootca” and “ssl import interca <vhostname>” commands.

If you already have a key and certificate pair from a trusted certificate authority, you can easily import them into the FortiBalancer appliance. This can be done by using the “**ssl import key**” and “**ssl import certificate**” commands.

- Step 1 Use existing certificate and key for newly created SSL virtual host

```
FortiBalancer(config)#ssl import key www.example.com 1
You may overwrite an existing key file. This may require you to purchase a new certificate type
"YES" without quotes to continue: YES
```

After you execute this command the appliance will ask you to cut and paste your existing key directly into the CLI. Make absolutely certain to follow the instructions as put forth by the appliance.

Also you can import a key file from a remote machine running the TFTP service. The file name defaults to <Host name>.key. In our case, the file name is “www.example.com.key”.

```
FortiBalancer(config)#ssl import key www.example.com 1 10.10.13.82 www.example.com.key
You may overwrite an existing key file. This may require you to purchase a new certificate type
"YES" without quotes to continue: YES
```

Then we will proceed with importing the certificate:

```
FortiBalancer(config)#ssl import certificate www.example.com 1 www.example.com.crt
You may overwrite an existing certificate file, type "YES" without quotes to continue: YES
Enter the certificate file in PEM format, use "..." on a single line, without quotes to terminate
import.

-----BEGIN CERTIFICATE-----
MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0
NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQ
QDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2Z
WxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgw
```

MTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEw
NET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIu
MC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQ
ADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIX
gRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6Yin
DTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEB
QADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4
pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnE
aiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cRQLU=
-----END CERTIFICATE-----

Note: You must import the key and then import the certificate. The FortiBalancer appliance supposes that the key is imported first.

After importing the certificate successfully, you will get a response from the CLI prompt. Then you can activate the certificate via the command "**ssl activate certificate** <*host_name*> *[cert_index]*". For example:

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

Then we can start the SSL subsystem:

```
FortiBalancer(config)#ssl start www.example.com
```

Now the FortiBalancer appliance is configured to take full advantage of the SSL functionality within the OS. At this point, administrators should be able to connect securely to the site by using a Web browser.

The FortiBalancer appliance allows you to import PEM (Privacy Enhanced Mail) formatted certificate and key through a cut and paste method via the CLI or web UI. If you have a "Non-PEM" formatted certificate and key pair, you will need to import the certificate and key via TFTP. This is explained in the following section.

Import Certificate and Key from IIS and NS iPlanet Web Servers

IIS

If you are using the Microsoft IIS server, the FortiBalancer appliance will allow you to import the certificate from IIS versions 4 and 5 through TFTP mechanism. IIS stores the SSL key and certificate in the same file. This file is in .PFX format. You need to put this file onto a TFTP server in its TFTP root directory and rename it as <host_name>.crt. This file then can be imported into FortiBalancer appliance through the "**ssl import certificate**" command. This command takes TFTP server IP as an extra argument.

```
FortiBalancer(config)#ssl import certificate www.example.com 1 10.10.0.3
```

This command will download a file that is named <host_name>.crt. In our case it is "www.example.com.crt" from the TFTP server (10.10.0.3).

After importing the certificate successfully, you will get a response from the CLI prompt. Then you can activate the certificate via the command "**ssl activate certificate** <*host_name*> *[cert_index]*". For example:

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

Once the certificate and key import is successful through TFTP server, you need to start the SSL service with the "**ssl start**" command.

```
FortiBalancer(config)#ssl start www.example.com
```

Netscape/iPlanet

If you are using the Netscape or iPlanet servers, the FortiBalancer appliance will also allow you to import the certificate and key. The iPlanet server stores the key/cert pair in the directory /<serverroot>/alias/ where <serverroot> is the directory where the server is installed. In that directory there will be two files of the form <serverid-hostname>-key3.db and <serverid-hostname>-cert7.db. You will need to copy the first file to your TFTP server's root directory and name it the same as your virtual host with a .key extension. The cert will be the same, but with a .crt extension. These have to be exact, or the SSL subsystem will not load them correctly.

Now we can import the certificate and key.

```
FortiBalancer(config)#ssl import key www.example.com 1 10.10.0.3 www.example.com.key
```

This command imports the key from 10.10.0.3 with the file name “www.example.com.key”.

Note: You must first import the certificate and then import the key when importing an SSL cert/key pair from iPlanet.

```
FortiBalancer(config)#ssl import certificate www.example.com 1 10.10.0.3
www.example.com.crt
```

This command imports the certificate from 10.10.0.3 with the filename “www.example.com.crt”.

Once the key is imported, theOS will ask you for a password. This password is the one you use for the database password on the iPlanet server.

After importing the certificate successfully, you will get a response from the CLI prompt. Then you can activate the specific certificate via the command “**ssl activate certificate** *<host_name> [cert_index]*”. For example:

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

Then we can start the SSL subsystem:

```
FortiBalancer(config)#ssl start www.example.com
```

Now the FortiBalancer appliance is configured to take full advantage of the SSL functionality within the OS.

IMPORTANT: In this section we have created an SLB virtual service and configured SSL for it. This SLB virtual service is now ready to be used, and need to be linked with one or more SLB real services so that the SLB module can complete the SSL requests coming to this SLB virtual. To get information on “How to associate an SLB virtual with an SLB real”, please refer to the SLB configuration section.

11.3.2.3 Creating an SSL Real Host

The FortiBalancer appliance allows you to use the SSL subsystem to talk to SSL enabled real servers. This allows an encrypted transaction between the OS and the backend servers.

Configuration of SSL real host is very simple and can be explained as follows:

The first step in this procedure is to define the SLB real service. To do this first we will employ an SLB related command. This command will create the SLB real service. The second step is to use the “**ssl host**” command to define the SSL real host.

For the definition and meaning of each parameter supplied in this command, please refer to the SLB CLI section of CLI Reference.

```
FortiBalancer(config)#slb real https real1https 192.168.1.20 443 tcps
FortiBalancer(config)#ssl host real www.myreal.com real1https
```

In the above example, please note that “real1https” is our newly created SLB real service, which represents a backend server running on IP 192.168.1.20 and port 443 and is capable of handling SSL requests. As a final step, we can start the SSL subsystem:

```
FortiBalancer(config)#ssl start www.myreal.com
```

Now the FortiBalancer appliance is configured to take full advantage of the SSL functionality while communicating with the backend server.

IMPORTANT: In this section we have created an SLB real service and configured SSL for it. This SLB real service is now ready to be used, and need to be linked with an SLB virtual service so that the SLB module can direct the traffic to this SSL enabled SLB real service. To get information on “How to associate an SLB real with an SLB virtual” please refer to the SLB configuration section.

11.3.2.4 Advanced SSL Configuration for SSL Virtual Host

You can configure different SSL settings for your SSL virtual host.

➢ Step 1 Stop SSL virtual host

```
FortiBalancer(config)#ssl stop www.example.com
```

This will stop SSL virtual host and will allow you to change SSL settings for this virtual host.

➢ Step 2 Configure ciphersuites for the SSL virtual host

```
FortiBalancer(config)#ssl settings ciphersuite “www.example.com” “DES-CBC3-SHA”
```

The cipher suite settings allow you to define ciphers for this SSL virtual host. The following lists the cipher suites allowed for an SSL virtual host:

Cipher Suites	Bits	Protocols - Virtual Hosts		
		SSLv3.0	TLSv1.0	TLSv1.2
RC4-MD5	128	√	√	√
RC4-SHA	128	√	√	√
DES-CBC-SHA	64	√	√	x
DES-CBC3-SHA	192	√	√	√
AES128-SHA	128	√	√	√
AES256-SHA	256	√	√	√
AES128-SHA256	128	x	x	√
AES256-SHA256	256	x	x	√
EXP-RC4-MD5	40	√	x	x
EXP-DES-CBC-SHA	40	√	x	x

Note: In the preceding table, “√” indicates that the cipher suite is supported; “x” indicates that the cipher suite is not supported.

To enable multiple ciphers for a single SSL virtual host, you will need to specify the ciphers in the form of a colon (:) separated list. The following command enables all the ciphers for an SSL virtual host:

```
FortiBalancer(config)#ssl settings ciphersuite “www.example.com”
“RC4-MD5:RC4-SHA:DES-CBC3-SHA:DES-CBC-SHA:AES128-SHA:AES256-SHA:AES1
28-SHA256:AES256-SHA256:EXP-RC4-MD5:EXP-DES-CBC-SHA”
```

➢ Step 3 Configure protocol version for SSL virtual host

```
FortiBalancer(config)#ssl settings protocol "www.example.com" "SSLv3:TLSv1:TLSv12"
```

The FortiBalancer appliance supports the protocols Secure Sockets Layer version 3 (SSLv3), Transport Layer Security Protocol version 1.0 (TLSv1), and Transport Layer Security Protocol version 1.2 (TLSv1.2). You can use one, two or all of these protocols for the SSL virtual host settings.

Note: Parameter value "TLSv12" stands for the TLSv1.2 protocol.

➢ Step 4 Configure session reuse for SSL virtual host

```
FortiBalancer(config)#ssl settings reuse "www.example.com"
```

This allows you to enable SSL session reuse for an SSL virtual host. This feature is enabled by default.

➢ Step 5 Configure client authentication for SSL virtual host

The FortiBalancer appliance supports the SSL based client authentication. You can enable client authentication for an SSL virtual host. If enabled, the FortiBalancer appliance will require each client to present an SSL certificate for authorization, before the client can access the SSL virtual host.

```
FortiBalancer(config)#ssl settings clientauth "www.example.com"
```

IMPORTANT: If you enable SSL client authentication for an SSL virtual host, you must provide a trusted CA certificate. This will be used by the FortiBalancer appliance to verify client certificates.

```
FortiBalancer(config)#ssl import rootca "www.example.com"
```

This command will prompt you to cut and paste the trusted authority certificate in PEM format. You may configure multiple trusted authorities for one SSL virtual host.

Furthermore, the SSL virtual host will check the client certificate based on the configured certificate filters (by using the command "**ssl settings certfilter**"). If the client certificate fails the certificate verification, the SSL host will reject the client's access. At most three pieces of "certfilter" configuration (by using the "ssl settings certfilter" command) can be configured for an SSL virtual host. The logical relationship among the three pieces of "certfilter" configuration is "OR". If the client certificate does not match any piece of "certfilter" configuration, the SSL virtual host will reject the client's access.

The filters can be configured with any of the supported RDNs on the FortiBalancer appliances.

Table 11-3 Supported RDN on FortiBalancer

RDN	Standard Name
C	Country Name
ST	State or Province Name
L	Locality Name
O	Organization Name
OU	Organizational Unit Name
CN	Common Name
SN	Serial Number
dnQualifier	DN Qualifier
Pseudonym	Pseudonym
Title	Title
GQ	Generation Qualifier
Initials	Initials

RDN	Standard Name
Name	Name
givenName	Given Name
Surname	Surname
DC	Domain Component
emailAddress	Email Address
{OID expression}	OID information, for example: 1.2.3.4

For example:

```
FortiBalancer(config)#ssl settings certfilter vhost
“subject:/C=US/O=Fortinet/OU=QA/emailAddress=support@fortinet.com”
“issuer:/C=US/”
```

In this example, client certificates can pass the certificate verification only when the following conditions are both met:

- In the “subject” field, “C” is “US”, “O” is “Fortinet”, “OU” is “QA” and “emailAddress” is “support@fortinet.com”.
- In the “issuer” field, “C” is “US”.

Otherwise, the client will fail the authentication.

Two kinds of client authentication modes are supported: mandatory and non-mandatory. Client authentication mode defaults to mandatory. In non-mandatory client authentication mode, when the server sends a certificate request to the client, if the client has no matched certificate or cancels the authentication by clicking the Cancel button, the server will permit the client to access limited network resources instead of dropping the SSL connection. However, all the networks resources which can be published to non-authenticated clients need to be defined by using the “**http acl url**” command.

➢ Step 6 Configure client certificate parsing for SSL virtual host

You also can define the certificate parse method for the SSL virtual host.

```
FortiBalancer(config)#ssl settings cerparse www.example.com
FortiBalancer(config)#ssl settings verifymethod www.example.com fast
```

➢ Step 7 Configure CRL for SSL virtual host

FortiBalancer supports the CRL (Certificate Revocation List) functionality. You can configure the FortiBalancer appliance to fetch the CRL file periodically from a CRL Distribution Point (CDP) by using HTTP or FTP.

For our example, let’s consider a case when you have put your CRL file (Fortinet.crl) on an HTTP Web server (www.crldp.com) and you want to fetch it every one minute.

You can configure the FortiBalancer appliance as follows:

```
FortiBalancer(config)#ssl settings crl offline www.example.com
“http://www.crldp.com/Fortinet.crl” 1
```

This will cause the FortiBalancer appliance to fetch the CRL file at the regular interval of one minute from the “www.crldp.com” site by utilizing HTTP.

You can also specify an FTP URL to download the CRL file.

```
FortiBalancer(config)#ssl settings crl offline www.example.com
“ftp://ftp.crldp.com/Fortinet.crl” 1
```

You may also specify an LDAP URL to download the CRL file.

```
FortiBalancer(config)#ssl settings crl offline www.example.com
"ldap://ldap.crldp.com/cn=fortibalancer,dc=fortinet,dc=com" 1
```

➢ Step 8 Configure OCSP for SSL virtual host to check the certificate validation online

The FortiBalancer appliance supports the OCSP (Online Certificate Status Protocol) protocol. You may configure the FortiBalancer appliance to validate the certificate on an OCSP server online.

For our example, configure an OCSP server (ocsp.crldp.com:8888) and to validate the certificate online, you may configure the FortiBalancer appliance as follows:

```
FortiBalancer(config)#ssl settings ocsp www.example.com "http:// ocsp.crldp.com:8888"
```

Note: The OCSP has top priority. When configured, the OCSP will validate the certification by only checking the OCSP server.

➢ Step 9 Configure redirect for clients without strong encryption support

The FortiBalancer appliance provides you with a facility to redirect the weak clients (clients who are not using strong ciphers) to another URL. You can specify the minimum strength of the cipher as acceptance criteria. Any client that uses a cipher weaker than this will be redirected to the configured URL.

For example, consider a scenario where you want to redirect all clients that does not support cipher suites with at least 168 bits key length to a different site "www.example2.com".

This can be configured by using the following command:

```
FortiBalancer(config)#ssl settings minimum www.example.com 168
"http://www.example2.com"
```

➢ Step 10 Apply modified SSL settings

You will need to activate the SSL virtual host to take advantage of all the configuration steps taken to this point.

```
FortiBalancer(config)#ssl start www.example.com
```

11.3.2.5 Advanced SSL Configuration for SSL Real Host

You can configure different SSL settings for your SSL real host.

➢ Step 1 Stop SSL real host

```
FortiBalancer(config)#ssl stop www.myreal.com
```

This will stop SSL real host and will allow you to change SSL settings for this host.

➢ Step 2 Configure ciphersuites for the SSL real host

```
FortiBalancer(config)#ssl settings ciphersuite "www.myreal.com" "DES-CBC3-SHA"
```

The cipher suite settings allow you to define ciphers for this SSL real host. Only a limited set of ciphers are allowed for real hosts.

- DES-CBC3-SHA
- RC4-SHA
- RC4-MD5
- AES128-SHA

- AES256-SHA

➢ Step 3 Configure protocol version for SSL real host

```
FortiBalancer(config)#ssl settings protocol "www.myreal.com" "SSLv3:TLSv1"
```

The FortiBalancer appliance supports the protocols SSLv3 and TLSv1. You may use one or both of the two protocols.

➢ Step 4 Configure session reuse for SSL real host

This allows you to enable SSL session reuses between the FortiBalancer appliance and backend servers. This feature is enabled by default.

```
FortiBalancer(config)#ssl settings reuse www.myreal.com
```

➢ Step 5 Configure client authentication for SSL real host

The FortiBalancer appliance can use SSL client authentication while communicating with the backend server. If this setting is enabled, the FortiBalancer appliance will submit the client certificate to the backend sever for authentication during SSL handshake.

```
FortiBalancer(config)#ssl settings clientauth www.myreal.com
```

IMPORTANT: If you want to enable client authentication for an SSL real host, you will need to import a certificate and key pair for the SSL real host. The SSL real host will present this certificate to the backend server for authentication. This may be accomplished by using the "ssl import certificate" and "ssl import key" commands for an SSL real host. These two commands work exactly the same for an SSL virtual host and an SSL real host. For detailed instruction on using these commands, please refer to the SSL virtual host configuration described earlier.

➢ Step 6 Configure checking common name of real server certificate

If you want to verify the certificate of the real backend server, you will need to turn on global settings for verifying the server certificate. In addition, make certain the common name of the server certificate matches a specific name by running the command "**ssl settings servername**".

For example, if the certificate common name of the real server associated with the real host "www.myreal.com" is "Myreal Inc.", you can use the following command:

```
FortiBalancer(config)#ssl settings servername www.myreal.com "Myreal Inc."
```

➢ Step 7 Import trusted CA certificate for SSL real host

Since the SSL subsystem acts like a client to the real server, it has several root CA certificates just like a common Web browser. If you are using a self-signed certificate, or a certificate issued by your own local CA on your origin servers, then you need to use the "**ssl import rootca**" command to import the self-signed certificate that is on the real server or the local CA certificate.

The certificate must be in PEM format and is imported the same way you import a PEM certificate. The FortiBalancer appliance will prompt you to cut and paste the text to the terminal and enter "..." to accept the certificate.

```
FortiBalancer(config)#ssl import rootca
```

➢ Step 8 Apply modified SSL settings

You will need to activate the SSL real host to take advantage of all the configuration steps taken to this point.

```
FortiBalancer(config)#ssl start www.myreal.com
```

Chapter 12 Quality of Service (QoS)

12.1 Overview

This chapter introduces how to setup the QoS (Quality of Service) function on the FortiBalancer appliance. We setup the QoS functionality to provide administrators with the control over network bandwidth and allow them to manage the network from the business perspective, rather than the technical perspective.

12.2 Understanding QoS

QoS for networks is an industry-wide set of standards and mechanisms for ensuring high-quality performance for critical applications. By using QoS mechanisms, network administrators can use existing resources efficiently and ensure the required service level without reactively expanding or over-provisioning their networks.

QoS provides network administrators with the capacity of TCP, UDP and ICMP flow management by using queuing mechanism and packet filtering policies. By using queuing mechanism and filter rules, QoS supports both bandwidth management and priority control.

12.2.1 Queuing Mechanism

The FortiBalancer appliance has developed a queue-based QoS. Queue means a queue of network packet buffers. After the packet at the beginning of the queue has been processed, a new packet to be processed will be put at the end of the queue.

Each queue is bound with a particular network interface and controls either incoming or outgoing network traffic of that interface. QoS queues are organized in tree-like structures. On the top of a tree, a root queue is defined for either incoming or outgoing traffic of a network interface. Under the root queue, there can be multiple sub-queues. Sub-queues can also have their sub-queues. For each interface, at most two queue trees can be configured: one for the incoming traffic, and the other for the outgoing data.

Each queue is configured with bandwidth limit and priority for packet processing.

12.2.2 Packet Filter Rule

A QoS filter is a rule which associates particular network traffic with a QoS queue.

In filter rule, the network traffic is specified by five parameters: source IP subnet, source port, destination IP subnet, destination port and protocol. By this association, administrators can deploy either application-oriented or link-oriented QoS control. Normally, application-oriented filter rules have TCP or UDP ports defined while link-oriented filter rules focus on source or destination IP addresses.

12.2.3 Bandwidth Management

Bandwidth management is realized by a set of QoS filter rules which bind particular network traffic to pre-defined QoS queues with limited bandwidth settings. The QoS filter rules help FortiBalancer appliance servers to allocate appropriate bandwidth to satisfy the needs from various applications and links.

For more flexible bandwidth control, “BORROW/UNBORROW” strategy is applied to QoS queues in a tree-like structure. When a queue’s “BORROW” flag is turned on, its bandwidth can be expanded by borrowing from its parent queue. If the parent queue does not have extra bandwidth to share, it can also fall back on its parent, until the parent queue is the root queue.

12.2.4 Priority Control

Priority Control is accomplished by QoS queues in different priorities. All packets from different applications or links are firstly classified by QoS filter rules and then distributed to predefined queues enjoying the pre-configured priorities.

This priority mechanism works well especially when the network become crowded. If the traffic reaches a peak, packet loss will arise when the number of packets waiting for processing exceeds the maximum queuing buffers. Under such circumstance, the packets belonging to the queues with the highest priority will be processed in the first place, while other packets with lower priorities may be dropped. In this way, the mission-critical applications will be assigned with the highest priority, therefore the functionality of the most important transactions is guaranteed.

12.3 QoS Configuration

12.3.1 Configuration Guidelines

Table 12-1 General Settings of QoS

Operation	Command
Configure QoS interface	**qos interface** *<interface_name> [direction] [bandwidth]*
Define QoS queue	**qos queue root** *<queue_name> <interface_name> [direction] [bandwidth] [priority] [borrow] [default]* **qos queue sub** *<queue_name> <parent_queue> [bandwidth] [priority] [borrow] [default]*
Define QoS filter rules	**qos filter** *<filter_name> <queue_name> < src_addr> <smask> <sport> <dst_addr> <dmask> <dport> <proto> [priority]*
Enable QoS	**qos enable** *<interface_name> [direction]*

12.3.2 Configuration Example via CLI

➢ Step 1 Define QoS interfaces

```
FortiBalancer(config)#qos interface port1 OUT 5Mb
FortiBalancer(config)#qos interface port1 IN 5Mb
```

➢ Step 2 Define outgoing QoS queues

```
FortiBalancer(config)#qos queue root qr_oall port1 OUT 5Mb 3
FortiBalancer(config)#qos queue sub qs_ossh qr_oall 2Mb 3 UNBORROW NONDEFAULT
FortiBalancer(config)#qos queue sub qs_oftp qr_oall 512kb 2 UNBORROW NONDEFAULT
FortiBalancer(config)#qos queue sub qs_odeflt qr_oall 8kb 3 UNBORROW DEFAULT
```

Default queue is for all the other packets which cannot hit any defined queues.

➢ Step 3 Define incoming QoS queues

```
FortiBalancer(config)#qos queue root qr_iall port1 IN 5Mb 3
FortiBalancer(config)#qos queue sub qs_issh qr_iall 2Mb 3 BORROW NONDEFAULT
FortiBalancer(config)#qos queue sub qs_iftp qr_iall 2Mb 2 BORROW NONDEFAULT
FortiBalancer(config)#qos queue sub qs_ideflt qr_iall 8kb 3 BORROW DEFAULT
```

➢ Step 4 Define QoS filter rules

```
FortiBalancer(config)#qos filter fltr_ftp_o qs_oftp 0.0.0.0 0.0.0.0 0 10.3.54.40 255.255.255.255
0 tcp 2
FortiBalancer(config)#qos filter fltr_ftp_i qs_iftp 10.3.54.40 255.255.255.255 0 0.0.0.0 0.0.0.0 0
tcp 2
```

```
FortiBalancer(config)#qos filter fltr_ssh_o qs_ossh 0.0.0.0 0.0.0.0 22 0.0.0.0 0.0.0.0 0 tcp 3
FortiBalancer(config)#qos filter fltr_ssh_i qs_issh 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 22 tcp 3
```

- Step 5 Enable QoS

```
FortiBalancer(config)#qos enable port1 OUT
FortiBalancer(config)#qos enable port1 IN
```

Chapter 13 Link Load Balancing (LLB)

13.1 Overview

This chapter details the configuration of the following Inbound and Outbound Link Load Balancing implementations:

- Single FortiBalancer appliance and two ISPs
- Dual FortiBalancer appliances and two ISPs

13.2 Understanding LLB

LLB (Link Load Balancing) allows TCP/IP network traffic to be load balanced through up to 128 upstream Internet Service Providers (ISPs). Load balancing can be performed on egress to the Internet (outbound LLB) or on ingress from the Internet (inbound LLB). LLB methods include rr (Round Robin), wrr (Weighted Round Robin), sr (Shortest Response), dd (Dynamic Detecting) and hi (Hash IP). LLB also includes ISP/link failure detection through default gateway and link patch health checking.

The FortiBalancer appliance identifies links based on the logical port and peer MAC address. The statistics of LLB links are also collected based on the logical port and peer MAC address.

13.2.1 Outbound LLB

Outbound LLB provides optimized outbound link utilization for environments that have more than one default gateway. In essence, it allows outbound traffic to be distributed among multiple upstream/ISP routers.

For example, let's say you have Internet connectivity provided by two ISPs: ISP1 and ISP2. ISP1 assigns address range 100.1.1.0/24 so that you may use them on your network devices. ISP2 assigns address range 200.1.1.0/24 so that you may use them on your network devices. Outbound LLB allows you to load balance outbound connections traffic through ISP1 and ISP2. Connections forwarded through ISP1 are NATTed to an address from the range assigned by ISP1. Connections forwarded through ISP2 are NATTed to an address from the range assigned by ISP2. Thus, inbound responses for those connections will return through the ISP that they were originally sent through. If Internet connectivity through one of the ISP links is lost or interrupted, the outbound traffic will no longer be sent through that ISP. All traffic will be distributed to the functional ISP.

FortiBalancer outbound LLB methods can work well on the data traffic based on TCP and UDP protocols. However, for the packets based on IP, IPsec or GRE protocols, FortiBalancer LLB methods cannot do load balancing well. To deal with this problem, FortiBalancer now supports LLB for the IP based packets so that the IP-based packets can be delivered to different links like the way the TCP/UDP packets are processed.

13.2.2 Inbound LLB

Inbound LLB provides service resiliency for inbound clients. Hosted services are visible to external clients via a separate IP address on the address space assigned by each ISP.

To illustrate, let's use the same example ISPs as mentioned previously. All external clients trying to connect to the addresses assigned by ISP1 will be routed through ISP1's backbone. All external clients trying to connect to addresses assigned by ISP2 will be routed through ISP2's backbone. Inbound LLB allows you to advertise a device or Virtual IP (VIP) using two IP addresses: one from ISP1 and the other from ISP2. A DNS server on the FortiBalancer appliance will respond to queries for configured domain names. The responses will contain an IP address from ISP1 or ISP2, both representing the same device or VIP. If Internet connectivity through one of the ISP links is lost, the DNS server will not respond with the address from the failed ISP. Clients will receive only the address from the functional ISP.

13.2.3 LLB Health Check

LLB Health Check is used to check whether the link between the FortiBalancer interface and the upstream device is available. This can be accomplished by broadcasting ARP requests at regular intervals and pinging a user-defined upstream IP address. Besides, TCP-based and DNS-based health checks are also supported. The ICMP, TCP and DNS types of health check all work in the userland. This greatly improves the health check performance.

Broadcasting ARP requests at regular intervals can check the availability of the link path between FortiBalancer interface and the upstream ISP router. Pinging a user-defined upstream IP address not only can verify if the link path between FortiBalancer interface and the upstream ISP router is available, but also verify the link path between upstream ISP router and user-defined upstream IP address. Multiple upstream IP addresses can be defined for reliable checking. If any of check point is pingable, the related link is usable. This ensures that the WAN link is up before forwarding traffic across that link.

13.2.4 LLB Methods

Outbound LLB supports the following load balancing methods:

- rr (Round Robin)
- wrrr (Weighted Round Robin)
- sr (Shortest Response Time)
- dd (Dynamic Detecting)
- hi (Hash IP)

Inbound LLB supports three load balancing methods:

- rr (Round Robin)
- wrrr (Weighted Round Robin)
- proximity

Round Robin distributes each new session to gateways in an alternating (round robin) way. This is the default load balancing method.

Weighted Round Robin is similar to Round Robin except that a bias (or weight) may be assigned to each gateway so that some gateways may receive more sessions than others. This allows more traffic to be directed through an ISP with higher bandwidth capacity.

Shortest Response Time: The link with the shortest response time will get the next request. Calculation of shortest response time of a link is based on the initiation process of each TCP connection (both inbound and outbound connections). For the most accurate result, there should be enough TCP traffic instead of a few long existing TCP connections or only UDP traffic.

Note:

If neither SLB traffic nor NAT traffic goes through the system, the LLB SR method cannot work properly.

The “sr” method cannot be used to load balance IP fragments, non-TCP/UDP packets, and reassembled UDP packets.

Dynamic Detecting performs proximity calculations through all available ISP paths to the destinations. By using parallel probe arithmetic, a request from the client will be sent to a destination by different ISP paths at the same time. When the first response returns, the optimal ISP with the shortest response time will be selected for this request and other ISP connections will be failed. For future outbound traffic to the same destination, FortiBalancer appliance will choose the best ISP connection, according to the results derived from these proximity calculations.

Hash IP distributes the outbound traffic among links in the way that the link with higher weight is routed with higher probability, by performing Hash operation on the source IP. When the chosen link is down, the system will carry another Hash operation on the links available. When HI is deployed as the LLB method, the IPflow function can be disabled.

Proximity: The IP address of the nearest DNS server will be sent to the client as the response. When a DNS request arrives, FortiBalancer will first search in the Eroute table reversely to find a proximity route matching the source address of the DNS request, and then give response to the client with the corresponding DNS server's IP address (A record) according to the Eroute gateway.

13.2.5 Policy-based Routing (Eroute)

LLB policies provide the methods necessary to allow administrators to direct outbound traffic to a preferred route based on the IP address (source and destination) and service type (mail, FTP, Web, etc.).Policy based routing, unlike regular routing, allows the inclusion of the source IP, source port and destination port as well as the protocol into the route selection. For example, using routing policy can ensure that all the traffic generated by AOL instant messenger always uses the same link. If instant messenger client uses different destination IP addresses in its requests and these requests are sent through the different routes, this might confuse the server and cause login failure. Configuring routing policy will prevent this problem. The CLI command for that would be:

```
FortiBalancer(config)#ip eroute aol_route 1500 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 5190 tcp gateway_ip 1
```

The FortiBalancer appliance supports at most 5000 eroutes.

IP region

Eroute supports IP region. Administrators are allowed to import pre-defined IP region table via HTTP, FTP or Local File method, and then execute the command "**ipregion route**" to apply the imported IP region table. This will generate a large number of Eroute configurations, without making complex configurations. Administrators are also allowed to export the IP region table via FTP URL or Local File method.

FortiBalancer appliance will check the contents of the file instead of the file type when an IP region file is imported. To ensure that the IP region file can be imported successfully, please pre-define the file contents strictly with the following items included in each entry:

- IP subnet (in CIDR format)
- Country name (optional, up to 7 bytes)
- Brief description (optional, up to 63 bytes)

These items must be separated with a "Tab". For example:

```
27.8.0.0/13     CN  China Unicom Chongqing Province network
27.36.0.0/14    CN  China Unicom Guangdong province network
```

Note:

1. By default, there are three predefined IP region tables including "predefined_cernet", "predefined_cnc" and "predefined_ct". It is recommended not to use the same name with the default predefined IP region tables.

2. The routes and proximity rules configured for IP region exist as a whole in the system. Administrators cannot change or remove a single route or a rule.

13.2.6 LLB Session Timeout

After an ISP link has been selected for an IP flow (source IP and destination IP) pair, all traffic with the same source IP and destination IP will be sent to the same ISP. After an IP flow has been idle for a period of time, the session will be removed. Subsequent IP flows will once again be distributed based on the load-balancing algorithm.

13.2.7 Route Priority

The administrator will need to provide the method necessary to allow end-users to direct outbound traffic to a preferred route based on the IP address and protocol type. FortiBalancer appliance supports variant types of routing rules in which eroute priority is higher than priority of the default and static routes. Default routes will have priority 1 and static routes 101-132 depending on the netmask; i.e. the static route with 24-bit netmask will have priority 124 and with 32-bit netmask will have priority 132. The routes that correspond to the interfaces will have priority 2000. The routes created based on the traffic that come from the local subnet are called droutes (Direct Route) and will have priority 2000.

The following table shows the priority of different types of routes:

Table 13-1 Route Priority

Name of Route	Priority
EROUTE-P	2001-2999
IROUTE, DROUTE	2000
RTS	1999
EROUTE-N	1001-1999
IPFLOW	1000-1999 (defaults to 1000)
STATIC ROUTE	101-132 (IPv4) 101-228 (IPv6)
DYNAMIC ROUTE	101-132 (IPv4) 101-228 (IPv6)
LLB LINK ROUTE	2
DEFAULT ROUTE	1

13.2.8 Link Bandwidth Management

For better link bandwidth management, the FortiBalancer appliance allows administrators to set a threshold value for the LLB link bandwidth.

When performing link selection for the outbound traffic, the system considers not only the routing policies configured for links but also the load status of each link. That is, when the current link has reached the configured bandwidth threshold, the FortiBalancer appliance will search for available links from matched routes according to the descending sequence of priorities. The FortiBalancer appliance first searches for available links from routes with the same priority as the current link. If all available links reach their bandwidth thresholds, the FortiBalancer appliance will search for available links from routes with lower priorities. If the gateways of all matched routes are down or reach the configured bandwidth thresholds, the FortiBalancer appliance will still choose the current link to transmit traffic.

In addition, the FortiBalancer appliance allows administrators to configure a priority for the LLB link bandwidth. If the priority of a matched route is higher than the LLB link bandwidth priority, the traffic will be directly forwarded through this route.

With the LLB bandwidth management function, you do not need to configure Eroutes with the same priorities for multiple links. This improves the efficiency and flexibility of link bandwidth configuration and management.

Note:

1. If the traffic hits a RTS or IPflow route, the traffic will be directly forwarded through the relevant LLB link no matter whether the LLB link reaches the bandwidth threshold.

2. If an Eroute has been configured with the source IP address, source mask, source port number, destination IP address, destination mask, and destination port number and these IP addresses and masks are set to 0.0.0.0 and port numbers are set to 0, the FortiBalancer appliance will not search for available links from the matched routes whose priorities are lower than 1000.

13.2.9 IPv6 Support for LLB

The FortiBalancer appliance provides broad IPv6 support for the LLB module, of which the Eroute, inbound and outbound LLB, link health check and IP region can all work in the IPv6 network environment. For the Eroute, the source IP, destination IP, gateway IP and IP region can all be configured with the IPv6 addresses. However, please note that only IPv4 or only IPv6 addresses can be configured in one IP region table. For outbound LLB, only route-based LLB supports IPv6 configurations, while NAT-based LLB does not.

13.3 LLB Configuration

13.3.1 Outbound LLB Configuration (One FortiBalancer Appliance)

In this implementation example, one FortiBalancer appliance will be configured to load balance outbound traffic through two ISPs.

If the single FortiBalancer appliance stopped working, the network connectivity would be interrupted. Therefore, we recommend the implementation example with two FortiBalancer appliances in section 13.3.2 Outbound LLB Configuration (Two FortiBalancer Appliances).

13.3.1.1 Configuration Guidelines

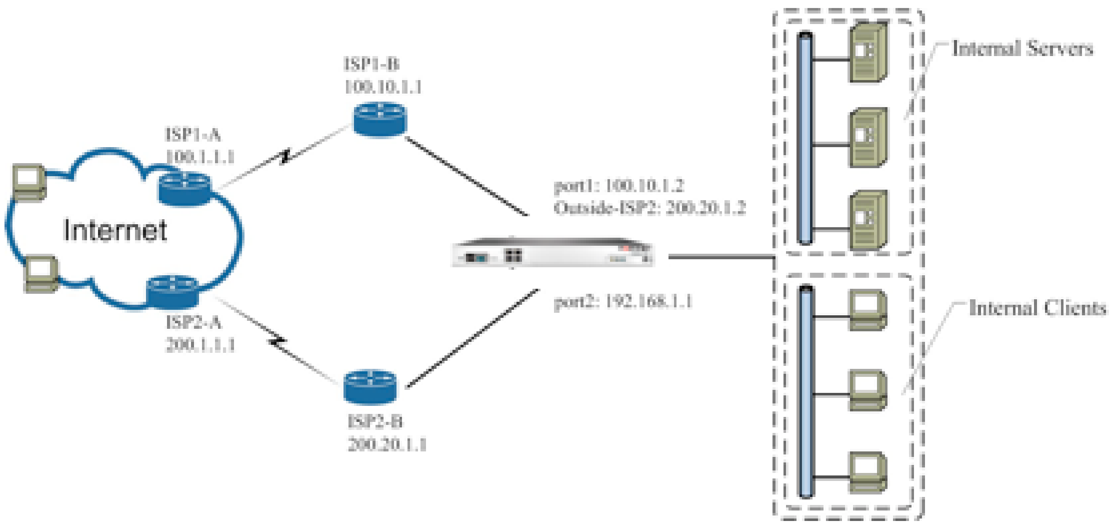


Figure 13-1 Outbound LLB (One FortiBalancer Appliance)

Table 13-2 General Settings of Outbound LLB

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}*
Configure MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*

Operation	Command
Configure LLB health check	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}** **llb link health checker icmp** *<link_name> <host> [hc_interval] [timeout] [hc_up] [hc_down]*
Configure outbound LLB method	**llb method outbound {rr|wrr|sr|hi}** **llb method outbound dd** *[netmask] [prefix]*
Configure Eroutes and manage link bandwidth	**ip eroute** *<name> <priority> <srcip> {srcmask|prefix} <srcport> <dsthost> {dstmask|prefix} <dstport> <proto> <gatewayip> [weight]* **llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link bw_priority** *<priority>*
Configure NAT	**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*

Enable IPflow and RTS	**ip ipflow {on|off}**
	ip rts {on|off}

13.3.1.2 Configuration Example via the CLI

➢ Step 1 Configure interface IP addresses

The Port1 interface IP will have an address from ISP1's address range. In order to assign an additional IP address on the Port1 interface, you must define and configure a multi-netted virtual interface (MNET). You will create an MNET named "outside_isp2" and assign it an IP address from ISP2's address range.

```
FortiBalancer(config)#ip address port1 100.10.1.2 255.255.255.0
FortiBalancer(config)#mnet port1 outside_isp2
FortiBalancer(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

Then, configure the IP address of the Port2 interface.

```
FortiBalancer(config)#ip address port2 192.168.1.1 255.255.255.0
```

➢ Step 2 Configure basic LLB health check

ISP link health checks are performed to ensure that the WAN link between the local router and the ISP router is up. This health check uses ICMP Ping to test connectivity.

Perform health check on an IP address on the other side of ISP1's WAN router:

```
FortiBalancer(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 0Mbps
```

Perform health check on an IP address on the other side of ISP2's WAN router:

```
FortiBalancer(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 0Mbps
```

Enter the following command to enable link health check:

```
FortiBalancer(config)#llb link health on
```

If a link is unstable, administrators can manually disable the link via the command "**llb link disable** *<link_name>*". For example, if the link ISP1 is found unstable, executing the command "**llb link disable ISP1**" will disable the link, and no outbound traffic will go through this link anymore. To enable a link, use the command "**llb link enable** *<link_name>*".

➢ Step 3 Configure additional LLB health check

Multiple health checkers can be configured for an ISP link.

```
FortiBalancer(config)#llb link health checker icmp ISP1 100.1.1.2 10
FortiBalancer(config)#llb link health checker icmp ISP1 100.1.1.3 10
FortiBalancer(config)#llb link health checker icmp ISP1 100.1.1.4 10
```

Here, 100.1.1.2, 100.1.1.3 and 100.1.1.4 are another three WAN routers of ISP1. Only when all the health checks (including basic health check) for ISP1 have failed, will the link ISP1 be deemed as down.

➢ Step 4 Configure outbound LLB method (optional)

The outbound LLB supports the following methods:

- Round Robin (rr)
- Weighted Round Robin (wrr)
- Shortest Response (sr)
- Dynamic Detecting (dd)
- Hash IP (hi)

The default method is “rr”.

In this example, we use the “wrr” method.

```
FortiBalancer(config)#llb method outbound wrr
```

➢ Step 5 Configure Eroutes and manage link bandwidth

To make different traffic go through different links, configure the Eroutes for two LLB links.

```
FortiBalancer(config)#ip eroute "er1" 1600 10.3.0.0 255.255.0.0 0 192.168.4.0 255.255.255.0 0
any 100.10.1.1 1
FortiBalancer(config)#ip eroute "er2" 1400 10.4.0.0 255.255.0.0 0 192.168.5.0 255.255.255.0 0
any 200.20.1.1 1
```

To make traffic that does not match the preceding Eroute configurations go through ISP1, configure the following Eroute:

```
FortiBalancer(config)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
```

If necessary, update the LLB link bandwidth thresholds.

```
FortiBalancer(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
FortiBalancer(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
```

You can set a priority for the link bandwidth threshold to determine whether the configured link bandwidth threshold takes effect for the relevant LLB link.

```
FortiBalancer(config)#llb link bw_priority 1500
```

Because the priority of Eroute “er1” is higher than the bandwidth priority, the gateway specified by the Eroute is not affected by the bandwidth threshold of ISP1. By comparison, the gateway specified by Eroute “er2” is affected by the bandwidth threshold of ISP2.

➢ Step 6 Configure NAT rules for outbound LLB

For an ISP that is selected for a session based on specific LLB method, the NAT rules for the ISP VIP must be pre-configured. These rules will be applied to the outbound traffic.

NAT for ISP1:

```
FortiBalancer(config)#nat port 100.10.1.10 192.168.1.0 255.255.0.0
```

NAT for ISP2:

```
FortiBalancer(config)#nat port 200.20.1.10 192.168.1.0 255.255.0.0
```

➢ Step 7 Other required configuration

Execute the following command to ensure that packets from the same connection will be directed to the same link by using the same NAT rule. By default, the IPflow function is disabled.

```
FortiBalancer(config)#ip ipflow on
```

RTS (Return to Sender) should be turned on by executing the following command to ensure that a response packet (e.g. ICMP response) will be directed to the link from which its corresponding request packet (e.g. ICMP request) is sent. By default, the RTS function is disabled.

```
FortiBalancer(config)#ip rts on
```

13.3.2 Outbound LLB Configuration (Two FortiBalancer Appliances)

In this implementation example, two FortiBalancer appliances will be configured to load balance outbound traffic through two ISPs. This is the preferred implementation approach because the secondary FortiBalancer appliance provides physical fault tolerance. If either FortiBalancer appliance should fail, network connectivity will not be interrupted.

13.3.2.1 Configuration Guidelines

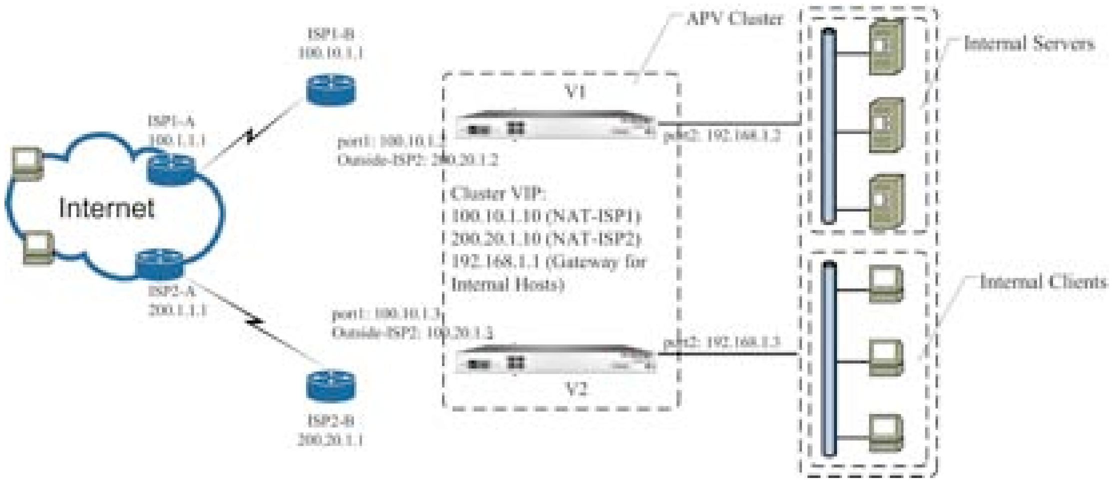


Figure 13-2 Outbound LLB (Two FortiBalancer Appliances)

Table 13-3 General Settings of Outbound LLB

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}*
Configure MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
Configure a cluster virtual router	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]* **cluster virtual ifname** *<interface_name> <cluster_id>* **cluster virtual vip** *<interface_name> <cluster_id> <vip>* **cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*

Operation	Command
Configure LLB health check	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}**
Configure cluster Virtual IPs for NATing traffic	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]* **cluster virtual ifname** *<interface_name> <cluster_id>* **cluster virtual vip** *<interface_name> <cluster_id> <vip>* **cluster virtual priority** *<interface_name> <cluster_id> <priority> [synconfig_peer_name]*
Configure Eroutes and manage link bandwidth	**ip eroute** *<name> <priority> <srcip> {srcmask|prefix} <srcport> <dsthost> {dstmask|prefix} <dstport> <proto> <gatewayip> [weight]* **llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link bw_priority** *<priority>*
Configure NAT	**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*

Enable IPflow and RTS	**ip ipflow {on|off}**
	ip rts {on|off}

13.3.2.2 Configuration Example via the CLI

Follow these steps to configure Outbound Link Load Balancing with clustered FortiBalancer appliances. Due to the additional configuration required for a secondary FortiBalancer appliance and to eliminate redundancy, this example assumes an understanding of the basic configuration that was illustrated in the previous section. Also, optional configuration settings will be left at their default values, and as a result, will not be illustrated in this example.

➢ Step 1 Configure interface IP addresses

You will need to define IP addresses on both FortiBalancer appliances. The same MNET names may be used on both FortiBalancer appliances.

(FortiBalancer1) Port1 and Port2 IP address configuration:

```
FortiBalancer1(config)#ip address port1 100.10.1.2 255.255.255.0
FortiBalancer1(config)#mnet port1 outside_isp2
FortiBalancer1(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
FortiBalancer1(config)#ip address port2 192.168.1.2 255.255.255.0
```

(FortiBalancer2) Port1 and Port2 IP address configuration:

```
FortiBalancer2(config)#ip address port1 100.10.1.3 255.255.255.0
FortiBalancer2(config)#mnet port1 outside_isp2
FortiBalancer2(config)#ip address outside_isp2 200.20.1.3 255.255.255.0
FortiBalancer2(config)#ip address port2 192.168.1.3 255.255.255.0
```

➢ Step 2 Configure a cluster virtual router for outbound traffic

Outbound traffic (from behind the FortiBalancer appliances) must be forwarded to an IP address on the FortiBalancer appliances. To provide a fault tolerant gateway for inside devices, a virtual cluster VIP must be configured.

(FortiBalancer1) Configure the first FortiBalancer appliance as the master virtual router for all interfaces so it will process outbound traffic. Assign it a higher priority than the secondary FortiBalancer appliance.

```
FortiBalancer1(config)#cluster virtual ifname port2 1
FortiBalancer1(config)#cluster virtual vip port2 1 192.168.1.1
```

```
FortiBalancer1(config)#cluster virtual priority port2 1 200
FortiBalancer1(config)#cluster virtual on 1 port2
```

(FortiBalancer2) Configure the secondary FortiBalancer appliance as a backup virtual router for all interfaces so it will not process outbound traffic unless the primary FortiBalancer appliance fails. Assign it a lower priority than the primary FortiBalancer appliance.

```
FortiBalancer2(config)#cluster virtual ifname port2 1
FortiBalancer2(config)#cluster virtual vip port2 1 192.168.1.1
FortiBalancer2(config)#cluster virtual priority port2 1 100
FortiBalancer2(config)#cluster virtual on 1 port2
```

- Step 3 Configure basic LLB health check

(Both FortiBalancers) Health check an interface on the other side of both ISPs' WAN routers and turn on default gateway health check:

```
FortiBalancer1(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 0Mbps
FortiBalancer1(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 0Mbps
FortiBalancer1(config)#llb link health on
FortiBalancer2(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 0Mbps
FortiBalancer2(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 0Mbps
FortiBalancer2(config)#llb link health on
```

- Step 4 Configure Eroutes and manage link bandwidth

To make different traffic go through different links, configure the Eroutes for two LLB links.

```
FortiBalancer(FortiBalancer1)#ip eroute "er1" 1600 10.3.0.0 255.255.0.0 0 192.168.4.0
255.255.255.0 0 any 100.10.1.1 1
FortiBalancer(FortiBalancer1)#ip eroute "er2" 1400 10.4.0.0 255.255.0.0 0 192.168.5.0
255.255.255.0 0 any 200.20.1.1 1
FortiBalancer(FortiBalancer2)#ip eroute "er1" 1600 10.3.0.0 255.255.0.0 0 192.168.4.0
255.255.255.0 0 any 100.10.1.1 1
FortiBalancer(FortiBalancer2)#ip eroute "er2" 1400 10.4.0.0 255.255.0.0 0 192.168.5.0
255.255.255.0 0 any 200.20.1.1 1
```

To make traffic that does not match the preceding Eroute configurations go through ISP1, configure the following Eroute:

```
FortiBalancer(FortiBalancer1)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any
100.10.1.1 1
FortiBalancer(FortiBalancer2)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any
100.10.1.1 1
```

If necessary, update the LLB link bandwidth thresholds.

```
FortiBalancer(FortiBalancer1)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
FortiBalancer(FortiBalancer1)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
FortiBalancer(FortiBalancer2)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
FortiBalancer(FortiBalancer2)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
```

You can set a priority for the link bandwidth threshold to determine whether the configured link bandwidth threshold takes effect for the relevant LLB link.

```
FortiBalancer(FortiBalancer1)#llb link bw_priority 1500
FortiBalancer(FortiBalancer2)#llb link bw_priority 1500
```

Because the priority of Eroute "er1" is higher than the bandwidth priority, the gateway specified by the Eroute is not affected by the bandwidth threshold of ISP1. By comparison, the gateway specified by Eroute "er2" is affected by the bandwidth threshold of ISP2.

➢ Step 5 Configure cluster Virtual IPs for NATing traffic

(FortiBalancer1) Cluster VIPs for NAT on each ISP. Assign a higher priority than the secondary FortiBalancer appliance.

```
FortiBalancer1(config)#cluster virtual ifname port1 1
FortiBalancer1(config)#cluster virtual vip port1 1 100.10.1.10
FortiBalancer1(config)#cluster virtual prio port1 1 200
FortiBalancer1(config)#cluster virtual on 1 port1
FortiBalancer1(config)#cluster virtual ifname outside-isp2 1
FortiBalancer1(config)#cluster virtual vip outside-isp2 1 200.20.1.10
FortiBalancer1(config)#cluster virtual prio outside-isp2 1 200
FortiBalancer1(config)#cluster virtual on 1 outside-isp2
```

(FortiBalancer2) Cluster VIPs for NAT on each ISP. Assign them a lower priority than the primary FortiBalancer appliance.

```
FortiBalancer2(config)#cluster virtual ifname port1 1
FortiBalancer2(config)#cluster virtual vip port1 1 100.10.1.10
FortiBalancer2(config)#cluster virtual prio port1 1 100
FortiBalancer2(config)#cluster virtual on 1 port1
FortiBalancer2(config)#cluster virtual ifname outside-isp2 1
FortiBalancer2(config)#cluster virtual vip outside-isp2 1 200.20.1.10
FortiBalancer2(config)#cluster virtual prio outside-isp2 1 100
FortiBalancer2(config)#cluster virtual on 1 outside-isp2
```

➢ Step 6 Configure NAT for outbound LLB sessions

(Both FortiBalancers) NAT rules for ISP1 and ISP2:

```
FortiBalancer1(config)#nat port 100.10.1.10 192.168.1.0 255.255.0.0
FortiBalancer1(config)#nat port 200.20.1.10 192.168.1.0 255.255.0.0
FortiBalancer2(config)#nat port 100.10.1.10 192.168.1.0 255.255.0.0
FortiBalancer2(config)#nat port 200.20.1.10 192.168.1.0 255.255.0.0
```

➢ Step 7 Other required configuration

Execute the following command to ensure that packets from the same connection will be directed to the same link by using the same NAT rule. By default, the IPflow function is disabled.

```
FortiBalancer(config)#ip ipflow on
```

RTS (Return to Sender) should be turned on by executing the following command to ensure that a response packet (e.g. ICMP response) will be directed to the link from which its corresponding request packet (e.g. ICMP request) is sent. By default, the RTS function is disabled.

```
FortiBalancer(config)#ip rts on
```

13.3.3 Inbound LLB Configuration

In this implementation example, a single FortiBalancer appliance will be configured to load balance inbound traffic.

13.3.3.1 Configuration Guidelines

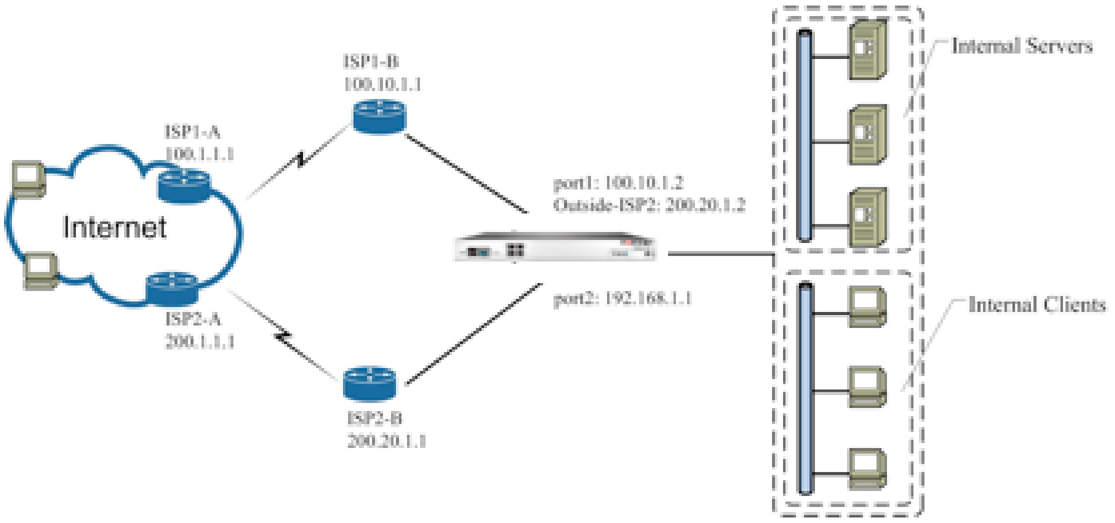


Figure 13-3 Inbound LLB

Table 13-4 General Settings of Inbound LLB

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}*
Configure MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
Configure LLB health check	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}**
Configure SLB	**slb real http** *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns] [hc_up] [hc_down]* **slb virtual http** *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]* **slb policy static** *<virtual_name> <real_name>*
Configure LLB DNS host and TTL	**llb dns host** *<host_name> <ip> [weight] [port] [link_name]* **llb dns ttl** *<host_name> [seconds]*
Configure load balancing method	**llb method inbound {rr|wrr|proximity}**

Enable IPflow and RTS	**ip ipflow {on|off}**
	ip rts {on|off}

13.3.3.2 Configuration Example via the CLI

Follow these steps to configure Inbound Link Load Balancing with a single FortiBalancer appliance.

➢ Step 1 Configure interface IP addresses

The Port1 interface IP will have an address from ISP1's address range. In order to assign an additional IP address on the Port1 interface, you must define and configure a multi-netted virtual interface (MNET). You will create an MNET named "outside_isp2" and assign it an IP address from ISP2's address range.

```
FortiBalancer(config)#ip address port1 100.10.1.2 255.255.255.0
FortiBalancer(config)#mnet port1 outside_isp2
FortiBalancer(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

Then, configure the IP address of the Port2 interface.

```
FortiBalancer(config)#ip address port2 192.168.1.1 255.255.255.0
```

- Step 2 Configure LLB health checks

ISP link health checks are performed to ensure that the WAN link between the local router and the ISP router is up. This health check uses ICMP Ping to test connectivity.

Perform health check on an interface on the other side of ISP1's WAN router:

```
FortiBalancer(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2
```

Perform health check on an interface on the other side of ISP2's WAN router:

```
FortiBalancer(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2
```

Enter the following command to enable link health check:

```
FortiBalancer(config)#llb link health on
```

- Step 3 Configure Server Load Balance

```
FortiBalancer(config)#slb virtual http vip1 100.10.1.10
FortiBalancer(config)#slb virtual http vip2 200.20.1.10
FortiBalancer(config)#slb real http server1192.168.1.100
FortiBalancer(config)#slb policy static vip1 server1
FortiBalancer(config)#slb policy static vip2 server1
```

- Step 4 Configure LLB DNS host and TTL for inbound

```
FortiBalancer(config)#llb dns host llb.fortinet.com 100.10.1.10 2
FortiBalancer(config)#llb dns host llb.fortinet.com 200.20.1.10 1
FortiBalancer(config)#llb dns ttl llb.fortinet.com 60
```

- Step 5 Configure inbound load balancing method

```
FortiBalancer(config)#llb method inbound wrr
```

Note: To use the "proximity" method for inbound load balancing, please first make configurations about "ip eroute".

- Step 6 Other required configuration

Execute the following command to ensure that packets from the same connection will be directed to the same link by using the same NAT rule. By default, the IPflow function is disabled.

```
FortiBalancer(config)#ip ipflow on
```

RTS should be turned on by executing the following command to ensure that a response packet (e.g. ICMP response) will be directed to the link from which its corresponding request packet (e.g. ICMP request) is sent. By default, the RTS function is disabled.

```
FortiBalancer(config)#ip rts on
```

Chapter 14 Global Server Load Balancing (GSLB)

14.1 Overview

GSLB (Global Server Load Balance) is also known as Smart DNS (SDNS). This function allows you to distribute Web traffic among a collection of servers deployed in multiple geographic locations. We will cover introduction of GSLB and the examples of GSLB configuration in this chapter.

14.2 Understanding Global Server Load Balance

In GSLB solution, the FortiBalancer appliance works as a complementary DNS server which is able to resolve a set of defined domain names based on load balancing methods. When DNS queries (typically forwarded by corporate DNS server or ISP DNS server) for the domain name are received, GSLB function will resolve the domain name with IP addresses selected from its **Domain Name and IP Service Database** with configured load balancing method.

SDNS maintains a local **Domain Name and IP Service Database** by continuously exchanging their local load (Hello message) and domain name/IP address information (Report message) with other members (also FortiBalancer appliances) in the GSLB network. For example, when an FortiBalancer appliance joins the SDNS network, the FortiBalancer appliance will continuously send its local domain name/IP address information to all other participating members (see LLB configuration). For each message transmitted, a confirmation message is expected in return. If a confirmation message is missed or a message is not updated for a period of time (3 tries), GSLB will mark the non-responsive member as down and all the domain name/IP addresses that are hosted by that FortiBalancer appliance will be removed from its local **Domain Name and IP Service Database**.

The SDNS process works as follows:

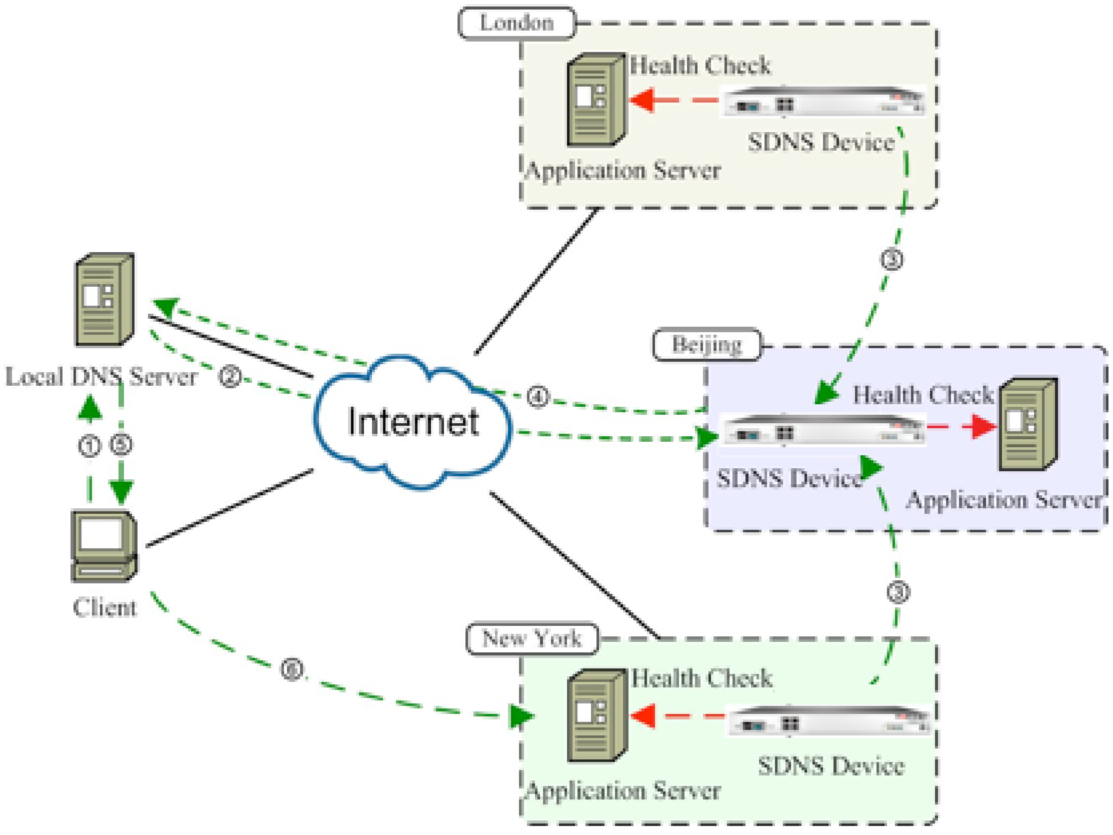


Figure 14-1 SDNS Working Mechanism

As shown in the above figure, the SDNS module will process a normal DNS request from the client as follows:

1. The client's browser generates a DNS request for the domain name of the Web site he wants to visit, and sends the request to its local DNS server.
2. The local DNS server receives the request and searches in its local cache. If no cache entry hits, it will forward the request to the upper-level SDNS device. In the above example figure, the request is sent to an SDNS server at Beijing according to configurations on the local DNS server.
3. The SDNS server at Beijing continuously collects the status information of all the application servers in its local Domain Name and IP Service Database, and then forwards the request to a proper application server based on pre-configured load balancing algorithms. In the above example, the application server at New York is selected.
4. The SDNS server at Beijing returns back the IP addresses of the application server at New York to the local application server of the client.
5. Upon receiving the response, the local application server forwards IP address to the client directly.
6. The client's browser uses the IP address in the response to open an HTTP connection with the corresponding FortiBalancer appliance and proceeds to download the Web page.

In this process, the response is cached on both the client's local DNS server and the client's browser.

Note: In this chapter, we will use the term "member" or "SDNS member" frequently. Either "member" or "SDNS member" is an FortiBalancer appliance which participates in the GSLB management.

14.2.1 SDNS Member Reporter-Receiver Hierarchy

All SDNS members can be divided into two groups: SDNS server and HTTP proxy cache server. They are all FortiBalancer appliances, while HTTP proxy cache servers serve as the "reporter" and SDNS servers serve as the "receiver".

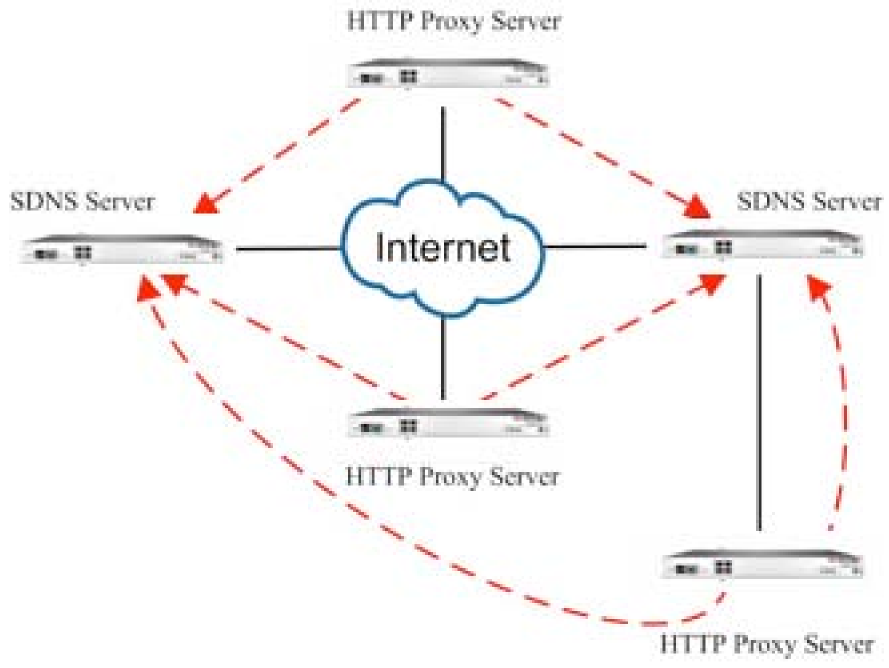


Figure 14-2 SDNS Reporter-Receiver Hierarchy

SDNS Servers

SDNS servers are responsible for DNS resolving. Every HTTP proxy cache server will report its status information to SDNS servers. The status information includes:

- The domain name configured on proxy cache servers
- The IPs which are configured for a domain name and their status ("UP" or "DOWN")
- The domain name traffic on proxy servers, IP traffic and proxy traffic
- The status of proxy cache servers ("UP" or "DOWN")

HTTP Proxy Cache Servers

HTTP proxy cache servers are responsible for HTTP services. All kinds of HTTP requests will be directed to HTTP proxy cache servers, mostly by the SDNS servers. The HTTP proxy cache servers will collect the local status information and send it to SDNS servers at specified frequency.

If an FortiBalancer appliance is a DNS server and a proxy cache server at the same time, it will report its local status information to all the SDNS servers (including itself) and collect the status information from all the proxy cache servers.

14.2.2 SDNS Load Balancing Methods

The following load balancing methods are supported by SDNS function:

- Global Round Robin (GRR)
- VIP-based Weighted Global Round Robin (VWGRR)
- Global Connection Overflow (GCO)
- Global Least Connection (GLC)
- IP Overflow (IPO)
- Proximity (Proximity)
- Region (Region)

14.2.2.1 GRR

When using the Global Round Robin (GRR) method, SDNS routes traffic to all participating members in a round robin fashion (See RR section of SLB methods for more details). When a user requests a service, the DNS query goes to SDNS servers. HTTP proxy cache servers will regularly report its local virtual service and link load/health information to SDNS servers. When a member receives such a message, it will re-calculate its round robin list. The maximum number of the returned IPs defaults to 3.

14.2.2.2 VWGRR

VIP-based Weighted Global Round Robin (VWGRR) is similar to GRR with the exception that each VIP is assigned a weight (i.e. the number of DNS hits). The traffic does not go to the next VIP until the DNS hits exceed the configured weight. After that, it moves on to the next VIP. When a member receives status report from other members, it will re-calculate its VWGRR list. The returned times of the host IP addresses equal to the weight value of the first returned IP. The maximum number of the returned IPs defaults to 3.

14.2.2.3 GCO

Global Connection Overflow (GCO) defines an overflow chain for a domain name. An overflow chain consists of different members. Each member is assigned a weight (the number of active TCP connections on the member). Network traffic is routed to the first FortiBalancer appliance in the overflow chain until the number of active TCP connections exceeds the maximum number configured for that appliance. Additional traffic overflows to the next member in the overflow chain, which can also overflow to the next one, and so on. An overflow chain has a default member (the last one). If all members are overflow, future traffic is routed to the default member. When an overflow member becomes under overflow, future traffic will be routed back to the first underflow FortiBalancer appliance in the chain. According to whether the number of the members in an overflow chain exceeds the maximum value of TCP connections or not, the host IPs on different members will be chosen. This method is useful to make sure no members (except the last one) are hosting too many TCP connections.

14.2.2.4 GLC

The Global Least Connection (GLC) algorithm will route traffic to the member that has the least number of TCP connections (See SLB LC for more information on this method). The host IP addresses on this member will be returned.

14.2.2.5 IPO

For a host name, there may be multiple related IP addresses. The client needs to set the priority for each IP address by using the "l**lb dns host**" command. In the VWGRR method, the last argument of the "**llb dns host**" command means the weight of an IP address. For the IP Overflow (IPO) method, the last argument of the "**llb dns host**" command means the priority. IPO method will resolve a host name to the related healthy IP address with the highest priority.

14.2.2.6 Proximity

When the Proximity method is configured for a host, all the queries for this host will be resolved based on a proximity algorithm. The resolving result depends on the distance between the client and the servers. The traffic will be routed to the client's nearest server. If the host method of a domain name is configured as "proximity", proximity and site distance should be configured by using the "**sdns proximity**" command and "**sdns site distance**" command. When a domain name is resolved, the request will be located in a site according to the source IP address and priority of the request packets. It will be directed to the appropriate site according to the site distance, and the host IP addresses on the members in this site will be returned.

14.2.2.7 Region

When the Region method is configured for a host name, "pool" should be created for the domain name and "rule" should be added to "pool". Besides, "proximity" should be configured by using the "**sdns proximity**" command. When a domain name is being resolved, firstly the request will be located to a site/region (pool) according to the source IP address and the priority of the request packets. Then the request will find the corresponding pool according to the priority which can be configured when the site/region is being configured. The host IP addresses will be returned according to the rule defined by the pool.

14.2.3 SDNS Member

Actually, a member is an FortiBalancer appliance. A member can be configured to be an SDNS server or an HTTP proxy cache server. It also can be configured to be an SDNS server and an HTTP proxy cache server at the same time.

Note: The SDNS members on the FortiBalancer 6.x version cannot cooperate with those on the FortiBalancer 8.x version since the data is encrypted in different formats. It is highly recommended that the FortiBalancer appliances configured as the SDNS sites be running the same software version.

14.2.4 SDNS Site

A site is a location which includes one or more members. A pool can be corresponding to a site. The commands related to "**sdns site**" work only when they are configured on SDNS servers. On HTTP proxy cache servers, these commands are invalid although they can be configured.

14.2.5 SDNS Proximity

Administrators can use the command "**sdns proximity**" to define proximity rules which are used to determine the site or region which a request source IP address belongs to.

SDNS Proximity supports IP region. Administrators are allowed to import pre-defined IP region table via HTTP, FTP or Local File method, and then execute the command "**sdns proximity ipregion**" to apply the imported IP region table. This will generate a large number of proximity rules, without making complex configurations. Administrators are also allowed to export the IP region table via FTP URL or Local File method.

To define the IP region files, the following entries should be included:

- IP subnet (in CIDR format)
- Country name (optional, up to 7 bytes)
- Brief description (optional, up to 63 bytes)

These items must be separated with a "Tab". For example:

```
27.8.0.0/13	CN	China Unicom Chongqing Province network
27.36.0.0/14	CN	China Unicom Guangdong province network
```

Note: By default, there are three predefined IP region tables including "predefined_cernet", "predefined_cnc" and "predefined_ct". It is recommended not to use the same name with the default predefined IP region tables. The routes and proximity rules configured for IP region exist as a whole in the system. Administrators cannot change or remove a single route or a rule.

14.2.6 SDNS Overflow

Only when the host method of a domain name is configured as "gco", which is configured by using the command "**sdns host method**", the overflow commands can work.

14.2.7 SDNS Region

Currently, SDNS region is only used in SDNS region method. Logically, "region" is a unit defined for classification management. A region can include one or more regions or sites and the lowest leaf node managed by a region is "site". Both region and site can be corresponding to a pool, and the logical classification ability of a region is designed for "pool" requirement. A pool name should be a region or a site name. Region commands must be used on SDNS servers. They are invalid on HTTP proxy cache servers.

14.2.8 SDNS Disaster Recovery Group

Disaster Recovery (DR) is defined by Disaster Recovery Group (DRG). A DRG contains two sub groups. One subgroup serves as the primary and the other as the standby. Each subgroup may contain one or more sites, while each site contains one or more FortiBalancer appliances. All the primary sites have higher priorities for being used as SDNS servers than the standby sites.

14.2.9 SDNS Bandwidth Management

Currently, bandwidth management only works with the SDNS region method. With bandwidth management, SDNS can limit the bandwidth of a member, site, region and IP, and collect the traffic statistics on a member, site, region and IP, and balance the Web requests to different sites and regions based on the current bandwidth usage.

Four kinds of bandwidth management are supported:

1. Bandwidth of a member (based on all the packets to or from the member)
2. Bandwidth of a site (sum of the bandwidth of all the members in this site)
3. Bandwidth of a region (sum of the bandwidth of all the sites or regions in this region)
4. Bandwidth of an IP address (based on the packets for the specified IP)

The system will collect the traffic of a domain name. The collected domain name traffic can be the domain name traffic of a site or region. When the DNS resolving is being done, the domain name traffic of the site or region will be considered. If the domain name traffic exceeds the configured bandwidth limit, it is considered that the DNS resolving will be done on the parent region of this region or site until the default region. If under the default region the host traffic still exceeds the bandwidth limit, this DNS resolving will return the host IPs by the method “round robin”.(A pool is corresponding to a region or site. All the above is only used under the condition that the host method is “region”.) Please note: if an SDNS member is configured as a “DNS” member, the SLB configuration on this member should be disabled; otherwise the bandwidth data will not be collected from proxy.

14.2.10 SDNS Pool

Currently, SDNS pool only works with the SDNS region method. When a host method of a domain name is configured as “region” by using the command “**sdns host method**” and the domain name is being resolved, “pool” is used. A pool contains a series of IP addresses, and a pool name is the same as a region or site name. The priority of a site or region is also the priority of a pool. IP addresses and rules can be added into a pool.

In SDNS, six pool methods are supported as follows:

Round Robin (rr)

If three IP addresses are in a pool, and round robin is chosen as pool method, the traffic will go to the next IP address in the order [1,2,3, 1, 2, 3…].

Weighted Round Robin (wrr)

Weighted Round Robin is similar to RR with the exception that each IP address is assigned a weight (i.e. the number of DNS hits) via the “**sdns pool ip**” command. The traffic does not go to the next IP address in a pool until the DNS hits exceed the configured weight. After that, it moves on to the next IP address.

IP Overflow (ipo)

In a pool, there may be multiple IP addresses. The client needs to set the priority for each IP address by using the “**sdns pool ip**” command. For “IPO” pool method, the last argument of the “**sdns pool ip**” command means the priority. IPO methods will resolve a host name to the related healthy IP address with the highest priority.

Hash IP (hi)

SDNS selects an IP address from all the IP addresses in a pool according to a hash number for the local DNS. The hash number is calculated via a hash function. As long as the hash number has the same value, the same IP address will be returned.

Persistent IP (pi)

When the SDNS server receives the first query request from a local DNS server, it will return the first IP address in the SDNS pool and record the returned IP address and the local DNS server's IP address. Hereafter, the recorded IP address will always be returned as response to the same query request until the local DNS times out. A timeout value shall be set for the local DNS in this pool method by using the command "**sdns persistent timeout**". In the meantime, the SDNS server counts the number of times an IP address is returned. When the number exceeds the weight value of the IP address, the SDNS server will return the next IP address as response to the query request from the unrecorded local DNS servers.

Simple Network Management Protocol (snmp)

By using the SNMP protocol, the SDNS server will collect the running status information of load balancing appliances or application servers at specified interval. The information collected includes six types: CPU usage, memory usage, total concurrent connections, new connections, throughput and user-defined SNMP service.

Since each commonly used SNMP service has a general OID (there might be a group of OIDs for throughput), besides the system defined SNMP services, users can also self-define the OID of the SNMP services as they need. This greatly facilitates the user of administrators.

When the SDNS server sends SNMP requests to the load balancing appliances or application servers to check their running status, the load balancing appliances or application servers will return the requested status information to the SDNS server. The information will be saved on the SDNS server and used together with the DNS resolving policies configured on the SDNS server for DNS resolving.

Note: The SDNS SNMP service can only be configured in the address pool of the load balancing appliances or application servers which are configured with the "region" method.

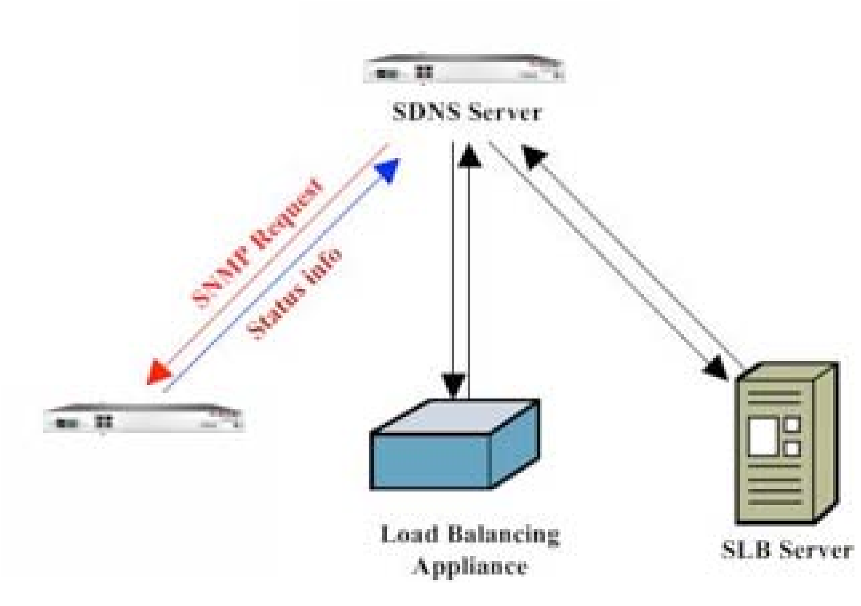


Figure 14-3 SDNS SNMP

The SNMP protocol is used to collect status information of load balancing appliances or application servers. To ensure the security of information exchange, users can configure the SNMP community required by the load balancing appliances or application servers by using the command "**sdns snmp ip**". Each appliance will report their status to the SDNS server. Upon receiving such status report, the SDNS server will re-calculate the IP priority information it saves for DNS resolving.

If only one SNMP service has been selected, in the "**des**" mode, the SDNS server will resolve the domain name to the IP address with the highest SNMP service value; while in the "**asc**" mode, the SDNS server will resolve the domain name to the IP address with the lowest SNMP service value.

If multiple SNMP services have been selected, users need to execute the command "**sdns pool snmp**" to set a weight value for each SNMP service.

The weight value of each host can be calculated according to the following formula:

```
metric = snmp service1 * weight1 + snmp service2 * weight2 + snmp service3 * weight3
```

The SDNS server will do DNS resolving based on the SNMP service value and configured weight value, and the IP address of the load balancing appliance or application server in the optimal status will be selected.

14.2.11 SDNS Pool Switchover

If the "region" method is configured for a domain name and the SDNS pool of the domain name adopts the "IP overflow (IPO)" method, the SDNS module will automatically select the IP address with the highest priority from the SDNS pool when resolving the domain name. Provided that the resolved IP address becomes unavailable, the IP address with the second highest priority available in the SDNS pool will be resolved. However, the new resolved IP address may not be the desired IP address. To solve this issue, the FortiBalancer appliance supports manually switching the resolved IP address to the desired IP address according to actual requirements.

In addition, when the resolved IP address switching is required for multiple domain names, you can add these domain names to a group and switch the resolved IP addresses for a group of domain names in batch.

14.2.12 SDNS IANA

SDNS integrates the latest global IANA addresses. When any IP address is entered, SDNS will inquire the latest global IANA address list and gain the corresponding country/district name.

14.2.13 SDNS Dynamic Proximity System (DPS)

SDNS Dynamic Proximity aims at providing a dynamically generated proximity rule table, instead of statically configured proximity rules for SDNS. All the dynamic proximity entries in the table are collected by proximity detection methods.

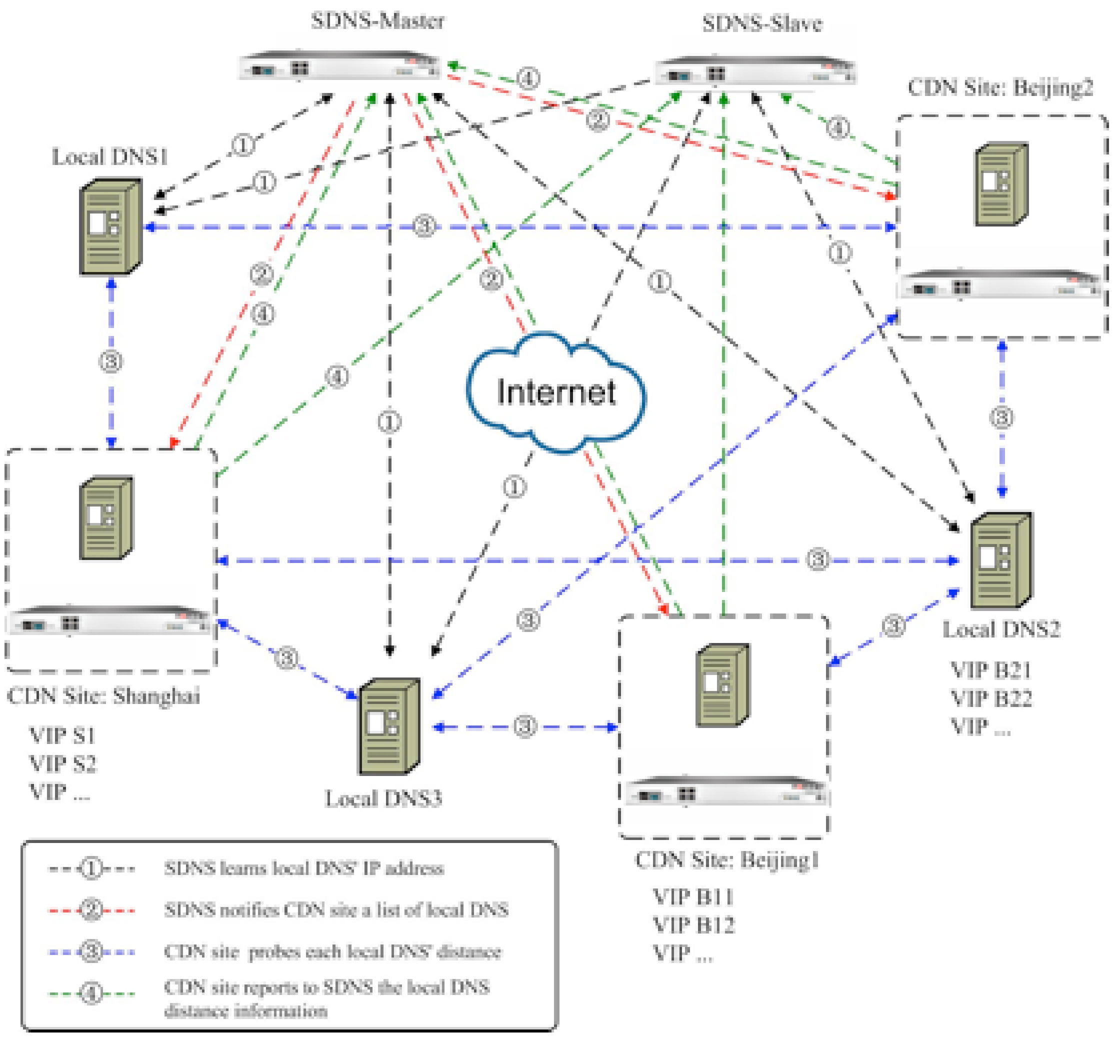

SDNS-Master
SDNS-Slave
CDN Site: Beijing2
Local DNS1
Internet
Local DNS2
VIP B21
VIP B22
VIP ...
CDN Site: Shanghai
VIP S1
VIP S2
VIP ...
Local DNS3
CDN Site: Beijing1
VIP B11
VIP B12
VIP ...
① SDNS learns local DNS' IP address
② SDNS notifies CDN site a list of local DNS
③ CDN site probes each local DNS' distance
④ CDN site reports to SDNS the local DNS distance information

Figure 14-4 SDNS DPS System

For SDNS DPS, DPS detectors are required for proximity detection and DPS servers are used for DNS resolution:

DPS detector:

- Get all the local DNS IP addresses from DPS masters for proximity detection
- Detect three kinds of dynamic proximity information by sending network requests to remote localDNS
- Report dynamic proximity detection results to DPS servers based on which DPS server may generate dynamic proximity rules

Note: The DPS detectors can be deployed on not only FortiBalancer appliances and but also servers that running the Linux or FressBSD operating system.

DPS server:

- Collect local DNS data and send them to DPS detectors for proximity detection ("**sdns statistics on localdns**")
- Generate dynamic proximity rules used for DNS resolution based on detection results
- Accept DNS requests and resolve domain names based on dynamic proximity rules

How SDNS DPS Works

Referring to the above figure, a complete DPS works as follows:

1. Users send DNS requests to SDNS DPS servers time and again;
2. SDNS DPS servers can have a collection of domain names and corresponding client IP addresses (local DNS IP addresses) after a certain period of time;
3. Some SDNS DPS servers (DPS masters) send the collected local DNS IP addresses to DPS detectors which are placed at all the CDN (Content Distribution Network) sites;
4. DPS detectors begin to detect the proximities between their CDN sites and local DNSs;
5. Once proximity detection is done, DPS detectors report the detection results to all the SDNS DPS servers;
6. SDNS DPS servers will resolve domain names based on the proximity detection results.

When SDNS function is turned off, SDNS DPS severs will be turned off as well. However, if SDNS is turned on, SDNS DPS server will not be turned on automatically, and it can be turned on by using the command "**sdns dps on**".

14.2.14 SDNS Full-DNS

SDNS supports all DNS resource record types, such as A record, AAAA record, MX record, CNAME record, PTR record etc. As an excellent domain name server, BIND 9 is integrated as a local DNS server in SDNS. Any query packets of A, AAAA and CNAME record types are processed by SDNS, while any other query packets are processed by BIND 9.

Note:

BIND 9 can only be configured via web UI, and CNAME can be configured via web UI or CLI.

CNAME records are only supported for the "region" method. If an SDNS host is not configured with the "region" method, the CNAME configurations on the host cannot work.

When SDNS recursive query is off, if the SDNS module or BIND 9 fails to process query packets, it will directly return failure responses. When SDNS recursive query is on, if the SDNS module fails to process query packets of the A, AAAA, or CNAME record type, BIND 9 will take over the processing of these query packets. Then, if BIND 9 still cannot process the query packets of the A, AAAA, or CNAME record type taken over from the SDNS module or the query packets of other record types, the FortiBalancer appliance will perform recursive queries by forwarding the query packets to other DNS servers for processing and return the final query results to the client.

Assume that only resource records of the A, AAAA, or CNAME type are available for the domain name "image.example.com". To ensure that the query packets of the other record types can be correctly processed, you are advised to configure a TXT record in the following format for the domain name "example.com" in the BIND 9 zone file:

```
image          IN          TXT          "A text string"
```

Note: The "text string" can be any descriptions about this domain name.

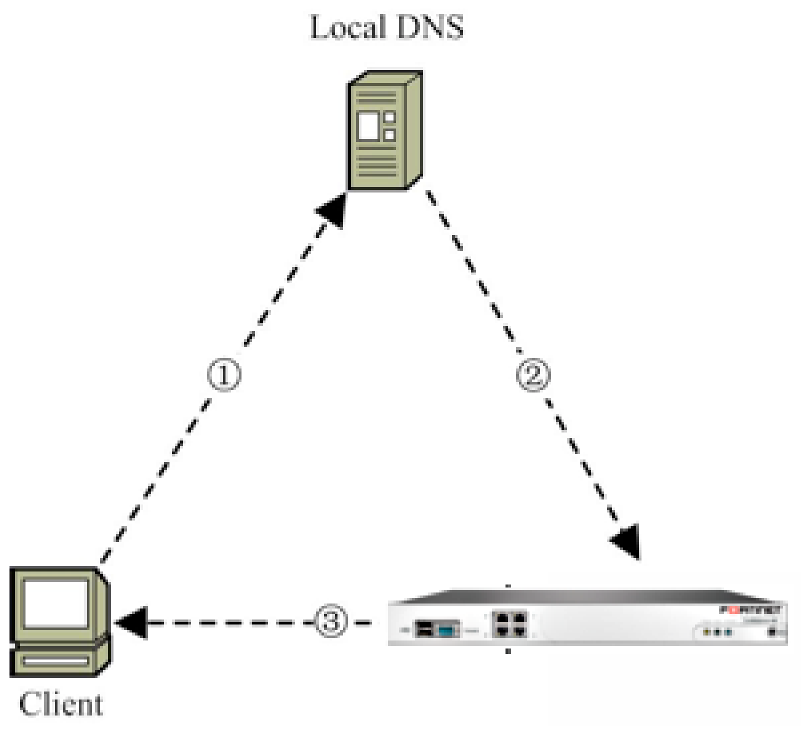


Figure 14-5 Full-DNS Working Flow

1. The client sends a DNS query to the local DNS.
2. The local DNS sends the query packets of any DNS record types (including A, AAAA, MX, CNAME, PTR etc) to FortiBalancer appliance.
3. FortiBalancer returns the corresponding DNS record responses.

Note: If the DNS query is of the A record type and a corresponding A record is found in the address pool, the FortiBalancer returns the A record. If the corresponding A record is not found, the FortiBalancer searches for the corresponding CNAME. If the CNAME is available, the FortiBalancer returns the CNAME and the A record corresponding to the CNAME. The FortiBalancer implements the same process on the DNS query of the AAAA record type. If the DNS query is of the CNAME type, the FortiBalancer searches for the corresponding CNAME. If the CNAME is available, the FortiBalancer returns only the CNAME.

14.2.15 IPv6 support for SDNS

The SDNS module can create the mapping between hostnames and IPv4 or IPv6 addresses through the A and AAAA records. Therefore, the hostnames in DNS queries not only from IPv4 clients but also from IPv6 clients can be resolved to IPv4 or IPv6 addresses.

To configure the AAAA record, it is required to use the “sdns ipv6”or “sdns pool ipv6” command. The AAAA records configured by using the “llb dns host” command cannot be used for the SDNS module. For the AAAA queries, the SDNS module supports only the RR method and does not support health check. Additionally, the IPv6 region table cannot be used in the SDNS module because SDNS proximity rules do not support IPv6.

14.3 GSLB Configuration

14.3.1 Configuration Guidelines

Table 14-1 General Settings of GSLB

Operation	Command
Configure SLB	Refer to the SLB Configuration section.
Configure LLB	Refer to the LLB Configuration section.
Configure basic SDNS	**sdns on** *[check|nocheck]* **sdns interval heartbeat** *[seconds]* **sdns interval report** *[seconds]* **sdns member attribute** *<member_name> <ip> [port] [member_type]* **sdns member local** *<member_name> [max_tcp_connections]*
Configure SDNS host method	**sdns host method** *<host_name> <method> [chain_name]* **sdns host ttl** *<host_name> <ttl>*
Configure region	**sdns region location** *<region_name> [region_weight]* **sdns region division** *<region_name> {region/site_name}*
Configure pool	**sdns pool method** *<host_name> {region|site} <pool_method> <number_of_vips> [pool_type]* **sdns pool rule** *<rule_name> {region|site} <pool_method> <number_of_vips>* **sdns pool ip** *{host|rule_name} <pool_name> <vip> [weight]*
Configure disaster recovery	**sdns group dr** *<group_name> <host_name>* **sdns group standby** *<group_name> <site_name>* **dns group primary** *<group_name> <site_name>*
Configure bandwidth	**sdns bandwidth** *{region|site|member|vip} {region|site|member|ip address} <mode> <maxbandwidth>*
Configure DPS	**sdns dps {on|off}** **sdns dps master {on|off}** *<port>* **sdns dps interval send** *<interval>* **sdns dps interval query** *<interval>* **sdns dps history** *<interval>* **sdns dps member** *<member_ip>* **sdns dps detector** *<site_name> <ip> [port] [detect_interval]*

14.3.1.1 Topology 1

Three FortiBalancer appliances are configured as “all”-type SDNS member in the figure below.

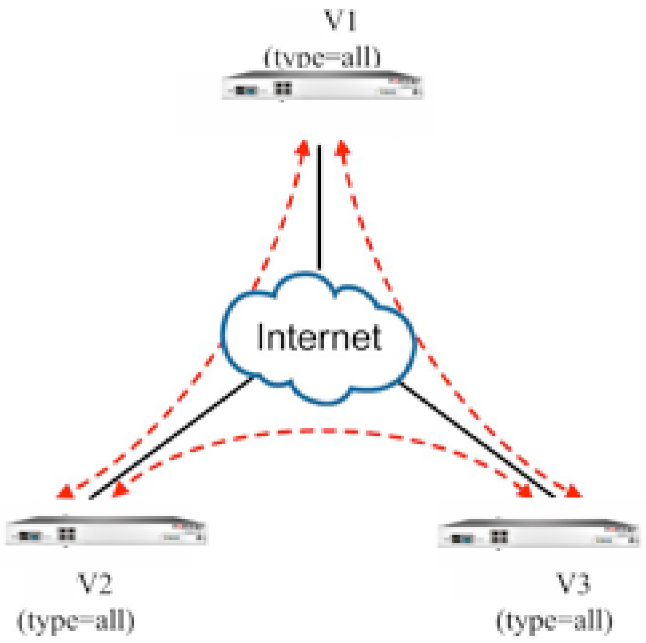


Figure 14-6 Network Topology 1

The port2 IP addresses of the three FortiBalancer appliances are as follows:

FortiBalancer1: 10.3.200.1

FortiBalancer2: 10.3.200.2

FortiBalancer3: 10.3.200.3

14.3.1.2 Topology 2

Among the three FortiBalancer appliances in the figure below, FortiBalancer1 is configured as “dns”-type SDNS member while FortiBalancer2 and FortiBalancer3 are configured as “proxy”-type SDNS members.

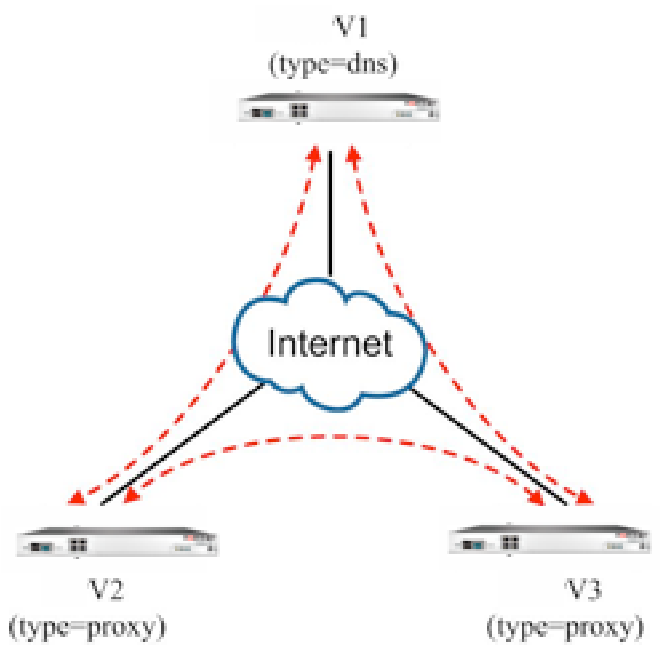


Figure 14-7 Network Topology 2

The port2 IP addresses of the three FortiBalancer appliances are as follows:

FortiBalancer1: 10.3.200.1

FortiBalancer2: 10.3.200.2

FortiBalancer3: 10.3.200.3

14.3.2 Configuration Example Based on Topology 1 via CLI

The basic configurations fall in to three sections as follows:

- SLB configuration: configure virtual IP address.
- LLB configuration: configure host name and assign IP addresses for it.
- SDNS basic parameter configurations.

14.3.2.1 Configuring SLB

FortiBalancer1

➢ Step 1 Configure a real server

```
FortiBalancer(config)#slb real http "rs1" 10.3.200.110 8080 1000 tcp 1 1
```

- Step 2 Configure avirtual server

```
FortiBalancer(config)#slb virtual http "vs1" 10.3.210.1 80
```

- Step 3 Configure SLB policy

```
FortiBalancer(config)#slb policy static "vs1" "rs1"
```

FortiBalancer2

- Step 1 Configure a real server

```
FortiBalancer(config)#slb real http "rs1" 10.3.200.110 8080 1000 tcp 1 1
```

- Step 2 Configure a virtual server

```
FortiBalancer(config)#slb virtual http "vs1" 10.3.220.1 80
```

- Step 3 Configure SLB policy

```
FortiBalancer(config)#slb policy static "vs1" "rs1"
```

FortiBalancer3

- Step 1 Configure a real server

```
FortiBalancer(config)#slb real http "rs1" 10.3.200.110 8080 1000 tcp 1 1
```

- Step 2 Configure a virtual server

```
FortiBalancer(config)#slb virtual http "vs1" 10.3.230.1 80
```

- Step 3 Configure SLB policy

```
FortiBalancer(config)#slb policy static "vs1" "rs1"
```

14.3.2.2 Configuring LLB

FortiBalancer1

- Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned with one IP address here.

```
FortiBalancer(config)#llb dns host "www.a.com" 10.3.210.1
FortiBalancer(config)#llb dns host "www.b.com" 10.3.210.1
FortiBalancer(config)#llb dns host "*.c.com" 10.3.210.1
```

- Step 2 Configure LLB DNS TTL (Time To Live)

```
FortiBalancer(config)#llb dns ttl "www.a.com" 60
FortiBalancer(config)#llb dns ttl "www.b.com" 60
FortiBalancer(config)#llb dns ttl "*.c.com" 60
```

FortiBalancer2

- Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned with one IP address here.

```
FortiBalancer(config)#llb dns host "www.a.com" 10.3.220.1
FortiBalancer(config)#llb dns host "www.b.com" 10.3.220.1
FortiBalancer(config)#llb dns host "*.c.com" 10.3.220.1
```

- Step 2 Configure LLB DNS TTL (Time To Live)

```
FortiBalancer(config)#llb dns ttl "www.a.com" 60
FortiBalancer(config)#llb dns ttl "www.b.com" 60
FortiBalancer(config)#llb dns ttl "*.c.com" 60
```

FortiBalancer3

- Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned with one IP address here.

```
FortiBalancer(config)#llb dns host "www.a.com" 10.3.230.1
FortiBalancer(config)#llb dns host "www.b.com" 10.3.230.1
FortiBalancer(config)#llb dns host "*.c.com" 10.3.230.1
```

- Step 2 Configure LLB DNS TTL (Time To Live)

```
FortiBalancer(config)#llb dns ttl "www.a.com" 60
FortiBalancer(config)#llb dns ttl "www.b.com" 60
FortiBalancer(config)#llb dns ttl "*.c.com" 60
```

14.3.2.3 Configuring Basic SDNS

FortiBalancer1

- Step 1 Enable SDNS

```
FortiBalancer(config)#sdns on
```

- Step 2 Configure SDNS members (their types are "all")

```
FortiBalancer(config)#sdns member attribute FortiBalancer1 10.3.200.1 5888 all
FortiBalancer(config)#sdns member attribute FortiBalancer2 10.3.200.2 5888 all
FortiBalancer(config)#sdns member attribute FortiBalancer3 10.3.200.3 5888 all
```

- Step 3 Configure FortiBalancer1 as a local member

```
FortiBalancer(config)#sdns member local FortiBalancer1
```

FortiBalancer2

- Step 1 Enable SDNS

```
FortiBalancer(config)#sdns on
```

- Step 2 Configure SDNS members (their types are "all")

```
FortiBalancer(config)#sdns member attribute FortiBalancer1 10.3.200.1 5888 all
FortiBalancer(config)#sdns member attribute FortiBalancer2 10.3.200.2 5888 all
FortiBalancer(config)#sdns member attribute FortiBalancer3 10.3.200.3 5888 all
```

- Step 3 Configure FortiBalancer2 as a local member

```
FortiBalancer(config)#sdns member local FortiBalancer2
```

FortiBalancer3

- Step 1 Enable SDNS

```
FortiBalancer(config)#sdns on
```

- → Step 2 Configure SDNS members (their types are “all”)

```
FortiBalancer(config)#sdns member attribute FortiBalancer1 10.3.200.1 5888 all
FortiBalancer(config)#sdns member attribute FortiBalancer2 10.3.200.2 5888 all
FortiBalancer(config)#sdns member attribute FortiBalancer3 10.3.200.3 5888 all
```

- Step 3 Configure FortiBalancer3 as a local member

```
FortiBalancer(config)#sdns member local FortiBalancer3
```

14.3.2.4 Configuring Host Method

grr

FortiBalancer3 is configured as local DNS to resolve a domain name “www.a.com” by following the above basic configurations.

- Step 1 Assign “grr” host method to “www.a.com” on FortiBalancer3

```
FortiBalancer(config)#sdns host method “www.a.com” grr
```

The resolving results are displayed through nslookup of Windows as follows:

```
www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.230.1, 10.3.220.1, 10.3.210.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.220.1, 10.3.210.1, 10.3.230.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.210.1, 10.3.220.1, 10.3.230.1
```

As is obvious from the above, the result of resolving “www.a.com” is round robin on the three IP addresses, 10.3.230.1, 10.3.220.1, and 10.3.210.1.

vwgrr

Besides the above basic configurations, it is necessary to set the weights of the IP addresses which a domain name is corresponding to. (In the basic configurations, the weights of all the IP addresses default to 1.) FortiBalancer3 is configured as local DNS to resolve “www.a.com”.

- Step 1 Set the weight of “www.a.com” to 1 on FortiBalancer1

```
FortiBalancer(config)#llb dns host “www.a.com” 10.3.210.1 1
```

- Step 2 Set the weight of “www.a.com” to 2 on FortiBalancer2

```
FortiBalancer(config)#llb dns host “www.a.com” 10.3.220.1 2
```

- Step 3 Set the weight of “www.a.com” to 3 on FortiBalancer3

```
FortiBalancer(config)#llb dns host "www.a.com" 10.3.230.1 3
```

➢ Step 4 Assign "vwgrr" host method to "www.a.com" on FortiBalancer3

```
FortiBalancer(config)#sdns host method "www.a.com" vwgrr
```

And the resolving results are displayed through nsookup of Windows as follows:

```
>www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.210.1, 10.3.220.1, 10.3.230.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.220.1, 10.3.230.1, 10.3.210.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.220.1, 10.3.230.1, 10.3.210.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.230.1, 10.3.220.1, 10.3.210.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.230.1, 10.3.220.1, 10.3.210.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Addresses: 10.3.230.1, 10.3.220.1, 10.3.210.1
```

As is obvious from the above, when "www.a.com" is resolved in terms of different weights of three IP addresses, the three IP addresses' return times are different (refer to the following table).

Table 14-2 Weight and Return Times of IP Addresses

IP	Weight	Continuous returned times
10.3.210.1	1	1
10.3.220.1	2	2
10.3.230.1	3	3

gco

FortiBalancer3 is configured as local DNS to resolve a domain name “www.a.com”. Besides the basic configurations, SDNS chain needs to be configured.

FortiBalancer3

- Step 1 Configure an overflow chain called “mychain” on FortiBalancer3

```
FortiBalancer(config)#sdns overflow chain mychain
```

- Step 2 Add FortiBalancer1, FortiBalancer2, and FortiBalancer3 into “mychain”

```
FortiBalancer(config)#sdns overflow member mychain FortiBalancer1
FortiBalancer(config)#sdns overflow member mychain FortiBalancer2
FortiBalancer(config)#sdns overflow member mychain FortiBalancer3
```

Note: The earlier an FortiBalancer is added, the higher priority it will be assigned.

- Step 3 Assign “gco” host method to “www.a.com” on FortiBalancer3

```
FortiBalancer(config)#sdns host method “www.a.com” gco mychain
```

- Step 4 Set the maximum number of TCP connections to 3

```
FortiBalancer(config)#sdns member local FortiBalancer3 3
```

FortiBalancer1

- Step 1 Set the maximum number of TCP connections to 1

```
FortiBalancer(config)#sdns member local FortiBalancer1 1
```

FortiBalancer2

- Step 1 Set the maximum number of TCP connections to 2

```
FortiBalancer(config)#sdns member local FortiBalancer2 2
```

The resolving results are displayed through nslookup of Windows as follows:

```
> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Address: 10.3.210.1

(Set up one TCP connection to FortiBalancer1)
> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Address: 10.3.220.1

(Set up two TCP connections to FortiBalancer2 and at the same time maintain the TCP connection
to FortiBalancer1)
> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3
```

```
Name: www.a.com
Address: 10.3.230.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3
(Break up the TCP connection to FortiBalancer1)
Name: www.a.com
Address: 10.3.210.1
```

As is obvious from the above, because the indexes of FortiBalancer1, FortiBalancer2, and FortiBalancer3 are respectively 1, 2, 3, the initial resolving of “www.a.com” will return IP addresses on FortiBalancer1. As the number of TCP connection on FortiBalancer1 is set to 1, the resolving of “www.a.com” will transfer to FortiBalancer2 after maintaining a connection to FortiBalancer1. The rest may be deduced by analogy. The resolving of “www.a.com” will transfer to FortiBalancer3 after maintaining two connections to FortiBalancer2. Once the TCP connection to FortiBalancer1 is broken up, the resolving of “www.a.com” will reuse the IP addresses on FortiBalancer1.

glc

FortiBalancer3 is configured as local DNS to resolve a domain name “www.a.com”. Besides the above basic configurations, TCP connection of every FortiBalancer appliance needs to be configured.

FortiBalancer3

- Step 1 Assign “glc” host method to “www.a.com” on FortiBalancer3

```
FortiBalancer(config)#sdns host method “www.a.com” glc
```

- Step 2 Set the maximum number of TCP connections to 3

```
FortiBalancer(config)#sdns member local FortiBalancer3 3
```

FortiBalancer1

- Step 1 Set the maximum number of TCP connections to 3

```
FortiBalancer(config)#sdns member local FortiBalancer1 3
```

FortiBalancer2

- Step 1 Set the maximum number of TCP connections to 3

```
FortiBalancer(config)#sdns member local FortiBalancer2 3
```

The resolving results are displayed through nslookup of Windows as follows:

```
> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3
(The number of TCP connection to FortiBalancer1 is 1, and 2 to FortiBalancer2 and
FortiBalancer3.)
Name: www.a.com
Address: 10.3.210.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3
```

```
(The number of TCP connection to FortiBalancer2 is 1, and 2 to FortiBalancer1 and
FortiBalancer3.)
Name: www.a.com
Address: 10.3.220.1

> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

(The number of TCP connection to FortiBalancer3 is 1, and 2 to FortiBalancer1 and
FortiBalancer2.)
Name: www.a.com
Address: 10.3.230.1
```

As is obvious from above, when “www.a.com” is resolved, the IP address on the FortiBalancer appliance with the least TCP connections will be returned.

ipo

FortiBalancer3 is configured as local DNS to resolve a domain name “www.a.com”. Besides the above basic configurations, IP address’ priority should be configured.

FortiBalancer3

- Step 1 Assign “ipo” host method to “www.a.com” on FortiBalancer3

```
FortiBalancer(config)#sdns host method “www.a.com” ipo
```

- Step 2 Set “www.a.com” priority to 3

```
FortiBalancer(config)#llb dns host “www.a.com” 10.3.230.1 3
```

FortiBalancer1

- Step 1 Set www.a.com priority to 1

```
FortiBalancer(config)#llb dns host “www.a.com“ 10.3.210.1 1
```

FortiBalancer2

- Step 1 Set “www.a.com” priority to 2

```
FortiBalancer(config)#llb dns host “www.a.com” 10.3.220.1 2
```

And the resolving results are displayed through nslookup of Windows as follows:

```
> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Address: 10.3.230.1

llb dns host “www.a.com“ 10.3.220.1 5
> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Address: 10.3.220.1

(Set the priority of 10.3.210.1 to 8 which is the highest value among the three IP addresses.)
```

```
lb dns host "www.a.com" 10.3.210.1 8
> www.a.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.a.com
Address: 10.3.210.1
```

This shows that every DNS resolving will return the IP address with the highest priority.

proximity

The logical architecture related to SDNS site should be mentioned here. The labeled numbers in the following figure are the setting distance values. (These values have nothing to do with the length of the lines in this figure.)

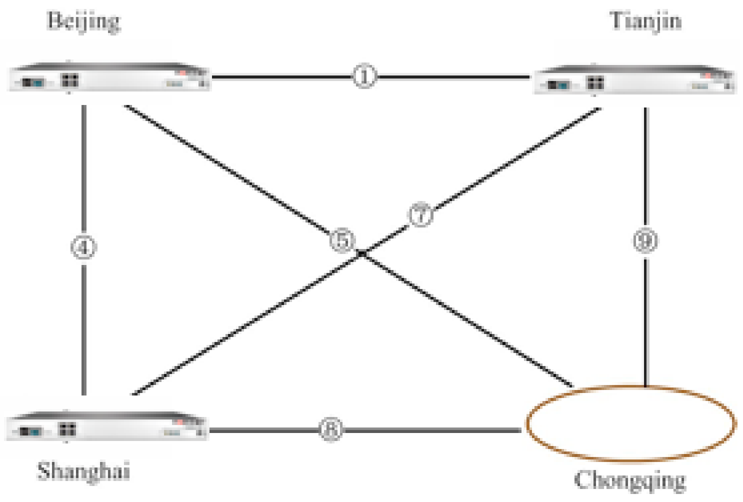


Figure 14-8 Proximity Method

In the above figure, every site has a member, but Chongqing site has no member.

FortiBalancer1

➢ Step 1 Configure each site (respectively Beijing, Tianjin, Shanghai and Chongqing)

```
FortiBalancer(config)#sdns site location beijing 42
FortiBalancer(config)#sdns site location tianjin 32
FortiBalancer(config)#sdns site location shanghai 22
FortiBalancer(config)#sdns site location chongqing 12
```

➢ Step 2 Configure the distance value between two sites

```
FortiBalancer(config)#sdns site distance "beijing" "tianjin" 1
FortiBalancer(config)#sdns site distance "beijing" "shanghai" 7
FortiBalancer(config)#sdns site distance "beijing" "chongqing" 5
FortiBalancer(config)#sdns site distance "tianjin" "shanghai" 9
FortiBalancer(config)#sdns site distance "tianjin" "chongqing" 5
FortiBalancer(config)#sdns site distance "shanghai" "chongqing" 8
```

➢ Step 3 Add the members into sites (Chongqing site has no member)

```
FortiBalancer(config)#sdns site member beijing FortiBalancer1
FortiBalancer(config)#sdns site member tianjin FortiBalancer2
FortiBalancer(config)#sdns site member shanghai FortiBalancer3
```

➢ Step 4 Configure proximity

```
FortiBalancer(config)#sdns proximity 10.3.50.7 255.255.255.255 beijing 0
FortiBalancer(config)#sdns proximity 10.3.200.107 255.255.255.255 tianjin 0
FortiBalancer(config)#sdns proximity 10.3.200.108 255.255.255.255 chongqing 0
```

- Step 5 Set “www.b.com” method to proximity

```
FortiBalancer(config)#sdns host method “www.b.com” proximity
```

- Step 6 Add IP address into “www.b.com”

```
FortiBalancer(config)#slb virtual http “vs2” 10.3.210.2 80
FortiBalancer(config)#slb virtual http “vs3” 10.3.210.3 80
FortiBalancer(config)#slb virtual http “vs4” 10.3.220.4 80
FortiBalancer(config)#slb policy static “vs2” “rs1”
FortiBalancer(config)#slb policy static “vs3” “rs1”
FortiBalancer(config)#slb policy static “vs4” “rs1”
FortiBalancer(config)#llb dns host “www.b.com” 10.3.210.2
FortiBalancer(config)#llb dns host “www.b.com” 10.3.210.3
FortiBalancer(config)#llb dns host “www.b.com” 10.3.220.4
```

FortiBalancer2

- Step 1 Add IP address into “www.b.com”

```
FortiBalancer(config)#slb virtual http “vs2” 10.3.210.2 80
FortiBalancer(config)#slb virtual http “vs3” 10.3.210.3 80
FortiBalancer(config)#slb virtual http “vs4” 10.3.220.4 80
FortiBalancer(config)#slb policy static “vs2” “rs1”
FortiBalancer(config)#slb policy static “vs3” “rs1”
FortiBalancer(config)#slb policy static “vs4” “rs1”
FortiBalancer(config)#llb dns host “www.b.com” 10.3.210.2
FortiBalancer(config)#llb dns host “www.b.com” 10.3.210.3
FortiBalancer(config)#llb dns host “www.b.com” 10.3.220.4
```

FortiBalancer3

- Step 1 Add IP address into “www.b.com”

```
FortiBalancer(config)#slb virtual http “vs2” 10.3.210.2 80
FortiBalancer(config)#slb virtual http “vs3” 10.3.210.3 80
FortiBalancer(config)#slb virtual http “vs4” 10.3.220.4 80
FortiBalancer(config)#slb policy static “vs2” “rs1”
FortiBalancer(config)#slb policy static “vs3” “rs1”
FortiBalancer(config)#slb policy static “vs4” “rs1”
FortiBalancer(config)#llb dns host “www.b.com” 10.3.210.2
FortiBalancer(config)#llb dns host “www.b.com” 10.3.210.3
FortiBalancer(config)#llb dns host “www.b.com” 10.3.220.4
```

Request for resolving “www.b.com” on three clients (their IP addresses are respectively 10.3.50.7, 10.3.200.107, and 10.3.200.108）by using nslookup of Windows. The resolving result is as follows:

The client whose IP address is 10.3.200.107 sets local DNS to 10.3.200.1.

```
> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.220.1, 10.3.220.2, 10.3.220.3
```

```
> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.220.2, 10.3.220.3, 10.3.220.4

> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.220.3, 10.3.220.4, 10.3.220.1
```

The result is as above. FortiBalancer appliance locates to Tianjin site by SDNS proximity, and then returns the IP addresses on the FortiBalancer2 of Tianjin site.

The client whose IP address is 10.3.50.7 sets local DNS to 10.3.200.3.

```
> www.b.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.b.com
Addresses: 10.3.210.1, 10.3.210.2, 10.3.210.3

> www.b.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.b.com
Addresses: 10.3.210.2, 10.3.210.3, 10.3.210.4

> www.b.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.b.com
Addresses: 10.3.210.3, 10.3.210.4, 10.3.210.1
```

Referring to the above results, FortiBalancer appliance locates to Beijing site by SDNS proximity, and then returns the IP addresses on the FortiBalancer1 of Beijing site.

The client whose IP address is 10.3.200.108 sets local DNS to10.3.200.1.

```
> www.b.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.b.com
Addresses: 10.3.210.1, 10.3.210.2, 10.3.210.3

> www.b.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.b.com
Addresses: 10.3.210.2, 10.3.210.3, 10.3.210.4
```

```
> www.b.com
Server: [10.3.200.3]
Address: 10.3.200.3

Name: www.b.com
Addresses: 10.3.210.3, 10.3.210.4, 10.3.210.1
```

The result is as above. FortiBalancer appliance locates to Chongqing site by SDNS proximity. But no member is added in Chongqing site. FortiBalancer appliance will compare the distance value between Chongqing site and another site and it will find that the distance between Chongqing site and Beijing site (the distance value is 4) is shorter than the distance between Chongqing site and Tianjin site (the distance value is 5). So at last FortiBalancer appliance will locate to Beijing site and return the IP addresses on the FortiBalancer1of Beijing site.

14.3.2.5 Configuring Disaster Recovery

Here we assume the host method is configured as “proximity”, then start to configure disaster recovery function.

FortiBalancer1

- Step 1 Create a new host name

```
FortiBalancer(config)#llb dns host “www.example.com” 10.3.210.1
```

- Step 2 Add a disaster recovery group

```
FortiBalancer(config)#sdns group dr arr “www.example.com”
```

- Step 3 Add a primary site

```
FortiBalancer(config)#sdns group primary arr Beijing
```

- Step 4 Add a standby site

```
FortiBalancer(config)#sdns group standby arr Tianjin
```

FortiBalancer2

- Step 1 Create a new host name

```
FortiBalancer(config)#llb dns host “www.example.com” 10.3.220.1
```

- Step 2 Add a disaster recovery group

```
FortiBalancer(config)#sdns group dr arr “www.example.com”
```

- Step 3 Add a primary site

```
FortiBalancer(config)#sdns group primary arr Beijing
```

- Step 4 Add a standby site

```
FortiBalancer(config)#sdns group standby arr Tianjin
```

“www.example.com” is being resolved through nslookup. The resolving results are as follows:

```
>www.example.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.example.com
```

```
Address: 10.3.210.1

>www.example.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.example.com
Address:10.3.210.1
```

From the above, the IP address of "www.example.com" on the FortiBalancer1 will be returned every time, because primary is set to Beijing and the member of Beijing site is FortiBalancer1.

Then, set 10.3.210.1 on the FortiBalancer1 to DOWN. The result is as follows:

```
>www.example.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.example.com
Address: 20.3.220.1

>www.example.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.example.com
Address: 10.3.220.1
```

At this time, the result is the IP address of "www.example.com" configured on the FortiBalancer2, because this is configured on the FortiBalancer2 of standby Tianjin site.

14.3.3 Configuration Example Based on Topology 2 via CLI

14.3.3.1 Configuring SLB

SLB configuration is the same for Topology 1 and Topology 2.

FortiBalancer1

➢ Step 1 Configure a real server

```
FortiBalancer(config)#slb real http "rs1" 10.3.200.110 8080 1000 tcp 1 1
```

➢ Step 2 Configure multiple virtual servers

```
FortiBalancer(config)#slb virtual http "vs1" 10.3.210.1 80
FortiBalancer(config)#slb virtual http "vs2" 10.3.210.2 80
FortiBalancer(config)#slb virtual http "vs3" 10.3.210.3 80
```

➢ Step 3 Configure SLB policy

```
FortiBalancer(config)#slb policy static "vs1" "rs1"
FortiBalancer(config)#slb policy static "vs2" "rs1"
FortiBalancer(config)#slb policy static "vs3" "rs1"
```

FortiBalancer2

➢ Step 1 Configure a real server

```
FortiBalancer(config)#slb real http "rs1" 10.3.200.110 8080 1000 tcp 1 1
```

➢ Step 2 Configure multiple virtual servers

```
FortiBalancer(config)#slb virtual http “vs1” 10.3.220.1 80
FortiBalancer(config)#slb virtual http “vs2” 10.3.220.2 80
FortiBalancer(config)#slb virtual http “vs3” 10.3.220.3 80
```

➢ Step 3 Configure SLB policy

```
FortiBalancer(config)#slb policy static “vs1” “rs1”
FortiBalancer(config)#slb policy static “vs2” “rs1”
FortiBalancer(config)#slb policy static “vs3” “rs1”
```

FortiBalancer3

➢ Step 1 Configure a real server

```
FortiBalancer(config)#slb real http “rs1” 10.3.200.110 8080 1000 tcp 1 1
```

➢ Step 2 Configure multiple virtual servers

```
FortiBalancer(config)#slb virtual http “vs1” 10.3.230.1 80
FortiBalancer(config)#slb virtual http “vs2” 10.3.230.2 80
FortiBalancer(config)#slb virtual http “vs3” 10.3.230.3 80
```

➢ Step 3 Configure SLB policy

```
FortiBalancer(config)#slb policy static “vs1” “rs1”
FortiBalancer(config)#slb policy static “vs2” “rs1”
FortiBalancer(config)#slb policy static “vs3” “rs1”
```

14.3.3.2 Configuring LLB

The “**llb dns host/ttl**” command does not need to be executed for FortiBalancer1 on Topology 2. LLB configurations for FortiBalancer2 and FortiBalancer3 are the same on Topology 1 and Topology 2.

FortiBalancer2

➢ Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned three IP addresses here.

```
FortiBalancer(config)#llb dns host “www.a.com” 10.3.220.1
FortiBalancer(config)#llb dns host “www.a.com” 10.3.220.2
FortiBalancer(config)#llb dns host “www.a.com” 10.3.220.3

FortiBalancer(config)#llb dns host “www.b.com” 10.3.220.1
FortiBalancer(config)#llb dns host “www.b.com” 10.3.220.2
FortiBalancer(config)#llb dns host “www.b.com” 10.3.220.3

FortiBalancer(config)#llb dns host “*.c.com” 10.3.220.1
FortiBalancer(config)#llb dns host “*.c.com” 10.3.220.2
FortiBalancer(config)#llb dns host “*.c.com” 10.3.220.3
```

➢ Step 2 Configure LLB DNS TTL (Time to Live)

```
FortiBalancer(config)#llb dns ttl “www.a.com” 60
FortiBalancer(config)#llb dns ttl “www.b.com” 60
FortiBalancer(config)#llb dns ttl “*.c.com” 60
```

FortiBalancer3

➢ Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned three IP addresses here.

```
FortiBalancer(config)#llb dns host “www.a.com” 10.3.230.1
FortiBalancer(config)#llb dns host “www.a.com” 10.3.230.2
FortiBalancer(config)#llb dns host “www.a.com” 10.3.230.3

FortiBalancer(config)#llb dns host “www.b.com” 10.3.230.1
FortiBalancer(config)#llb dns host “www.b.com” 10.3.230.2
FortiBalancer(config)#llb dns host “www.b.com” 10.3.230.3

FortiBalancer(config)#llb dns host “*.c.com” 10.3.230.1
FortiBalancer(config)#llb dns host “*.c.com” 10.3.230.2
FortiBalancer(config)#llb dns host “*.c.com” 10.3.230.3
```

➢ Step 2 Configure LLB DNS TTL (Time to Live)

```
FortiBalancer(config)#llb dns ttl “www.a.com” 60
FortiBalancer(config)#llb dns ttl “www.b.com” 60
FortiBalancer(config)#llb dns ttl “*.c.com” 60
```

14.3.3.3 Configuring Basic SDNS

The basic SDNS configurations on Topology 2 are different from these configurations on Topology 1. Here, FortiBalancer1 needs to be configured as “dns” while FortiBalancer2 and FortiBalancer3 are configured as “proxy”.

FortiBalancer1

➢ Step 1 Enable SDNS

```
FortiBalancer(config)#sdns on
```

➢ Step 2 Configure SDNS members

```
FortiBalancer(config)#sdns member attribute FortiBalancer1 10.3.200.1 5888 dns
FortiBalancer(config)#sdns member attribute FortiBalancer2 10.3.200.2 5888 proxy
FortiBalancer(config)#sdns member attribute FortiBalancer3 10.3.200.3 5888 proxy
```

➢ Step 3 Configure FortiBalancer1 as a local member

```
FortiBalancer(config)#sdns member local FortiBalancer1
```

FortiBalancer2

➢ Step 1 Enable SDNS

```
FortiBalancer(config)#sdns on
```

➢ Step 2 Configure SDNS members

```
FortiBalancer(config)#sdns member attribute FortiBalancer1 10.3.200.1 5888 dns
FortiBalancer(config)#sdns member attribute FortiBalancer2 10.3.200.2 5888 proxy
FortiBalancer(config)#sdns member attribute FortiBalancer3 10.3.200.3 5888 proxy
```

➢ Step 3 Configure FortiBalancer2 as a local member

```
FortiBalancer(config)#sdns member local FortiBalancer2
```

FortiBalancer3

➢ Step 1 Enable SDNS

```
FortiBalancer(config)#sdns on
```

➢ Step 2 Configure SDNS members

```
FortiBalancer(config)#sdns member attribute FortiBalancer1 10.3.200.1 5888 dns
FortiBalancer(config)#sdns member attribute FortiBalancer2 10.3.200.2 5888 proxy
FortiBalancer(config)#sdns member attribute FortiBalancer3 10.3.200.3 5888 proxy
```

➢ Step 3 Configure FortiBalancer3 as a local member

```
FortiBalancer(config)#sdns member local FortiBalancer3
```

14.3.3.4 Configuring Host Method

region

The logical architecture related to SDNS site/region/pool in this example should be introduced firstly.

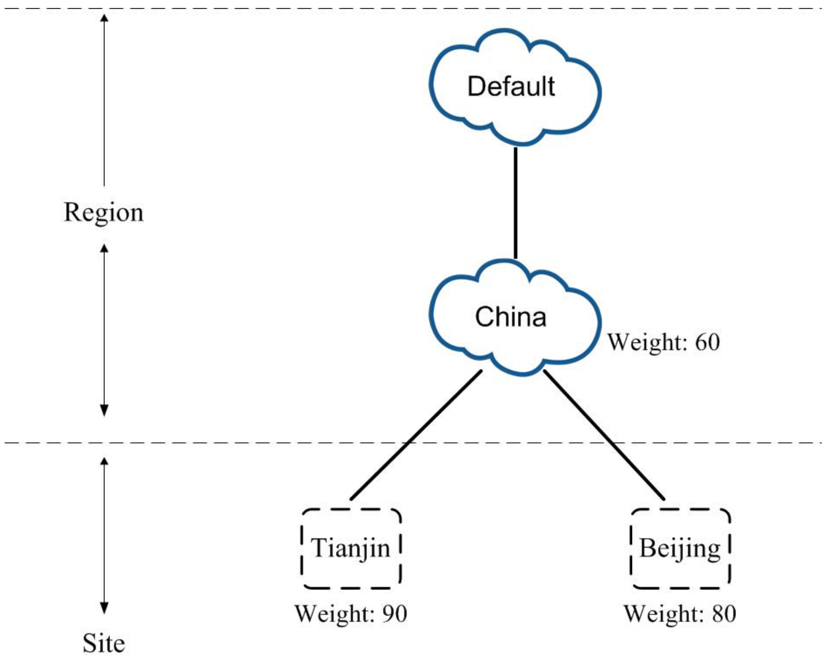


Figure 14-9 SDNS Region Method

In the section Configuring Basic SDNS, FortiBalancer1 needs to be configured as "dns", while FortiBalancer2 and FortiBalancer3 need to be configured as "proxy".

FortiBalancer1

➢ Step 1 Create two sites: Beijing and Tianjin

```
FortiBalancer(config)#sdns site location beijing 90
FortiBalancer(config)#sdns site location tianjin 80
```

➢ Step 2 Add the members into the sites

```
FortiBalancer(config)#sdns site member beijing FortiBalancer2
FortiBalancer(config)#sdns site member tianjin FortiBalancer3
```

➢ Step 3 Create two regions: China and Default

```
FortiBalancer(config)#sdns region location china 60
FortiBalancer(config)#sdns region location default 30
```

- Step 4 Add the region/site into the region

```
FortiBalancer(config)#sdns region division china beijing
FortiBalancer(config)#sdns region division china Tianjin
FortiBalancer(config)#sdns region division default china
```

- Step 5 Create a pool (www.b.com-beijing) and configure its IP addresses

```
FortiBalancer(config)#sdns pool method “www.b.com” beijing rr 2
FortiBalancer(config)#sdns pool ip “www.b.com” beijing 10.3.220.1 5
FortiBalancer(config)#sdns pool ip “www.b.com” beijing 10.3.220.2 5
FortiBalancer(config)#sdns pool ip “www.b.com” beijing 10.3.220.3 5
```

- Step 6 Create a pool (www.b.com-tianjin) and configure its IP addresses

```
FortiBalancer(config)#sdns pool method “www.b.com” tianjin ipo 1
FortiBalancer(config)#sdns pool ip “www.b.com” tianjin 10.3.230.1 6
FortiBalancer(config)#sdns pool ip “www.b.com” tianjin 10.3.230.2 5
FortiBalancer(config)#sdns pool ip “www.b.com” tianjin 10.3.230.3 4
```

- Step 7 Create a pool (www.b.com-china) and configure its IP addresses

```
FortiBalancer(config)#sdns pool method “www.b.com” china pi 2
FortiBalancer(config)#sdns pool ip “www.b.com” china 10.3.220.1 3
FortiBalancer(config)#sdns pool ip “www.b.com” china 10.3.230.1 3
FortiBalancer(config)#sdns persistent timeout 12
```

- Step 8 Create a pool (www.b.com-default) and configure its IP addresses

```
FortiBalancer(config)#sdns pool method “www.b.com” default rr 2
FortiBalancer(config)#sdns pool ip “www.b.com” default 10.3.220.4 5
```

- Step 9 Create a pool rule (rule1-china) and configure its IP addresses

```
FortiBalancer(config)#sdns pool rule “rule1” china rr 3
FortiBalancer(config)#sdns pool ip “rule1” china 10.3.220.1 10
```

- Step 10 Set the rule1 to host www.a.com

```
FortiBalancer(config)#sdns host rule “rule1” www.a.com
```

- Step 11 Set the host method to “region”

```
FortiBalancer(config)#sdns host method “www.a.com” region
FortiBalancer(config)#sdns host method “www.b.com” region
```

- Step 12 Set the SDNS proximity

```
FortiBalancer(config)#sdns proximity 10.3.200.107 255.255.255.255 beijing
FortiBalancer(config)#sdns proximity 10.3.50.7 255.255.255.255 tianjin
```

Request for resolving “www.b.com” on two clients (their IP addresses are respectively 10.3.50.7 and 10.3.200.107）by using nslookup of Windows.

The client whose IP address is 10.3.200.107 will set local DNS to 10.3.200.1

```
> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.220.1, 10.3.220.2
```

```
> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.220.2, 10.3.220.3

> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.220.3, 10.3.220.1
```

As is obvious from the above, the packet whose corresponding source IP address is configured as10.3.200.107 in SDNS proximity will be located to Beijing pool. So the IP address of "www.b.com-beijing" pool will be returned. Because the returned IP address' number of the pool is assigned to 2, every time two IP addresses will be returned in round robin.

The client whose IP address is 10.3.50.7 sets local DNS to 10.3.200.1.

```
region:
> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.230.1

> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.230.1

> www.b.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.b.com
Addresses: 10.3.230.1
```

As is obvious from the above, the packet whose corresponding source IP address is set to 10.3.50.7 in SDNS proximity will be located to Tianjin pool. So the IP address of "www.b.com" with the highest priority in Tianjin pool will be returned. Because the returned IP address' number of the pool is assigned to 1, every time the IP address with the highest priority will be returned.

14.3.3.5 Configuring SDNS Bandwidth

If we want to manage the SDNS bandwidth, we need to go on the configuration of bandwidth for "region" host method.

Set the bandwidth of "region", "site" and "member"

- Step 1 Set the "china region" bandwidth limit to 10M and the statistics mode is inout

```
FortiBalancer(config)#sdns bandwidth region china 1 10
```

➢ Step 2 Set the “beijing site” bandwidth limit to 2M, and the statistics mode is inout

```
FortiBalancer(config)#sdns bandwidth site beijing 3 2
```

➢ Step 3 Set the “tianjin site” bandwidth limit to 1M, and the statistics mode is in

```
FortiBalancer(config)#sdns bandwidth site tianjin 2 1
```

➢ Step 4 Set the FortiBalancer1 member bandwidth limit to 1M, and the statistics mode is inout

```
FortiBalancer(config)#sdns bandwidth member FortiBalancer2 1 1
```

➢ Step 5 Set the FortiBalancer2 member bandwidth limit to 1M, and the statistics mode is inout

```
FortiBalancer(config)#sdns bandwidth member FortiBalancer3 1 1
```

Access “www.b.com” from 10.3.200.107 (DNS server is set to 10.3.200.1). The traffic is displayed as follows:

```
FortiBalancer1(config)#show sdns band
Name          Site/Region ID          Limit        Usage        Mode         Status
china             3                   10000000     1231638      1

Region: china
www.b.com         3                   10000000     1254880      8

default           4                   -1           0            0

Region: default

beijing           1                   2000000      615906       3

Site: beijing
www.b.com         1                   5000000      0            7

tianjin           2                   1000000      666          2
Site: tianjin

FortiBalancer3                            1000000      901          1
FortiBalancer2                            1000000      1230737      1            Full
FortiBalancer1                            -1           0            0

The bandwidth of vips:
```

14.3.4 Configuration Example for SDNS DPS

We should configure a DPS master and a DPS slave, and enable DPS detectors for sites.

Note: SDNS DPS master generates SDNS DPS type 1 packets, sends them to SDNS DPS Detector and receives SDNS DNS packet type 2 from SDNS DPS Detector. SDNS DPS slave receives SDNS DPS type 2 packets from SDNS DPS Detector.

FortiBalancer1 (DPS master)

➢ Step 1 Basic SDNS configuration

```
FortiBalancer(config)#sdns on Check
FortiBalancer(config)#sdns interval heartbeat 2
FortiBalancer(config)#sdns site location beijing 0
FortiBalancer(config)#sdns site location shanghai 0
FortiBalancer(config)#sdns interval report 30
```

- Step 2 SDNS DPS master configuration

```
FortiBalancer(config)#sdns dps interval send 15
FortiBalancer(config)#sdns dps interval query 15
FortiBalancer(config)#sdns dps history 9000
FortiBalancer(config)#sdns dps method hops
FortiBalancer(config)#sdns dps detector beijing 10.3.17.19 44544 15
FortiBalancer(config)#sdns dps detector shanghai 172.16.63.204 44544 15
FortiBalancer(config)#sdns dps member 10.3.17.100
FortiBalancer(config)#sdns dps member 10.3.17.20
FortiBalancer(config)#sdns dps on
FortiBalancer(config)#sdns dps master on 55456
FortiBalancer(config)#sdns statistics on localdns
```

FortiBalancer2 (DPS slave)

- Step 1 Basic SDNS configuration

```
FortiBalancer(config)#sdns on Check
FortiBalancer(config)#sdns interval heartbeat 2
FortiBalancer(config)#sdns site location beijing 0
FortiBalancer(config)#sdns site location shanghai 0
FortiBalancer(config)#sdns interval report 30
```

- Step 2 SDNS DPS slave configuration

```
FortiBalancer(config)#sdns dps interval send 15
FortiBalancer(config)#sdns dps interval query 15
FortiBalancer(config)#sdns dps history 9000
FortiBalancer(config)#sdns dps method rtt
FortiBalancer(config)#sdns dps detector beijing 10.3.17.19 44544 15
FortiBalancer(config)#sdns dps detector shanghai 172.16.63.204 44544 15
FortiBalancer(config)#sdns dps on
FortiBalancer(config)#sdns dps master off
```

Enabling DPS Detectors for Sites

Assume that the “beijing” site (10.3.17.19) uses the FortiBalancer appliance as its DPS detector and the “shanghai” site uses the proxDetector installed on a server that runs the Linux operating system as its DPS detector. The following configuration example describes how to enables the DPS detectors for the “beijing” and “shanghai” sites.

- Step 1 Enable the DPS detector for the “beijing” site by executing the following commands:

```
FortiBalancer(config)#sdns on
FortiBalancer(config)#sdns dps localdetector "det_bj" 0.0.0.0 "all" 53455 44544 30
```

- Step 2 Enable the DPS detector for the “shanghai” site by executing the following command on the Linux server as a root user:

```
./proxDetector -a 0.0.0.0 -p 53455 -P 44544 -t 30
```

Chapter 15 Access Control

15.1 WebWall

15.1.1 Overview

The WebWall functionality of the FortiBalancer appliance allows you to create permit/deny rules to filter packets passing through your network infrastructure. The WebWall supports the filtering of TCP, UDP and ICMP packets that are using the IPv4 or IPv6 address. To use access lists you will define these "permit" and "deny" rules and apply them to access groups. Once the access lists are configured, you may apply or bind the group to an interface within the network.

The steps for basic WebWall configurations are explained in this section, along with some advanced features and general knowledge of how WebWall works. For the OS, the WebWall feature can independently control each interface.

WebWall permits TCP and UDP health check traffic, but cannot permit ICMP health check traffic automatically.

15.1.2 Understanding WebWall

WebWall is a full-fledged stateful Firewall. It bridges the gap between speed and security. The FortiBalancer appliance houses and integrates the WebWall feature into a single platform, along with many of other features such as Layer 4-7 load balancing, caching, SSL acceleration, authentication and authorization.

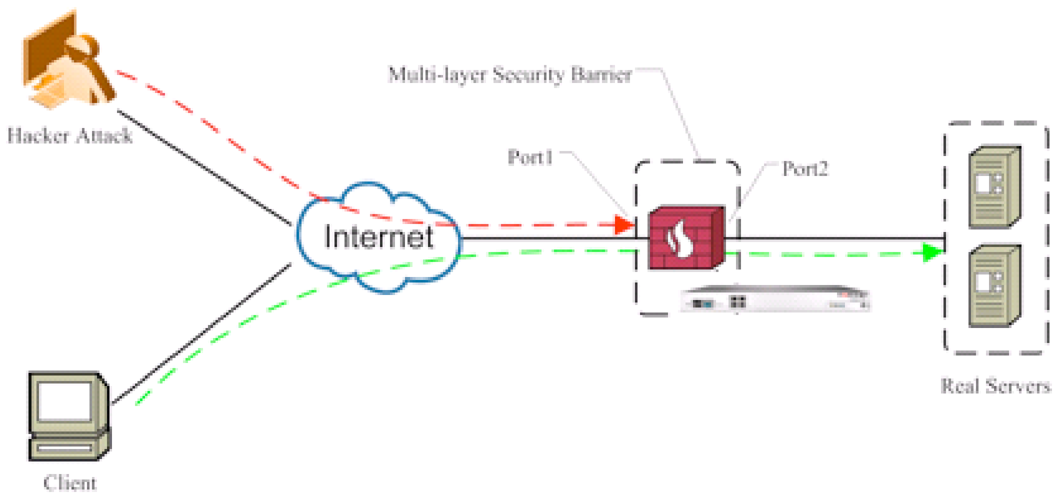


Figure 15-1 WebWall

WebWall contains several security mechanisms to protect Web servers from attack, including:

- ACL (Access Control List) filtering
- Protection against Syn-Flood, Fragmentation and DoS (Denial Of Service) attacks
- Stateful packet inspection
- Single packet attack prevention

ACL Filtering provides tight control over who may and may not enter the network by utilizing FortiBalancer's ultra-fast rules engine. WebWall access control list filtering mechanism ensures virtually no performance loss with up to 1,000 ACL rules, while never consuming more than one percent of OS capability.

In addition to ACL filtering, the WebWall provides stateful packet inspection and protects against Syn-Flood, fragmentation, DoS and single packet attacks.

The WebWall is a default-deny firewall. Default-Deny refers to the notion that if you do not have any permit rules in your access control lists, no packets will be allowed to pass through the appliance. During the initial installation of the box it might be helpful to leave the WebWall in the off or disengaged state until your total configuration is complete.

Note: By default the WebWall is turned off. The WebWall function will remain disabled until it is activated via the "webwall on" command.

15.1.3 WebWall Configuration

15.1.3.1 Configuration Guidelines

Let's start with the basic step for configuring the WebWall. To better assist you with configuration strategies that maximize the power of the FortiBalancer appliance, please take a moment to familiarize yourself with basic network architecture.

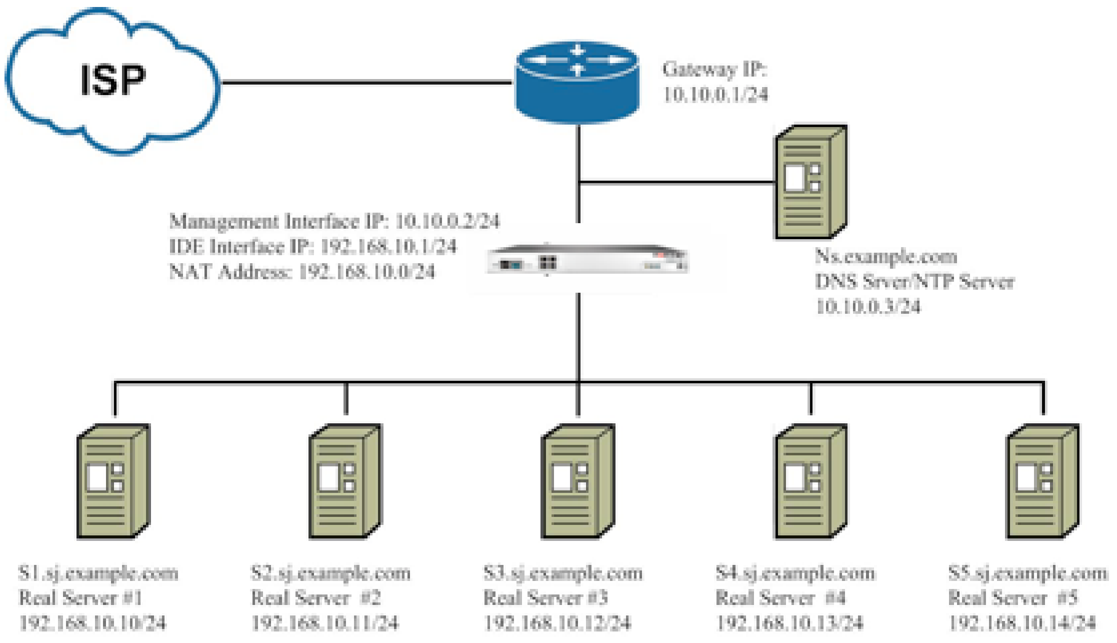


Figure 15-2 WebWall Configuration

Then we must define what we want to deny and permit. Since "example.com" is a relatively small site, let's begin with the following:

- Permit port 80 to our VIP (10.10.0.10).
- Permit port 22 to the Management IP of the FortiBalancer appliance (for SSH access).
- Permit port 8888 to the Management IP of the FortiBalancer appliance for web UI access.
- Deny network 10.10.20.0/255.255.255.0, since that network has been abusing its privileges.
- Allow all inside hosts to ping the IP address of the interface "port2" (inside interface).

Initially we will define our access groups as follows:

- 50 All miscellaneous rules
- 100 All Management IP related rules
- 150 All VIP (Virtual IP) related rules

Table 15-1 General Settings of WebWall

Operation	Command
Configure access group	**accessgroup** *<accesslist_id> <interface>*
Configure ACL rules	**accesslist permit icmp echoreply** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>* **accesslist permit icmp echorequest** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>* **accesslist permit tcp** *<source_ip> <source_mask|source_prefix> <source_port> <destination_ip> <destination_mask|destination_prefix> <destination_port> <accesslist_id>* **accesslist permit udp** *<source_ip> <source_mask|source_prefix> <source_port> <destination_ip> <destination_mask|destination_prefix> <destination_port> <accesslist_id>* **accesslist permit esp** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>* **accesslist permit ah** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>* **accesslist deny icmp echoreply** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>* **accesslist deny icmp echorequest** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>* **accesslist deny tcp** *<source_ip> <source_mask|source_prefix> <source_port> <destination_ip> <destination_mask|destination_prefix> <destination_port> <accesslist_id>* **accesslist deny udp** *<source_ip> <source_mask|source_prefix> <source_port> <destination_ip> <destination_mask|destination_prefix> <destination_port> <accesslist_id>* **accesslist deny esp** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>* **accesslist deny ah** *<source_ip> <source_mask|source_prefix> <destination_ip> <destination_mask|destination_prefix> <accesslist_id>*
Enable/Disable WebWall	**webwall** *<interface>* **on** *[mode]* **webwall** *<interface >* **off**
View WebWall configurations	**show interface** **show accesslist** **show accessgroup**

15.1.3.2 Configuration Example via CLI

15.1.3.2.1 Configuring Access Groups

We may define any number of access groups and apply multiple groups to a designated interface via CLI. Pertaining to our example model, the command should be executed as such:

```
FortiBalancer(config)#accessgroup 100 port1
FortiBalancer(config)#accessgroup 150 port1
FortiBalancer(config)#accessgroup 50 port1
```

You might have noticed that we also have specified what interfaces these access groups will be applied to.

15.1.3.2.2 Configuring ACL Rules

Now we define the "permit" and "deny" rules based on these assumptions.

The first entry allows a single host with IP 10.10.10.30 to connect to the server using port 22:

```
FortiBalancer(config)#accesslist permit tcp 10.10.10.30 255.255.255.255 0 10.10.10.10
255.255.255.255 22 100
```

The second entry allows a C class subnet to connect to the server via port 8888.

```
FortiBalancer(config)#accesslist permit tcp 10.10.10.0 255.255.255.0 0 10.10.10.10
255.255.255.255 8888 100
```

The third allows any host to connect to the server using port 80.

```
FortiBalancer(config)#accesslist permit tcp 0.0.0.0 0.0.0.0 0 10.10.10.20 255.255.255.255 80
150
```

The first three rules are fairly straightforward, and they permit all TCP traffic to the destination IP/port specified and are tied to the access group (via the last argument to the command).

With the fourth entry, we are excluding one host from gaining access through the subnet. It is in access group 50 since it does not allow access to a specific destination IP. Logically the deny rule could fit into both access group 100 and 150, so for administrative ease we will make another group.

```
FortiBalancer(config)#accesslist deny tcp 10.10.10.33 255.255.255.255 0 10.10.10.10
255.255.255.255 0 50
```

The last two rules allow the inside hosts on the network to ping the "port2" interface when the WebWall function is on.

```
FortiBalancer(config)#accesslist permit icmp echorequest 192.168.10.0 255.255.255.0
192.168.10.1 255.255.255.255 50
FortiBalancer(config)#accesslist permit icmp echoreply 192.168.10.1 255.255.255.255
192.168.10.0 255.255.255.0 50
```

Note: The IP address is not an IP on the FortiBalancer appliance. It is the IP of the default gateway.

The priority of the command "**accesslist deny**" is higher than "**accesslist permit**". If we configure "permit" and "deny" rules for the port 22 to the Management IP of the FortiBalancer appliance (for SSH access) at the same time as follows:

```
FortiBalancer(config)#accesslist permit tcp 10.10.10.30 255.255.255.255 0 10.10.10.10
255.255.255.255 22 100
FortiBalancer(config)#accesslist deny tcp 10.10.10.30 255.255.255.255 0 10.10.10.10
255.255.255.255 22 100
```

When the administrators attempt to access the FortiBalancer appliance via the management IP through SSH, the access will be denied.

15.1.3.2.3 Configuring WebWall

At last once you complete the configuration of the other features of the FortiBalancer appliance, and you should turn the WebWall feature back on by issuing the command:

```
FortiBalancer(config)#webwall port2 on
FortiBalancer(config)#webwall port1 on
```

Notes:

1. You should exercise with caution when adjusting the WebWall rules. It is possible to deny yourself from accessing the appliance if you are logged in remotely through SSH or the web UI and your session can be interrupted before configuration is completed.

2. If you configure the DNS servers and have WebWall turned on for the destination interface through which the DNS requests/replies go, you need to add the corresponding accesslist rules to allow that traffic.

3. If WebWall is turned on for the interface for which the "synconfig" command uses to synchronize with peer(s), you will need to add the corresponding accesslist rules to allow that traffic to come in through SSH port 22 on the Fortinet machines (FortiBalancer appliance and the sync peers).

15.1.3.3 Verification and Troubleshooting of the WebWall

After adding all the rules it is helpful to display the current lists and groups. To do this, employ the following commands.

```
FortiBalancer(config)#show accesslist
accesslist deny tcp 10.10.10.33 255.255.255.255 0 10.10.10.10 255.255.255.255 0 50
accesslist permit tcp 10.10.10.30 255.255.255.255 0 10.10.10.10 255.255.255.255 22 100
accesslist permit tcp 10.10.10.0 255.255.255.0 10.10.10.10 255.255.255.255 8888 100
accesslist permit tcp 0.0.0.0 0.0.0.0 0 10.10.10.20 255.255.255.255 80 150
accesslist permit icmp echorequest 10.10.10.0 255.255.255.0 10.10.10.10 255.255.255.255 50
accesslist permit icmp echoreply 0.0.0.0 0.0.0.0 10.10.10.10 255.255.255.255 50

FortiBalancer(config)#show accessgroup
accessgroup 50 port1
accessgroup 100 port1
accessgroup 150 port1
```

If you run into problems with access lists, keep your configurations simple. With multiple access groups, you can apply them once at a time and see which access list is causing problems. Of course you can turn the WebWall completely off to determine if the WebWall itself is indeed causing the problem.

To check the status of the firewall use the "**show interface**" command:

```
FortiBalancer(config)#show interface
port1(port1): flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 1500
        inet 10.3.20.100 netmask 0xffff0000 broadcast 10.3.255.255
        inet 10.3.20.56 netmask 0xffffffff broadcast 10.3.20.56
        ether 00:30:48:82:81:7a
        media: autoselect (100baseTX <full-duplex>)
        status: active
        webwall status: OFF
        Hardware is i82547gi
        Input queue: 435/512 (size/max)
                total: 19376 packets, good: 19376 packets, 2053879 bytes
                broadcasts: 19130, multicasts: 2
                11317 64 bytes, 4282 65-127 bytes,3242 128-255 bytes
                522 255-511 bytes,13 512-1023 bytes,0 1024-1522 bytes
                0 input errors
                0 runts, 0 giants, 0 Jabbers, 0 CRCs
                0 Flow Control, 0 Fragments, 0 Receive errors
                0 Driver dropped, 0 Frame, 0 Lengths, 0 No Buffers
```

```
                    0 overruns, Carrier extension errors: 0
          Output queue: 0/512 (size/max)
                    total: 18444 packets, good:    18444 packets, 7182692 bytes
                    broadcasts: 17, multicasts: 0
                    48 64 bytes, 6018 65-127 bytes,7512 128-255 bytes
                    785 255-511 bytes,1014 512-1023 bytes,3067 1024-1522 bytes
                    0 output errors
                    0 Collsions, 0 Late collisions, 0 Deferred
                    0 Single Collisions, 0 Multiple Collisions, 0 Excessive collsions
          0 lost carrier, 0 WDT reset
          packet drop (not permit): 0
                    tcp  0          udp  0          icmp  0          ah  0          esp  0
          packet drop (deny): 0
                    tcp  0          udp  0          icmp  0          ah  0          esp  0
          5 minute input rate 2160 bits/sec, 2 packets/sec
          5 minute output rate 80 bits/sec, 0 packets/sec
port2(port2): flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 1500
          inet 10.4.20.100 netmask 0xffff0000 broadcast 10.4.255.255
          ether 00:30:48:82:81:7b
          media: autoselect (100baseTX <full-duplex>)
          status: active
          webwall status: OFF
          Hardware is i82541gi
          Input queue: 71/512 (size/max)
                    total: 38464 packets, good: 38464 packets, 9320519 bytes
                    broadcasts: 18751, multicasts: 2
                    10779 64 bytes, 11545 65-127 bytes,10749 128-255 bytes
                    1305 255-511 bytes,1019 512-1023 bytes,3067 1024-1522 bytes
                    0 input errors
                    0 runts, 0 giants, 0 Jabbers, 0 CRCs
                    0 Flow Control, 0 Fragments, 0 Receive errors
                    0 Driver dropped, 0 Frame, 0 Lengths, 0 No Buffers
                    0 overruns, Carrier extension errors: 0
          Output queue: 0/512 (size/max)
                    total: 2094 packets, good:    2094 packets, 207035 bytes
                    broadcasts: 396, multicasts: 0
                    399 64 bytes, 1681 65-127 bytes,0 128-255 bytes
                    0 255-511 bytes,14 512-1023 bytes,0 1024-1522 bytes
                    0 output errors
                    0 Collsions, 0 Late collisions, 0 Deferred
                    0 Single Collisions, 0 Multiple Collisions, 0 Excessive collsions
          0 lost carrier, 0 WDT reset
          packet drop (not permit): 0
                    tcp  0          udp  0          icmp  0          ah  0          esp  0
          packet drop (deny): 0
                    tcp  0          udp  0          icmp  0          ah  0          esp 0
          5 minute input rate 2336 bits/sec, 3 packets/sec
5 minute output rate 224 bits/sec, 0 packets/sec
```

This command will also show if the interface is up and running and those IP addresses assigned to it. More detailed network information is also included, such as input queue and output queue information.

The following explains the terms and phrases used in the output:

- **Input queue size**: the current occupied input.
- **Input queue max**: the maximum items of input.

- **The numbers of different sizes**: the counts of the packages of each size.
- **Runt**: the number of received frames that have passed address filtering that are less than the minimum size (64 bytes from <Destination Address> through <CRC>, inclusively), and have a valid CRC.
- **Giant**: the number of received frames with valid CRC field that have passed address filtering and are larger than the maximum size.
- **Jabber**: the number of received frames that have passed address filtering that are greater than the maximum size and have a bad CRC. It may be the result of a bad NIC or electronic interfering.
- **CRC**: the number of received packets with alignment errors.
- **Flow Control**: the number of the received, unsupported flow control frames.
- **Fragments**: the number of received frames that have passed address filtering, are less than the minimum size and have a bad CRC.
- **Frame**: the number of received packets with alignment errors (the packet is not an integer number of bytes in length).
- **Lengths**: the number of received length error events.
- **No Buffers**: the number of times that frames are received when there are no available buffers in host memory to store those frames.
- **Overruns**: the number of missed packets. Packets are missed when the received FIFO has insufficient space to store the incoming packets. This can be caused by too few allocated buffers, or insufficient bandwidth on the PCI bus.
- **Carrier extension errors**: the number of received packets where the carrier extension error is signaled across the internal PHY interface.
- **Collisions**: the total number of collisions that are not late collisions as seen by the transmitter.
- **Late collisions**: late collisions are collisions that occur after 64-byte time into the transmission of the packet while working in 10-100 Mb/s data rate, and after 512-byte time into the transmission of the packet while working in the 1000 Mb/s data rate.
- **Deferred**: a deferred event occurs when the transmitter cannot immediately send a packet because the medium is busy or another device is transmitting.
- **Single Collisions**: the number of times that a successfully transmitted packet has encountered only one collision.
- **Multiple Collisions**: the number of times that a successfully transmitted packet has encountered more than one collision but less than 16.
- **Excessive collisions**: the number of times that 16 or more collisions have occurred on a packet.

15.2 Advanced ACL

15.2.1 Overview

The Advanced ACL function enables the administrator to restrict the connections per second (CPS) and concurrent connections (CC) allowed for the clients on a specified subnet when accessing virtual services by configuring ACL rules. This prevents not only the network bandwidth resources from being over-occupied by some clients, but the malicious attacks of large traffic, which ensures the security of intranet resources.

The Advanced ACL function also allows the administrator to configure ACL whitelists to make the clients on the whitelists free from restriction of ACL rules. Furthermore, when accessing the HTTP or HTTPS virtual services for which the insert cookie policy has been used, the clients that carry cookies inserted by the FortiBalancer appliance in the requests are free from the restriction of any ACL rule.

ACL rules and ACL whitelists can be applied not only to an individual virtual service, but also to virtual services globally. The Advanced ACL function supports all TCP-based virtual services.

15.2.1.1 ACL Rule

The ACL rule supports two control modes and two control types.

Control modes:

- "total" mode indicates that all the clients on a subnet will be restricted by the ACL rule as a whole.
- "per-ip" mode indicates that every client on a subnet will be restricted by the ACL rule individually.

Control types:

- "cps" indicates restriction on the CPS.
- "concurrent" indicates restriction on the CC.

The administrator needs to combine the control modes and control types as needed for configuring ACL rules for a subnet. There are four combinations of ACL rules for the same subnet. See the following figure.

Table 15-2 Four Rules for the Same Subnet

Mode / Type	cps	concurrent
total	The total CPS of all the clients on the subnet cannot exceed the specified maximum value.	The total CC of all the clients on the subnet cannot exceed the specified maximum value.
per-ip	The CPS of every client on the subnet cannot exceed the specified maximum value.	The CC of every client on the subnet cannot exceed the specified maximum value.

One or more rules of the preceding combinations are allowed for a subnet on the FortiBalancer appliance. If multiple rules have been configured for a subnet, these rules will work at the same time.

Subnet Nesting

The ACL rule supports subnet nesting. If the IP address of a client hits multiple subnets, the client is subject to the restriction of all the ACL rules of the smallest subnet hit, and of the "total" rules of all the upper subnets of the smallest subnet. The system supports 4-layer subnet nesting at most.

For example:

- 10.8.0.0/16: Both "total" and "per-ip" rules are configured.
- 10.8.0.0/24: Both "total" and "per-ip" rules are configured.

Then, clients on the subnet of 10.8.1.0/24 will be subject to the restriction of:

- 10.8.0.0/24: "total" and "per-ip"
- 10.8.1.0/16: "total" rules

Note:

ACL rules may impact on the system performance.

ACL rules take effect only for connections established after the ACL rules are configured.

15.2.1.2 ACL Whitelist

The administrator can configure ACL whitelists to make the clients on the whitelists free from restriction of ACL rules. When both ACL rules and whitelists are applied to the same virtual service, the whitelists have higher processing priorities than the ACL rules.

If the IP address of a client belongs to the subnet defined in an ACL whitelist, the client will not be subject to the restriction of any ACL rule. However, the CPS and CC of the client will be counted in the total CPS and CC of the subnet it belongs to respectively.

For example:

- Rule: 10.8.1.0/24; total concurrent≤10,000
- Whitelist: 10.8.1.10/32

If the CC on the client whose IP address is 10.8.1.10 is 1000,

Then, the maximum total CC of the subnet 10.8.1.0/24 (excluding 10.8.1.10) is 9000.

15.2.1.3 Application Scope

ACL rules and ACL whitelists can be applied not only to an individual virtual service, but also to virtual services globally. The advanced ACL function supports all TCP-based virtual services.

When accessing a specified virtual service, a client is subject to the restriction of not only the ACL rules applied to this virtual service individually but also the ACL rules applied globally.

For example:

- rule1: 0.0.0.0/0, per-ip concurrent≤1000
- rule2: 0.0.0.0/0, per-ip concurrent≤900
- rule1 is applied to vs1; rule2 is applied globally.

Then, when the client whose IP address is 10.8.1.10 accesses the virtual service “vs1”, the allowed maximum CC is 900.

15.2.1.4 Application of ACL Rules to HTTP-based Virtual Services

For Layer 7 HTTP and HTTPS virtual services with the insert cookie policy used, the FortiBalancer appliance will insert cookies in the responses sent to the clients. The FortiBalancer appliance can identify these clients with the inserted cookies. When a client accesses the virtual service again and carries the cookie inserted by the FortiBalancer appliance in the requests, the client will not be subject to the restriction of any ACL rule. However, the CPS and CC of the client will be counted in the total CPS and CC of the subnet it belongs to respectively.

15.2.2 Advanced ACL Configuration

To complete the Advanced ACL configuration, perform the following steps:

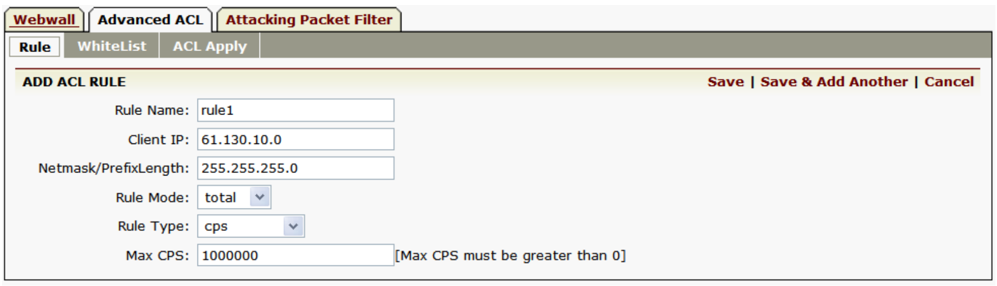
1. Configuring ACL rules
2. (Optional) Configuring ACL whitelists

15.2.2.1 Configuring ACL rules

To configure ACL rules, you need to first add ACL rules and then apply these rules to a specified virtual service or apply them globally.

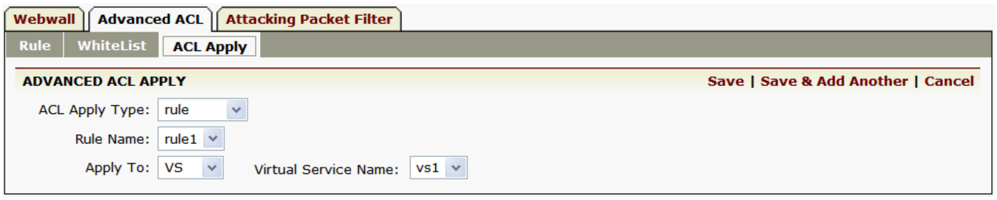
➢ **web UI:**

1. Select **System Configuration > Access Control > Advanced ACL > Rule**. In the **ACL Rule List** area, click the **Add** action link. In the new displayed page, specify the required parameters and click the **Save** action link to save the configuration.



Rule Name:	rule1
Client IP:	61.130.10.0
Netmask/PrefixLength:	255.255.255.0
Rule Mode:	total
Rule Type:	cps
Max CPS:	1000000 [Max CPS must be greater than 0]

2. Select the **ACL Apply** tab. In the **ACL Apply List** table, click the **Apply** action link. In the new displayed page, apply the ACL rule to the specified virtual service or apply it globally. Then, click the **Save** action link to save the configuration.



ACL Apply Type:	rule
Rule Name:	rule1
Apply To:	VS
Virtual Service Name:	vs1

➢ **CLI:**

1. Execute the following command to add ACL rules:

```
acl rule <rule_name> <client_ip> {netmask|prefix} <acl_mode> <acl_type>
<max_limit>
```

For example:

```
FortiBalancer(config)#acl rule rule1 61.130.10.0 255.255.255.0 total cps 1000000
FortiBalancer(config)#acl rule rule2 61.130.10.0 255.255.255.0 total concurrent
50000
FortiBalancer(config)#acl rule rule3 61.130.10.0 255.255.255.0 per-ip cps 10000
FortiBalancer(config)#acl rule rule4 61.130.10.0 255.255.255.0 per-ip concurrent 500
```

2. Execute the following command to apply ACL rules to virtual services:

```
acl apply rule virtual <rule_name> <vs_name>
```

For example:

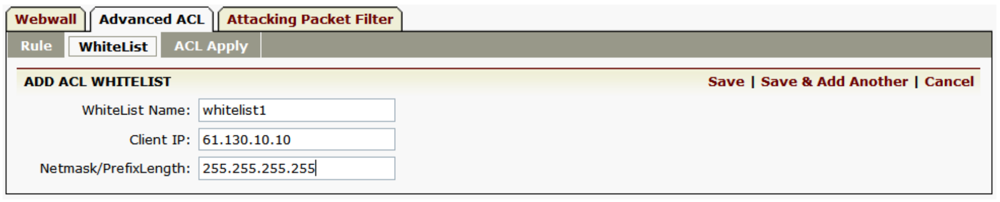
```
FortiBalancer(config)#acl apply rule virtual rule1 tcp_vs1
FortiBalancer(config)#acl apply rule virtual rule3 tcp_vs1
FortiBalancer(config)#acl apply rule virtual rule2 http_vs1
FortiBalancer(config)#acl apply rule virtual rule4 http_vs1
```

15.2.2.2 Configuring ACL Whitelists

To configure ACL whitelists, you need to first add ACL whitelists and then apply these whitelists to a specified virtual service or apply them globally.

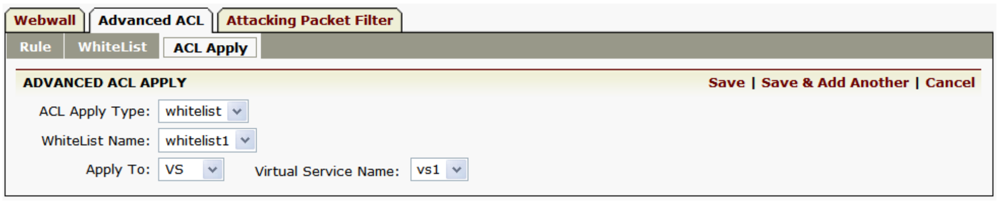
➢ **web UI:**

1. Select **System Configuration > Access Control > Advanced ACL > Whitelist**. In the **ACL Whitelist List** area, click the **Add** action link. In the displayed new page, specify the required parameters and click the **Save** action link to save the configuration.



WhiteList Name: whitelist1
Client IP: 61.130.10.10
Netmask/PrefixLength: 255.255.255.255

2. Select the **ACL Apply** tab. In the **ACL Apply List** table, click the **Apply** action link. In the new displayed page, apply the ACL whitelist to the specified virtual service or apply it globally. Then, click the **Save** action link to save the configuration.



ACL Apply Type: whitelist
WhiteList Name: whitelist1
Apply To: VS
Virtual Service Name: vs1

➢ **CLI:**

1. Execute the following command to add ACL whitelists:

```
acl whitelist <whitelist_name> <client_ip> {netmask|prefix}
```

For example:

```
FortiBalancer(config)#acl whitelist whitelist1 61.130.10.10 255.255.255.255
```

2. Execute the following command to apply ACL whitelists to virtual services:

```
acl apply whitelist virtual <whitelist_name> <vs_name>
```

For example:

```
FortiBalancer(config)#acl apply whitelist virtual whitelist1 tcp_vs1
FortiBalancer(config)#acl apply whitelist virtual whitelist1 http_vs1
```

15.2.2.3 Configuration Results

After the preceding configurations are completed, the FortiBalancer appliance will:

➢ **When clients are accessing the virtual service "tcp_vs1":**

- Restrict the maximum total CPS of all clients on the subnet 61.130.10.0/24 to 1,000,000.
- Restrict the maximum CPS of every client on the subnet 61.130.10.0/24 to 10,000.
- Not restrict the maximum CPS of the client whose IP address is 61.130.10.10.

➢ **When clients are accessing the virtual service "http_vs1":**

- Restrict the maximum total CC of all clients on the subnet 61.130.10.0/24 to 50,000.
- Restrict the maximum CC of every client on the subnet 61.130.10.0/24 to 500.
- Not restrict the maximum CC of the client whose IP address is 61.130.10.10.

Chapter 16 Advanced IPv6 Configuration

16.1 Overview

As the IPv4 addresses exhaust, how to transit from the IPv4 network to the IPv6 network becomes a challenge for many enterprises and organizations.

The FortiBalancer appliance provides comprehensive support for IPv6 to help enterprises and organizations with the IPv4-to-IPv6 transition without any business interruption. With the IPv4/IPv6 dual stack support on FortiBalancer, the IPv4 resources can be delivered to the IPv6 users, and vice versa. As a result, the IPv4-based and IPv6-based networks can be easily interconnected and intercommunicated. What's more, the FortiBalancer appliance in the IPv6 network can achieve the same level of secure and efficient application delivery as it does in the IPv4 network.

This chapter will introduce functions and configurations about IPv6 SLB, DNS64/NAT64, DNS46/NAT46, IPv6 NAT and NDP.

16.2 IPv6 SLB

16.2.1 Overview

FortiBalancer provides comprehensive IPv6 support for the SLB module. For the service layer, SLB services from Layer 2 to 7 all support IPv6. For the service type, all of the service types except RDP, SIPUDP and SIPTCP support IPv6. For the SLB method and policy, all of the SLB methods except SNMP and all of the SLB policies support IPv6. Additionally, for the three SLB deployment modes, IPv6 is supported as follows:

- IPv6 to IPv4 (supported by the reverse proxy mode)
- IPv4 to IPv6 (supported by the reverse proxy mode)
- IPv6 to IPv6 (supported by the reverse proxy mode, transparent mode and triangle mode)

Among which, the reverse proxy mode can work in the IPv4/IPv6 co-existing network environment, as well as in the IPv4 only or IPv6 only network environment. The transparent mode and triangle mode can only work in the IPv4 only or IPv6 only network environment.

The following figure shows the topology of the "IPv6 to IPv4" deployment method.

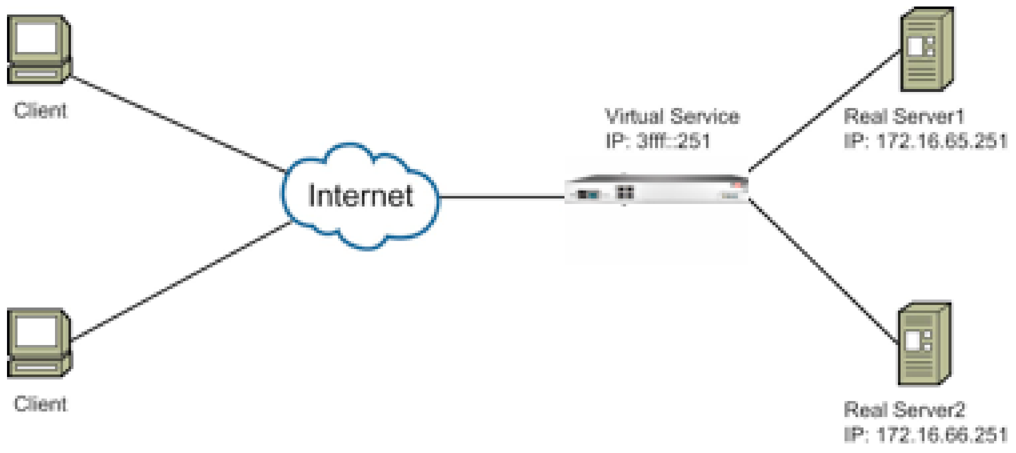


Figure 16-1 "IPv6 to IPv4" of the SLB topology

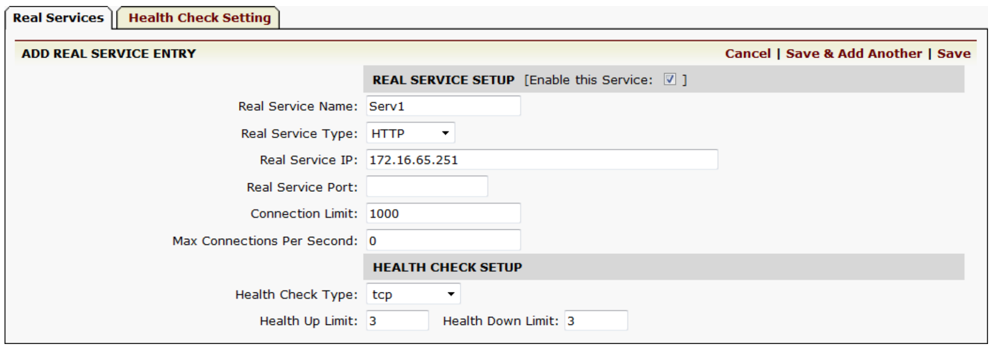
Likewise, the "IPv4 to IPv6" deployment method can be adopted by configuring an IPv4 address for the virtual service and an IPv6 address for the real server.

For the "IPv6 to IPv6" deployment method, both the virtual service and real server should be configured with IPv6 addresses.

16.2.2 Configuration Example

- **web UI:**

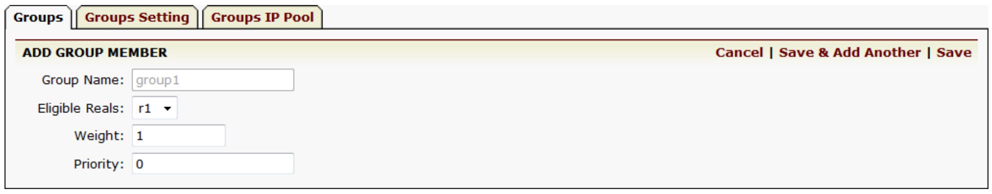
1. Select **Server Load Balance > Real Services > Real Services**. Click the **Add Real Service Entry** action link, specify parameters in the **Add Real Service Entry** area and click the **Save** action link.



Real Services
Health Check Setting
ADD REAL SERVICE ENTRY
Cancel | Save & Add Another | Save
REAL SERVICE SETUP [Enable this Service: ☑]
HEALTH CHECK SETUP

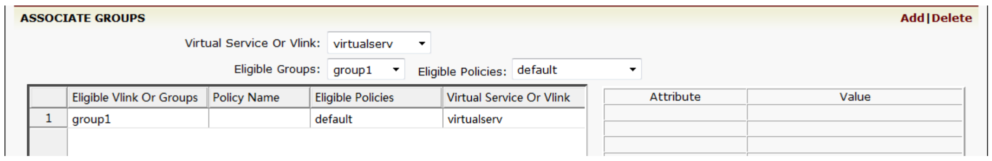
Real Service Name:	Serv1
Real Service Type:	HTTP
Real Service IP:	172.16.65.251
Real Service Port:	
Connection Limit:	1000
Max Connections Per Second:	0
Health Check Type:	tcp
Health Up Limit:	3
Health Down Limit:	3

2. Select **Server Load Balance > Groups > Groups**. Specify parameters in the **Add Group** area and click the **Add** action link. In the **Groups List** table, double-click a group. Click the **Add** action link in the **Group Members** area of the new popup page. Specify parameters in the **Add Group Member** area and click the **Save** action link.



Group Name:	group1
Eligible Reals:	r1
Weight:	1
Priority:	0

3. Select **Server Load Balance > Virtual Services > Virtual Services**. Specify parameters in the **Add Virtual Service** area and click the **Add** action link. In the **Virtual Service List** table, double-click a virtual service. Associate the virtual service with the default or backup policy in the **Associate Groups** area, and then click the **Add** action link.



Virtual Service Or Vlink:	virtualserv
Eligible Groups:	group1
Eligible Policies:	default

	Eligible Vlink Or Groups	Policy Name	Eligible Policies	Virtual Service Or Vlink
1	group1		default	virtualserv

Attribute	Value

- **CLI:**

1. Execute the following command to configure the IPv6 virtual service:

slb virtual http <virtual_name> <vip> [vport] [arp|noarp] [max_conn]

For example:

FortiBalancer(config)#**slb virtual http virtualserv 3fff::251 8080 1000 tcp 3 3**

2. Execute the following command to configure the SLB policy:

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

For example:

FortiBalancer(config)#**slb default virtualserv g1**

3. Execute the following command to configure the real server and SLB group:

slb real http *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|none] [hc_up] [hc_down]*

```
slb group method <group_name> [algorithm]
```

For example:

```
FortiBalancer(config)#slb real http Serv1 172.16.65.251
FortiBalancer(config)#slb real http Serv2 172.16.66.251
FortiBalancer(config)#slb group method g1 rr
```

16.3 DNS64 and NAT64

16.3.1 Oveview

The DNS64 function converts the DNS AAAA queries sent from IPv6 clients to DNS A queries and then converts the DNS A responses to DNS AAAA responses. This ensures that IPv6 clients can access IPv4 servers. The FortiBalancer appliance returns the translated IPv6 addresses to IPv6 clients. When IPv6 clients use these IP addresses to access IPv6 servers, the NAT64 (Network Address Translation IPv6 to IPv4) function converts the IPv6 packets sent from these clients to IPv4 packets. When the FortiBalancer appliance receives IPv4 packets from IPv4 servers, the NAT64 function converts IPv4 packets to IPv6 packets. This ensures that IPv6 clients can communicate with IPv4 servers normally. The DNS64 and NAT64 functions can be deployed on two FortiBalancer appliances separately, or deployed on one FortiBalancer appliance.

16.3.2 Working Mechanism

The DNS64 and NAT64 functions are applicable to the "IPv6 to IPv4" scenario, as shown in the following figure.

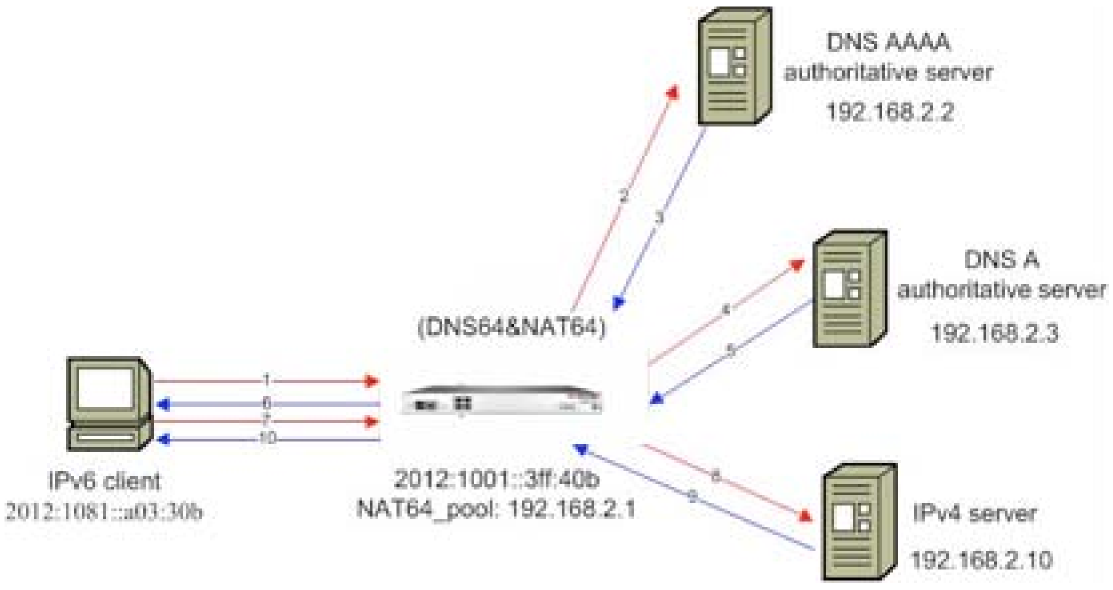


Figure 16-2 "IPv6 to IPv4" Application Scenario

The working process of the DNS64 and NAT64 functions is as follows:

1. An IPv6 client (2012:1081::a03:30b) sends a DNS AAAA query to the FortiBalancer appliance (2012:1001::3ff:40b) to resolve the domain name "www.example.com".

2. The FortiBalancer appliance sends the DNS AAAA query to the DNS AAAA authoritative server for the domain name.

3. If the DNS AAAA authoritative server has no AAAA record for the domain name, it will return an empty DNS AAAA response to the FortiBalancer appliance. The FortiBalancer appliance will ignore this response.

4. The FortiBalancer appliance waits for 100 ms after sending the DNS AAAA query. If the FortiBalancer appliance does not receive any valid DNS AAAA response, it will send a DNS A query to the DNS A authoritative server for the domain name.

5. The FortiBalancer appliance receives the DNS A response (for example, A: www.example.com - 192.168.2.10) from the DNS A authoritative server.

6. The FortiBalancer appliance converts the DNS A response to a DNS AAAA response (for example, AAAA： www.example.com - 64:ff9b::192.168.2.10) by adding the configured IPv6 prefix. Then, the FortiBalancer appliance returns the converted DNS AAAA response to the IPv6 client.

7. The IPv6 client uses the converted IPv6 address to access “www.example.com”.

8. The FortiBalancer appliance converts the IPv6 packet (src: 2012:1081::a03:30b; dst: 64:ff9b::192.168.2.10) sent from the client to an IPv4 packet (src: 192.168.2.1; dst: 192.168.2.10), and sends the IPv4 packet to the target IPv4 server.

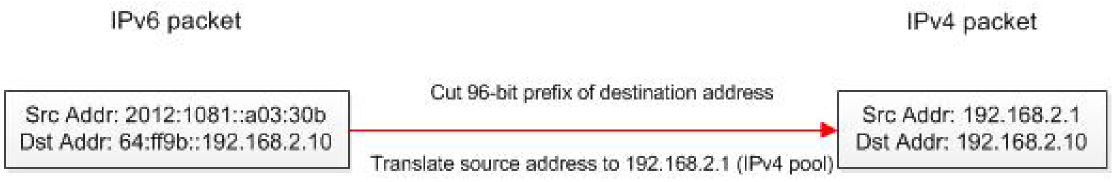

IPv6 packet
IPv4 packet
Src Addr: 2012:1081::a03:30b
Dst Addr: 64:ff9b::192.168.2.10
Cut 96-bit prefix of destination address
Translate source address to 192.168.2.1 (IPv4 pool)
Src Addr: 192.168.2.1
Dst Addr: 192.168.2.10

Figure 16-3 NAT64 Address Translation

9. The IPv4 sever returns an IPv4 packet (src: 192.168.2.10; dst: 192.168.2.1) to the FortiBalancer appliance.

10. The FortiBalancer appliance converts the IPv4 packet to an IPv6 packet (src: 64:ff9b::192.168.2.10; dst: 2012:1081::a03:30b) and returns the IPv6 packet to the IPv6 client.

16.3.3 Application Notes

The DNS64 function can be enabled on only one DNS virtual service. This virtual service, acting as the DNS proxy, converts DNS AAAA queries to DNS A queries and then converts DNS A responses to DNS AAAA responses.

To make the DNS64 function work properly, you need to configure the “default” and “backup” policies for this virtual service. The FortiBalancer appliance forwards DNS AAAA queries based on the “default” policy and forwards DNS A queries based on the “backup” policy. Therefore, the real servers associated with the “default” policy should be DNS servers that can answer AAAA records, and those associated with the “backup” policy should be DNS servers that can answer A records.

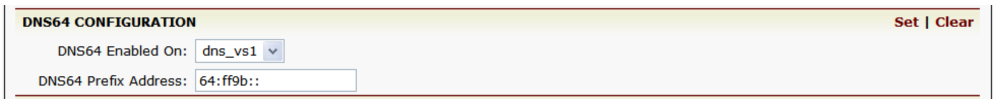
16.3.4 Configuring DNS64 and NAT64

➢ **web UI:**

1. Select **System Configuration** > **Advanced Networking** > **IP Pool**. In the **Add IP Pool** area, specify the required parameters and click the **Add** action link to save the configuration.
2. Select **Server Load Balance** > **Real Services** > **Real Services**. In the **SLB Real Services Configuration** area, click the **Add Real Service Entry** action link. In the **Add Real Service Entry** area of displayed page, specify the required parameters and click **Save** to save the configuration.
3. Select **Server Load Balance** > **Groups** > **Groups**. In the **Add Group** area, specify the required parameters and click the **Add** action link. In **Groups List**, double-click the newly added group. In the **Group Members** area of the displayed page, click the **Add** action link. In the **Add Group Member** area of the displayed page, specify the required parameters and click **Save** to save the configuration.
4. Select **Server Load Balance** > **Virtual Services** > **Virtual Services**. In the **Add Virtual Service** area, specify the required parameters and click the **Add** action link. In **Virtual Service List**, double-click the newly added virtual service. In the **Associate**

Groups area of the displayed page, associate the virtual service with the "default" or "backup" policy and click the **Add** action link.

5. Select **System Configuration > NAT > V4/V6 NAT**. In the **DNS64 Configuration** area, specify the required parameters and click the **Set** action link to save the configuration.

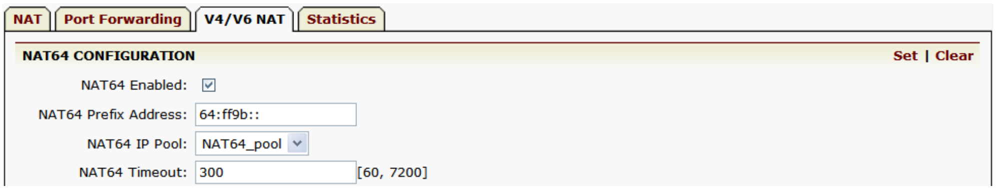


DNS64 CONFIGURATION — Set | Clear

DNS64 Enabled On: dns_vs1

DNS64 Prefix Address: 64:ff9b::

6. In the **NAT64 Configuration** area, specify the required parameters and click the **Set** action link to save the configuration.



NAT64 CONFIGURATION — Set | Clear

NAT64 Enabled: ☑

NAT64 Prefix Address: 64:ff9b::

NAT64 IP Pool: NAT64_pool

NAT64 Timeout: 300 [60, 7200]

➢ **CLI:**

1. Execute the following command to add an IPv4 address pool:

```
ip pool <pool_name> <start_ip> [end_ip]
```

For example:

```
FortiBalancer(config)#ip pool NAT64_pool 192.168.2.1
```

2. Execute the following commands to complete SLB configurations:

```
slb real dns <real_name> <ip> <port> [max_conn]
[dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|none] [hc_up] [hc_down] [timeout]
slb real enable <real_name>
slb group method <group_name> [algorithm]
slb group member <group_name> <real_name> [weight|cookie|url] [priority]
slb virtual dns <virtual_name> <vip> [vport] [arp|noarp] [max_conn]
slb policy default {virtual_name|vlink_name} {group_name|vlink_name}
slb policy backup {virtual_name|vlink_name} {group_name|vlink_name}
```

For example:

```
FortiBalancer(config)#slb real dns dns_rs1 192.168.2.2
FortiBalancer(config)#slb real dns dns_rs2 192.168.2.3
FortiBalancer(config)#slb group method g1 rr
FortiBalancer(config)#slb group member g1 dns_rs1
FortiBalancer(config)#slb group method g2 rr
FortiBalancer(config)#slb group member g2 dns_rs2
FortiBalancer(config)#slb virtual dns dns_vs1 2012:1001::3ff:40b
FortiBalancer(config)#slb policy default dns_vs1 g1
FortiBalancer(config)#slb policy backup dns_vs1 g2
FortiBalancer(config)#slb real enable dns_rs1
FortiBalancer(config)#slb real enable dns_rs2
```

Note: The DNS virtual service on which the DNS64 function is enabled can be associated with groups by using the "default" or "backup" policy only.

3. Execute the following commands to configure the DNS64 function:

```
ipv6 dns64 on <vs_name>
ipv6 dns64 prefix <dns64_prefix>
```

For example:

```
FortiBalancer(config)#ipv6 dns64 on dns_vs1
FortiBalancer(config)#ipv6 dns64 prefix “64:ff9b::”
```

4. Execute the following commands to configure the NAT64 function:

```
ipv6 nat64 on
ipv6 nat64 ippool <ipv4_pool_name>
ipv6 nat64 prefix <nat64_prefix>
ipv6 nat64 timeout <idle_timeout>
```

For example:

```
FortiBalancer(config)#ipv6 nat64 on
FortiBalancer(config)#ipv6 nat64 ippool NAT64_pool
FortiBalancer(config)#ipv6 nat64 prefix “64:ff9b::”
FortiBalancer(config)#ipv6 nat64 timeout 300
```

Note: If the DNS64 and NAT64 functions need to work together on one FortiBalancer appliance, make sure that the values of the parameters “dns64_prefix” and “nat64_prefix” are the same.

16.4 DNS46 and NAT46

16.4.1 Overview

The DNS46 function converts the DNS A queries sent from IPv4 clients to DNS AAAA queries and then converts the DNS AAAA responses to DNS A responses. It also creates mapping records between the IPv6 addresses and IPv4 addresses. The FortiBalancer appliance returns the translated IPv4 addresses to IPv4 clients. When IPv4 clients use these IPv4 addresses to access IPv6 servers, the NAT46 function converts IPv4 packets sent from clients to IPv6 packets based the created mapping records. When the FortiBalancer appliance receives IPv6 packets from IPv6 servers, the NAT46 function converts IPv6 packets to IPv4 packets. This ensures that IPv4 clients can communicate with IPv6 servers normally. Because the DNS46 and NAT46 functions both use the mapping records, they must be deployed on one FortiBalancer appliance.

16.4.2 Working Mechanism

The DNS46 and NAT46 functions are applicable to the “IPv4 to IPv6” scenario, as shown in the following figure.

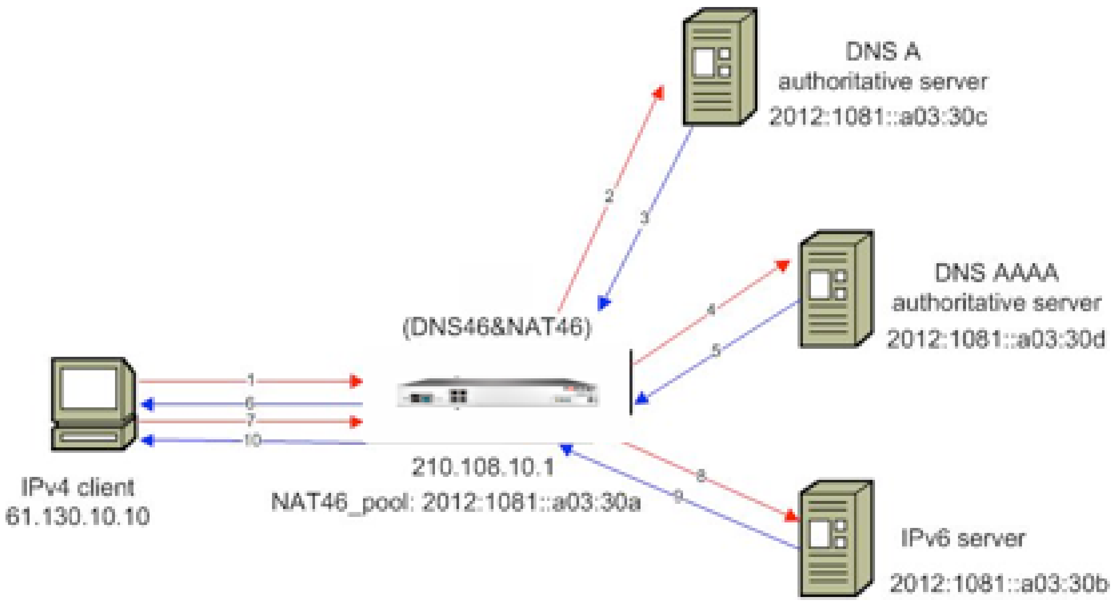


Figure 16-4 "IPv4 to IPv6" Application Scenario

The working process of the DNS46 and NAT46 functions is as follows:

1. An IPv4 client (61.130.10.10) sends a DNS A query to the FortiBalancer appliance (210.108.10.1) to resolve the domain name "www.example.com".

2. The FortiBalancer appliance sends the DNS A query to the DNS A authoritative server for the domain name.

3. If the DNS A authoritative server has no A record for the domain name, it will return an empty DNS A response to the FortiBalancer appliance. The FortiBalancer appliance will ignore this response.

4. The FortiBalancer appliance waits for 100 ms after sending the DNS A query. If the FortiBalancer appliance does not receive any valid DNS A response, it will send a DNS AAAA query to the DNS AAAA authoritative server for the domain name.

5. The FortiBalancer appliance receives the DNS AAAA response (for example, AAAA: www.example.com - 2012:1081::a03:30b) from the DNS AAAA authoritative server.

6. The FortiBalancer appliance converts the DNS AAAA response to a DNS A response (for example, A: www.example.com - 210.108.10.10) based on the configured address pool (that is, the IPv4 mapping subnet configured by using the command "**ipv6 dnsnat46 ipmap**"). Then, the FortiBalancer appliance returns the converted DNS A response to the IPv4 client. Meanwhile, the system creates a mapping record between 2012:1081::a03:30b and 210.108.10.10.

7. The IPv4 client uses the converted IPv4 address to access "www.example.com".

8. The FortiBalancer appliance converts the IPv4 packet (src: 61.130.10.10; dst: 210.108.10.10) sent from the client to an IPv6 packet (src: 2012:1081::a03:30a; dst：2012:1081::a03:30b) based on the created address mapping record, and sends the IPv6 packet to the target IPv6 server.

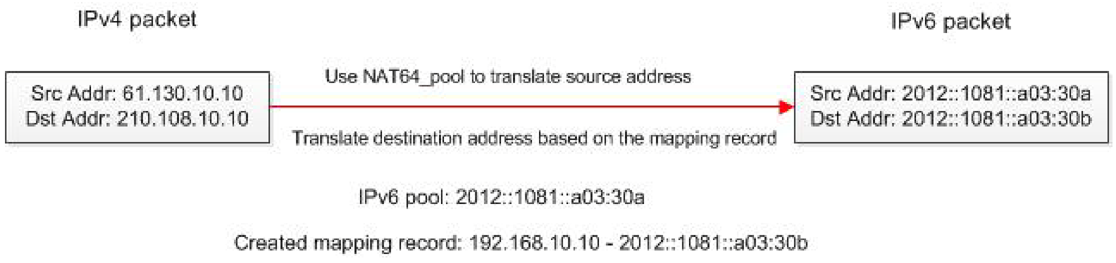

IPv4 packet
IPv6 packet
Src Addr: 61.130.10.10
Dst Addr: 210.108.10.10
Use NAT64_pool to translate source address
Translate destination address based on the mapping record
Src Addr: 2012::1081::a03:30a
Dst Addr: 2012::1081::a03:30b
IPv6 pool: 2012::1081::a03:30a
Created mapping record: 192.168.10.10 - 2012::1081::a03:30b

Figure 16-5 NAT46 Address Translation

9. The IPv6 server returns an IPv6 packet (src: 2012:1081::a03:30b; dst: 2012:1081::a03:30a) to the FortiBalancer appliance.

10. The FortiBalancer appliance converts the IPv6 packet to an IPv4 packet (src: 210.108.10.10; dst: 61.130.10.10) and returns the IPv6 packet to the IPv4 client.

16.4.3 Application Notes

The DNS46 and NAT46 functions can be enabled on only one DNS virtual service. This virtual service, acting as the DNS proxy, converts DNS A queries to DNS AAAA queries and then converts DNS AAAA responses to DNS A responses.

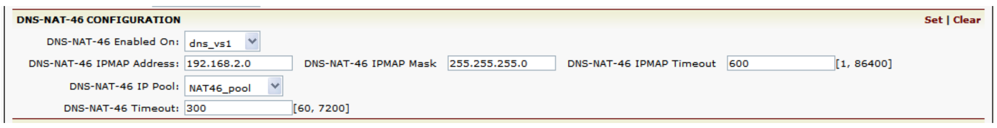
To make the DNS46 and NAT46 functions work properly, you need to configure the "default" and "backup" policies for this virtual service. The FortiBalancer appliance forwards DNS A queries based on the "default" policy and forwards DNS AAAA queries based on the "backup" policy. Therefore, the real servers associated with the "default" policy should be DNS servers that can answer A records, and those associated with the "backup" policy should be DNS servers that can answer AAAA records.

16.4.4 Configuring DNS46 and NAT46

➢ **web UI:**

1. Please refer to the section "**Configuring DNS64 and NAT64**" to complete the address pool and SLB configurations via web UI.

2. Select **System Configuration > NAT > V4/V6 NAT**. In the **DNS-NAT-46 Configuration** area, specify the required parameters and click the **Set** action link to save the configuration.



DNS-NAT-46 CONFIGURATION — Set | Clear

DNS-NAT-46 Enabled On: dns_vs1
DNS-NAT-46 IPMAP Address: 192.168.2.0 DNS-NAT-46 IPMAP Mask 255.255.255.0 DNS-NAT-46 IPMAP Timeout 600 [1, 86400]
DNS-NAT-46 IP Pool: NAT46_pool
DNS-NAT-46 Timeout: 300 [60, 7200]

➢ **CLI:**

1. Please refer to the section "**Configuring DNS64 and NAT64**" to complete the address pool and SLB configurations via CLI.

   For example:

```
FortiBalancer(config)#ip pool NAT46_pool 2012:1081::a03:30a

FortiBalancer(config)#slb real dns dns_rs1 2012:1081::a03:30c
FortiBalancer(config)#slb real dns dns_rs2 2012:1081::a03:30d
FortiBalancer(config)#slb group method g1 rr
FortiBalancer(config)#slb group member g1 dns_rs1
FortiBalancer(config)#slb group method g2 rr
FortiBalancer(config)#slb group member g2 dns_rs2
FortiBalancer(config)#slb virtual dns dns_vs1 210.108.10.1
FortiBalancer(config)#slb policy default dns_vs1 g1
FortiBalancer(config)#slb policy backup dns_vs1 g2
FortiBalancer(config)#slb real enable dns_rs1
FortiBalancer(config)#slb real enable dns_rs2
```

2. Execute the following command to enable both the DNS46 and NAT46 functions for a specified DNS virtual service:

```
ipv6 dnsnat46 on <vs_name>
```

   For example:

```
FortiBalancer(config)#ipv6 dnsnat46 on dns_vs1
```

3. Execute the following command to configure an IPv4 subnet used to create the address mapping table:

```
ipv6 dnsnat46 ipmap <ipv4_address> <netmask> [timeout]
```

For example:

```
FortiBalancer(config)#ipv6 dnsnat46 ipmap 192.168.2.0 255.255.255.0 600
```

4. Execute the following command to specify the IPv6 address pool used by the NAT46 function:

```
ipv6 dnsnat46 ippool <ipv6_pool_name>
```

For example:

```
FortiBalancer(config)#ipv6 dnsnat46 ippool NAT46_pool
```

5. Execute the following command to set the idle timeout period for NAT46 TCP connections:

```
ipv6 dnsnat46 timeout <idle_timeout>
```

For example:

```
FortiBalancer(config)#ipv6 dnsnat46 timeout 300
```

16.5 IPv6 support for NAT

16.5.1 Overview

The FortiBalancer appliance not only supports NAT64 and NAT46, but also supports “IPv6 to IPv6” NAT, which is able to translate the IPv6 addresses on the internal network to the IPv6 addresses on the Internet. In addition, NAT pool configurations also support IPv6.

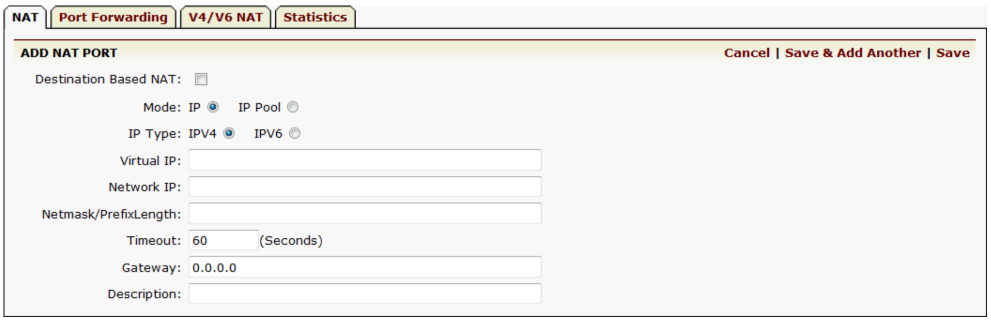
Note:

- “IPv6 to IPv6” NAT supports TCP and UDP packets, but does not support ICMP and FTP packets.
- When configuring the “IPv6 to IPv6” NAT, the parameter “gateway” cannot be configured in the “**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*” command.

16.5.2 Configuration Example

- **web UI:**

Select **System Configuration > NAT**. Click the Add NAT Port action link, specify parameters in the Add NAT Port area and click the Save action link.



Destination Based NAT:	☐
Mode:	IP ● IP Pool ○
IP Type:	IPV4 ● IPV6 ○
Virtual IP:	
Network IP:	
Netmask/PrefixLength:	
Timeout:	60 (Seconds)
Gateway:	0.0.0.0
Description:	

- **CLI:**

Execute the following command to configure the IPv6 NAT:

```
nat port {pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway]
[description]
```

For example:

```
FortiBalancer(config)#nat port ipv6_pool 3fff::12 64
```

16.6 NDP

16.6.1 Overview

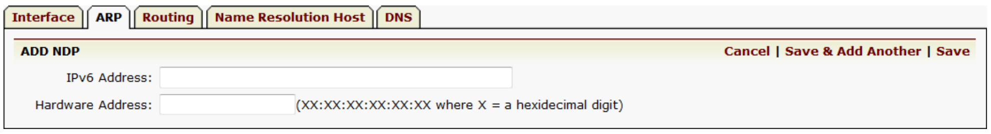
NDP (Neighbor Discovery Protocol), a key protocol of the IPv6 stack, can be used for obtaining the link address information of other neighbor nodes connected with the local nodes.

Similar to the ARP (Address Resolution Protocol) of the IPv4 stack, NDP can perform address transformation between the network layer and the link layer. The difference is that NDP uses ICMPv6 (Internet Control Message Protocol version 6) and multicast to manage the information exchanged among the neighbored nodes (within the same link), and keeps the address mapping between the network layer and the link layer in the same subnet.

16.6.2 Configuration Example

- **web UI:**

Select **System Configuration > Basic Networking > ARP**. Click the Add action link in the NDP Configuration area, specify parameters in the Add NDP area and click the Save action link.



IPv6 Address:

Hardware Address: (XX:XX:XX:XX:XX:XX where X = a hexidecimal digit)

- **CLI:**

Execute the following command to configure the NDP entry:

```
ipv6 ndp <ipv6_address> <mac_address>
```

For example:

```
FortiBalancer(config)#ipv6 ndp bb::55 00:21:9C:45:80:9F
```

Chapter 17 ePolicy

17.1 Overview

ePolicy is a script-based function for extending the capabilities of the FortiBalancer appliance. Using the scripts written in Tools Command Language (TCL), you can customize new features in addition to the existing functions on the FortiBalancer appliance. For example, the FortiBalancer appliance can be customized to support more application protocols, precisely control IP application traffic in both incoming and outgoing directions, or control the access of the specified client to real services.

17.2 ePolicy Elements

The elements of ePolicy are as follows:

- Event
- Command
- Command invocation rule

17.2.1 Event

ePolicy uses an event-driven and message-response mechanism. The FortiBalancer appliance defines an event for every action occurring in each Client-FortiBalancer-Server connection. When such an event occurs, the FortiBalancer appliance will process traffic according to preconfigured ePolicy commands.

17.2.2 Command

ePolicy uses commands to instruct the FortiBalancer appliance to process traffic after an event occurs, such as rewriting packet contents, selecting real servers, selecting groups, or querying whether a group has valid real servers.

17.2.3 Command Invocation Rule

Command invocation rules indicate the relationship between events and commands. Based on the command invocation rules, you can flexibly combine the events and commands to intercept, detect, convert, or redirect the IP application traffic in both incoming and outgoing directions.

For detailed information of events, commands, and command invocation rules, contact Fortinet Customer Support for related documents.

17.3 ePolicy Scripts

By functions, the scripts of ePolicy can be classified into the following:

- Setting script: specifies the traffic type of a virtual service. The following table lists the setting scripts that are currently supported:

Table 17–1 Content of Setting Scripts

Traffic Type	Content of the Setting Script
HTTP	message::type http
Diameter	message::type binary binary_message::length_start_offset 1 binary_message::length_end_offset 3
Generic TCP	message::type binary

- Runtime script: specifies the action of the FortiBalancer appliance for an event. The content of a runtime script should be written according to the actual requirement based on events, commands, and command invocation rules. For the examples of the runtime scripts, contact Customer Support for related documents.

17.4 ePolicy Applications

With ePolicy scripts, an FortiBalancer appliance can be customized to:

- Balance loads. When being applied to Server Load Balancing (SLB), ePolicy can work as SLB policies and collaborate with SLB methods to realize load balancing among real services.
- Analyze the packet contents of the HTTP, Simple Object Access Protocol (SOAP), eXtensible Markup Language (XML), and Diameter protocols.
- Receive, send, analyze, and discard Generic TCP and TCPS packets.
- Perform pattern matching for txt data
- Control TCP connections
- Monitor and take statistics of traffic

17.4.1 SLB Methods Collaborating with ePolicy

In SLB, ePolicy can collaborate with the following methods:

- Round Robin (rr)
- Least Connection (lc)
- Shortest Response (sr)
- Persistent IP (pi)
- Hash IP (hi)
- Hash IP and port (hip)
- Consistent hash IP (chi)
- SNMP (snmp)

17.4.2 SLB Polices and ePolicy

In SLB, ePolicy has higher priority than the existing SLB policies. When a virtual service is associated with ePolicy scripts, the SLB policies associated with the virtual service does not take effect.

17.5 ePolicy Configurations

To complete the ePolicy configuration, perform the following steps:

➢ Prepare setting and runtime scripts according to events, commands, and command invocation rules.

➢ Import setting and runtime scripts.

➢ Associate the virtual service with the setting script.

➢ Associate the virtual service with the runtime script.

The following takes balancing loads among servers according to HTTP request packet method to describe how to configure ePolicy.

17.5.1 Preparing Setting and Runtime Scripts

The following table shows the contents of the setting and runtime scripts:

Table 17–2 Setting and Runtime Scripts

	Name	Content
Setting Script	setting_http.tcl	message::type http

Runtime Script	http_slb.tcl	when HTTP_REQUEST_HEADER {
		    if { [http::method] == "GET" } {
		        slb::select_server realserver_1
		    } else {
		        slb::select_server realserver_2
		    }
		}

In which, “http_slb.tcl”, “realserver_1” and “realserver_2” are the names of SLB real servers.

For detailed information of events, commands, and command invocation rules, contact Customer Support for related documents.

17.5.2 Importing Setting and Runtime Scripts

➢ **CLI:**

1. Execute the following command to import the setting script:

```
epolicy import setting <url> <setting_script_name>
```

For example:

```
FortiBalancer(config)#epolicy import setting http://192.168.10.10/setting_http.tcl
setting_http.tcl
```

2. Execute the following command to import the runtime script:

```
epolicy import script <url> <script_name>
```

For example:

```
FortiBalancer(config)#epolicy import script http://192.168.10.10/http_slb.tcl
http_slb.tcl
```

17.5.3 Associating the Virtual Service with the Setting Script

➢ **CLI:**

Execute the following command to associate the virtual service with the setting script:

```
epolicy attach setting <vs_name> <setting_script_name>
```

For example:

```
FortiBalancer(config)#epolicy attach setting vs_epolicy setting_http.tcl
```

17.5.4 Associating the Virtual Service with the Runtime Script

➢ **CLI:**

Execute the following command to associate the virtual service with the runtime script:

```
epolicy attach script <vs_name> <script_name>
```

For example:

```
FortiBalancer(config)#epolicy attach script vs_epolicy http_slb.tcl
```

17.5.5 Configuration Results

After the preceding configurations are completed, the FortiBalancer appliance will:

- Direct HTTP request packets whose method is GET to real server realserver_1.
- Direct HTTP request packets with other methods to real server realserver_2.

Chapter 18 Logging

18.1 Overview

The Logging mechanism used by the FortiBalancer appliance is Syslog compliant. System error and HTTP access information during proxy application are logged by using the logging subsystem. Syslog is a standard program for Unix and there are also Syslog implementations for Windows. On the Unix platform, syslog is started by the syslogd daemon. The syslogd daemon takes charge of receiving and storing log messages from local machine or remote machine, which listens at UDP 514 port. FortiBalancer appliance supports three remote log servers.

18.2 Understanding Logging

18.2.1 Syslog

Syslog is a protocol that is used for the transmission of event notification message across networks.

Syslog logging has eight valid levels of log message severity: emerg, alert, crit, err, warning, notice, info and debug. And the supported facilities are LOCAL0 to LOCAL7. Users can view the internal log buffer, select the transport protocol, and configure syslog source and destination ports and the alerts on log message string match.

18.2.2 RFC 5424 Syslog

RFC5424 defines the standard format of syslogs. The FortiBalancer appliance supports the RFC 5424 syslog function. When the RFC 5424 syslog function is enabled, the system will generate system logs in the standard format defined by RFC 5424. The format is "<PRI>VER TIMESTAMP HOSTNAME APPNAME PROCID MSGID STRUCTURED-DATA MSG-CONTENT". (The PROCID and STRUCTURED-DATA fields are not supported temporarily and are displayed as "-".) By default, the RFC 5424 syslog function is disabled. The configuration of "**log rfc5424 on**" takes effect only when the system logging function has been enabled by using the "**log on**" command.

18.2.3 HTTP Access Logging

HTTP Access Logging is the logging of information about every HTTP request and its response in a specific predefined format.

HTTP Access Logging supports four standard formats: Combined, WELF (WebTrends Enhanced Log), Common and Squid. And users can define their own logging format by using the "**log http custom**" command.

Note: The FortiBalancer appliance will record an HTTP access log only after the HTTP communication between the client and the Web server is completed successfully.

18.2.4 Log Filtering

Log filtering is designed to filter logs to different log servers by matching filter strings which are configured in the command "**log filter**".

Log filtering in the OS allows administrators to collect only the logs that they are interested in instead of having to capture all the logs. For example, the administrator of "www.site1.com" may want to only collect the HTTP access logs for "www.site1.com". Knowing if the logs contain a keyword "site1.com", the administrator can create a filter for a log definition that captures only the logs which match the keyword. The administrator will now have a log file which contains only the desired logs.

If multiple log filters are set on a syslog host, the logs matching one of the filter strings will go to the syslog host.

18.3 Logging Configuration

18.3.1 Configuration Guidelines

Table 18-1 General Settings of Logging

Operation	Command
Enable the logging	**log {on|off}**
Enable RFC 5424 Syslog	**log rfc5424 {on|off}**
Configure the remote host	**log host** *<host_ip> [port] [udp|tcp] [host_id]*
Set log filters	**log filter** *<host_id> <filter_id> <filter_string>*
Set log level	**log level** *<level>*
Change log facility	**log facility** *<facility>*
Set HTTP access logging format	**log http** *{squid|common|combined|welf} [vip|novip] [host|nohost]* **log http custom** *<format>*

18.3.2 Configuration Example via CLI

➢ Step 1 Enable Logging function

The logging system is off by default.

```
FortiBalancer(config)#log on
```

➢ Step 2 Enable the RFC 5424 Syslog function

```
FortiBalancer(config)#log rfc5424 on
```

➢ Step 3 Set the remote host to which log messages will be sent

The remote host IP address must be specified in dotted IP format. The remote port is optional and the default value is 514. The transport protocol for the syslog messages can be either UDP or TCP and the default is UDP. In our example, the host of 10.2.37.1 is listening for log message at UDP 514 port.

```
FortiBalancer(config)#log host 10.2.37.1 514 udp 1
```

➢ Step 4 Set log filters for the configured host

No more than 3 log filters can be set on one syslog host. Log filter canot be set on the syslog host whose ID is 0 (it is configured by the command "**log host**"). After this command is executed, only the logs matching this filter string go to the syslog host.

```
FortiBalancer(config)#log filter 1 1 "index"
```

➢ Step 5 Change the minimum log level at which messages will be logged

Once a log level is set, messages with level below the configured level will be ignored. The default level is info.

```
FortiBalancer(config)#log level err
```

➢ Step 6 Change the syslog facility

The default facility is LOCAL0.

```
FortiBalancer(config)#log facility LOCAL0
```

- Step 7 Configure the HTTP access logging format

HTTP access information can be logged in one of the standard formats Squid, WELF, Common and Combined, or it can be logged in a custom format specified by the user.

```
FortiBalancer(config)#log http squid
```

- Step 8 Generate a test log

You can run the command “log test” to generate an emerg-level log.

```
FortiBalancer(config)#log test
```

- Step 9 View and clear logs

You can run the following command “**show log buff** *{forward|backward} [match_str]*” to view logs in the log buffer. The parameters “backward” and “forward” are used to display the logs that are latest and first generated respectively.

```
FortiBalancer(config)#show log buffer backward
start of buffer
<128>1 2012-07-17T06:35:26Z FortiBalancer - - 100021002 - Fortinet test message
```

You can run the command “**clear log buff**” to clear logs from the log buffer.

```
FortiBalancer(config)#clear log buffer
```

Chapter 19 System Management

19.1 Administrative Tools

19.1.1 Overview

This chapter will focus on various configuration maintenance elements, such as downloading new OS software, rebooting your FortiBalancer appliance, reverting your configuration to a previously saved status or returning the FortiBalancer appliance to its factory default settings among other closing strategies.

The final series of configuration options concern the running operation of your FortiBalancer appliance and its relationship with the rest of the network architecture. Through the various subfolders (within the web UI) that are revealed once you click on the "Admin Tools" folder you will discover a series of sub-folders allowing you to set administrative passwords, perform configuration synchronization, set SNMP traps and define reboot strategies among other operations. Otherwise all of these features may be configured via the CLI.

19.1.2 Administrative Tools Configuration

19.1.2.1 Configuration Guidelines

Table 19-1 General Settings of Administrative Tools

Operation	Command
Configuring External Authentication	**admin aaa {on|off}** **admin aaa method** *[radius|tac_x]* **admin aaa server** *<server_id> <host_name|ip_address> <port> <secret>*
System shutdown and reboot	**system shutdown** *[halt|poweroff]* **system reboot** *[interactive|noninteractive]*
Configuration file maintenance	**clear config file** **clear config secondary** **clear config primary** **clear config all** **clear config factorydefault** **clear config timeout** **write memory** **write file** *<file_name>* **write net tftp** *<ip_tftp> <file_name>* **write net scp** *{remote_server_ip|name} <user_name> <config_file_name>* **config memory** **config net tftp** *<tftp_server_ip> <config_file_name>* **config file** *<file_name>*
Software upgrade	**system update** *<url>*
Configuration Synchronization	**synconfig peer** *<peer_name> <peer_ip>* **synconfig to** *<name>* **synconfig from** *<name>*
SDNS Synchronization	**synconfig sdns peer** *<peer_name> <peer_ip>* **synconfig sdns to** *<peer_name>*
Monitoring	**graph name** *<new_name>* **graph rename** *<old_name> <new_name>* **graph settings displaymode** *{nostack|stack} <graph_name>* **graph item** *<graph_name> <module_name> <type> [service] <scale> <color> [order] [legend_string]*
NTP	**ntp {on|off}** **ntp server** *<ip> [version]*

Operation	Command
	show ntp **clear ntp**
XML RPC	**xmlrpc {on|off}** *[https|http]* **xmlrpc port** *<port>* **show xmlrpc** **clear xmlrpc**
Remote access	**ssh remote** *"user@hostname"* **telnet** *"host port"*

19.1.2.2 Configuration Example via CLI

19.1.2.2.1 Configuring External Authentication

If you have an external authentication server (RADIUS/Tacacs), you may use these servers to authenticate the SSH/web UI logon request. The external authentication will be performed when the "**admin aaa**" command is set to ON and the logon user name does not exist in the FortiBalancer system.

```
FortiBalancer(config)#admin aaa on
FortiBalancer(config)#admin aaa method RADIUS
FortiBalancer(config)#admin aaa server es01 "10.1.1.1" 1812 radiussecret
FortiBalancer(config)#admin aaa server es02 radius_host 1812 radiussecret
```

19.1.2.2.2 System Maintenance

Simply enough, employing the "**quit**" command will allow you to exit the CLI. In the event you want to terminate all FortiBalancer appliance interactions with your network, you will need to use the "**system shutdown**" command.

```
FortiBalancer(config)#system shutdown
```

The FortiBalancer appliance will prompt you with an alert to verify the shutting down process. By entering "**YES**", case sensitive, the FortiBalancer appliance will commence the shutting down operation. After a brief, 60-second period, users may turn off the appliance.

In some cases when dealing with configuration changes you might need to reboot the box.

```
FortiBalancer(config)#system reboot
```

19.1.2.2.3 Configuration File Maintenance

When working with configurations there may come a time that you want to experiment with a new configuration strategy, but not overwrite your known working configuration. The OS possesses several options for working with configurations files.

In general, you work with the running configuration and write it to disk by using the "**write memory**" command. You can also save the configuration to a file by using the "**config file**" command, on the FortiBalancer appliance. Finally, you may export and import the configuration by using TFTP.

To clear the running configuration on the FortiBalancer appliance:

```
FortiBalancer(config)#clear config all
```

Now the FortiBalancer appliance has been returned to its factory default settings.

When working with the "**write memory**" command, keep in mind that this is the configuration file that will be loaded when the FortiBalancer reboots. If you have made changes and want to clear the configuration currently running, use the "**clear config**" command.

At any point when you want to import a previously saved configuration, you will need to clear the current, running configuration as previously discussed in this chapter. Once this is completed, you can import the new configuration. The FortiBalancer appliance affords you the opportunity to save configurations to three separate places; the “memory” file which is where the FortiBalancer appliance calls up configuration settings upon reboot, the “file” where the FortiBalancer appliance can store several different configurations, and to the “net” which refers to saving a file to a remote location on the network. To save configuration files:

```
FortiBalancer(config)#write net tftp 10.10.0.3 default_config
```

To recall a previously saved configuration and merge it into the running parameters of the appliance:

```
FortiBalancer(config)#config memory
FortiBalancer(config)#config file new_lb
FortiBalancer(config)#config net tftp 10.10.0.3 default_config
```

When loading the configuration file while the box is running, it is important to remember that the configuration is merged with the running configuration. So you need to choose to clear the appropriate configuration from the FortiBalancer appliance before you load a configuration file. For example, if you have 5 real servers defined and execute the “**config net tftp 10.10.0.3 default_config**” command and if that configuration file has 5 real servers using the same real names you will get an error since you cannot have duplicate real server names.

19.1.2.2.4 Software Upgrade Procedure

To see the current version of OS software that is running, we use the “**show version**” command.

```
FortiBalancer(config)#show version

FortiBalancerOS Rel.TM.8.4.0.1 build on Mon Mar 18 18:12:09 2013

          Host name  :  FortiBalancer
         System CPU  :  Intel(R) Core(TM)2 Quad CPU
         System RAM  :  3842964 kbytes.
   System boot time  :  Mon Mar 18 19:10:19 GMT (+0000) 2013
       Current time  :  Tue Mar 19 19:54:09 GMT (+0000) 2013
     System up time  :  1 day, 00:44
  Platform Bld Date  :  Mon Mar 18 18:12:09 CST 2013
             SSL HW  :  HW ( 1X16C ) Initialized
     Compression HW  :  No HW Available
       Power supply  :  2U, AC, 2-cords, Redundancy
  Network Interface  :  4 x Gigabit Ethernet copper
              Model  :  FortiBalancer 2000
      Serial Number  :  0437A334520001000301104431 6464
  Licensed Features  :  WebWall   Clustering   L4SLB   L7SLB   Caching
                        SSL   tProxy   AppGateway   SwCompression   LLB   GSLB
                        QoS   MultiLang   DynRoute   FFO   REDUNDANT   IPv6
        License Key  :  f1bd6e06-d29016c1-c053e5eb-00d27cb7-d3f75a85-00000000-05d5d9
                        ab-99999999

Fortinet Customer Support
Update               :  please contact support for instructions
Website              :  http://www.fortinet.com

Other Root
Version
Rel.FBLOS.8.3.2.3 build on Fri Feb 22 17:35:11 2013
```

To upgrade to a newer release there are several steps to take.

First, contact Customer Support to gain access to the software and documentation repository. Contact your customer support representative or send email to: support@fortinet.com

Once you have received a password and verified with a customer support engineer that the OS needs upgrade, you can download the software image using the Fortinet website. You should download the image to either a local Web server or anonymous FTP server.

It is recommended that you use the serial console to upgrade the OS. Once you have a console connection you can upgrade the appliance by using the "**system update**" command. Currently the upgrade procedure supports two upgrade methods: HTTP or FTP. The commands are identical except from the URL.

For example, use the command to upgrade the appliance from 192.168.10.10:

```
FortiBalancer(config)#system update http://192.168.10.10/FortiOS_rel_FBL_8_4_0_1.fn

This will upgrade your system from http://192.168.10.10/ FortiOS_rel_FBL_8_4_0_1.fn
Power outages or other systems failures may corrupt the system.
It is highly recommended that you save your configuration on an
external system prior to upgrading or downgrading.
Any configuration changes that have not been "saved" will be lost.
After a successful patch the system will be rebooted.
Fortinet, Inc.

Type "YES" to confirm upgrade: YES
```

Note: If you are to use a DNS name like: system-update http://s5.sj.example.com, make sure that you have correctly setup the resolving on the FortiBalancer appliance, using the "ip nameserver" command to define your DNS server for the "s5" host or use the "ip host" command to locally define the IP address of the "s5" host. Otherwise you will get an error when you try to download the software image.

The OS will then shutdown all load balancing features and download the software image, verify that the software is produced at Fortinet and then install it. If there is any problem with the software image, the CLI will abort the upgrade and display a prompt on the screen. Otherwise you should get a prompt on the console stating that the upgrade was successful and the FortiBalancer appliance will reboot. Upon reboot, you should use the "**show version**" command to verify that the upgrade is successful.

Caution:

1. If executing this command via an SSH connection and if the connection is lost during update procedure, the FortiBalancer appliance will not be able to complete the update process.

2. Do not disconnect the connections to the FortiBalancer appliance during the system updating process.

Software Licenses

Some software features of the FortiBalancer appliance may be under software license key control. If you need these software features, please contact customer support (https://support.fortinet.com) to obtain a new license key.

19.1.2.2.5 Configuration Synchronization

The Configuration Synchronization feature of the FortiBalancer appliance allows administrators to transfer configuration information among FortiBalancer appliances within the same network. Configuration Synchronization is a set of commands that allow you to manage and configure boxes within a network. You may transfer configuration information from one FortiBalancer appliance in a network to other FortiBalancer appliances within the same network. By using configuration synchronization, you can quickly setup an Active-Standby configuration. The rest of the section will cover how to use this feature.

Note: Synconfig commands are executed via SSH, therefore SSH must be enabled.

➢ Step 1 Configure configuration synchronization on FortiBalancer1

```
FortiBalancer1(config)#synconfig peer FortiBalancer1 192.168.1.1
FortiBalancer1(config)#synconfig to FortiBalancer2
```

➢ Step 2 Configure configuration synchronization on FortiBalancer2

```
FortiBalancer2(config)#synconfig peer FortiBalancer1 192.168.1.1
FortiBalancer2(config)#synconfig peer FortiBalancer2 192.168.1.2
FortiBalancer2(config)#synconfig from FortiBalancer1
```

Note: If WebWall is turned on for the interface which the "synconfig" command uses to synchronize with peer, you need to add the corresponding accesslist rules to allow the traffic to come in through SSH port 22 on both FortiBalancer machines (FortiBalancer appliance and the sync peer).

19.1.2.2.6 SDNS Configuration Synchronization

Administrators can synchronize SDNS configurations and BIND9 zone files except SDNS member configurations from a local FortiBalancer appliance to remote peers.

In the following example, SDNS configurations and BIND9 zone files except SDNS member configurations on FortiBalancer1 are synchronized to remote FortiBalancer2.

➢ Step 1 Configure SDNS configuration synchronization on FortiBalancer1

```
FortiBalancer1(config)#synconfig sdns peer peerlocal 172.16.83.180
FortiBalancer1(config)#synconfig sdns peer peerremote 172.16.83.120
```

➢ Step 2 Start SDNS configuration synchronization from FortiBalancer1 to FortiBalancer2

```
FortiBalancer1(config)#synconfig sdns to peerremote
```

19.1.2.2.7 Monitoring

The FortiBalancer appliance allows the administrator to view a wide range of pertinent network data through a series of pre-designed and custom (administrator defined) graphs.

➢ Step 1 Establish custom graph items

```
FortiBalancer(config)#graph name aa
FortiBalancer(config)#graph rename aa bb
FortiBalancer(config)#graph settings displaymode stack bb
FortiBalancer(config)#graph item bb "System" "CPU Utilization" "1" "red" "2"
```

19.1.2.2.8 Component Update

Component update allows for the update of many components on the FortiBalancer appliances without requiring a reboot. The effect of the component update is instantaneous. Any number of component patches can be applied to the FortiBalancer appliances. However, only the most recent component update can be reverted. The list of patches applied using component update is visible in the output of "**show version**" command.

Component patches can only be generated by Fortinet. These are in the same ".click" format as the regular OS updates, but they are much smaller in size.

19.1.2.2.9 NTP Time Synchronizer

The Network Time Protocol (NTP) time synchronizer enables the FortiBalancer appliance to synchronize the system time with the specified NTP server.

After the NTP time synchronizer is enabled, the FortiBalancer appliance will automatically synchronize the system time with the specified NTP server at the interval of about 15 minutes.

Attention:

1. It is recommended that you change the time difference between the system time of the FortiBalancer appliance and the time of the NTP server to less than 1000s before enabling the NTP time synchronizer.

2. Do not change the system time of the FortiBalancer appliance after enabling the NTP time synchronizer.

FortiBalancer appliance should be used as the NTP client rather than the NTP server.

If multiple NTP servers are configured, the FortiBalancer appliance will calculate the round-trip delays according to the time information in the response packet from each NTP server, and synchronize its system time with the NTP server with the minimum delay.

- Step 1 Configure an NTP server

```
FortiBalancer1(config)#ntp server 207.46.197.32 4
```

- Step 2 Turn on NTP time synchronizer

```
FortiBalancer1(config)#ntp on
```

Users also can use the command “**show ntp**” to view the current NTP configuration.

```
FortiBalancer1(config)#show ntp
ntp server 207.46.197.32 4
ntp on
time since restart:     1481
time since reset:       1481
packets received:       21
packets processed:      0
current version:        0
previous version:       0
bad version:            0
access denied:          0
bad length or format: 0
bad authentication:     0
rate exceeded:          0
The following explains the items in the output information:
```

Time since restart: The time in hours since the system was last rebooted.

Time since reset: The time since the statistics were reset and the system statistics monitoring file was updated. This is designed for busy servers, such as those operated by NIST, USNO, and intended as early warning detector of clogging attacks.

Packets received:	The total number of packets received.
Packets processed:	The number of packets received in response to previous packets sent.
Current version:	The number of packets matching the current NTP version.
Previous version:	The number of packets matching the previous NTP version.
Bad version:	The number of packets matching neither NTP version.

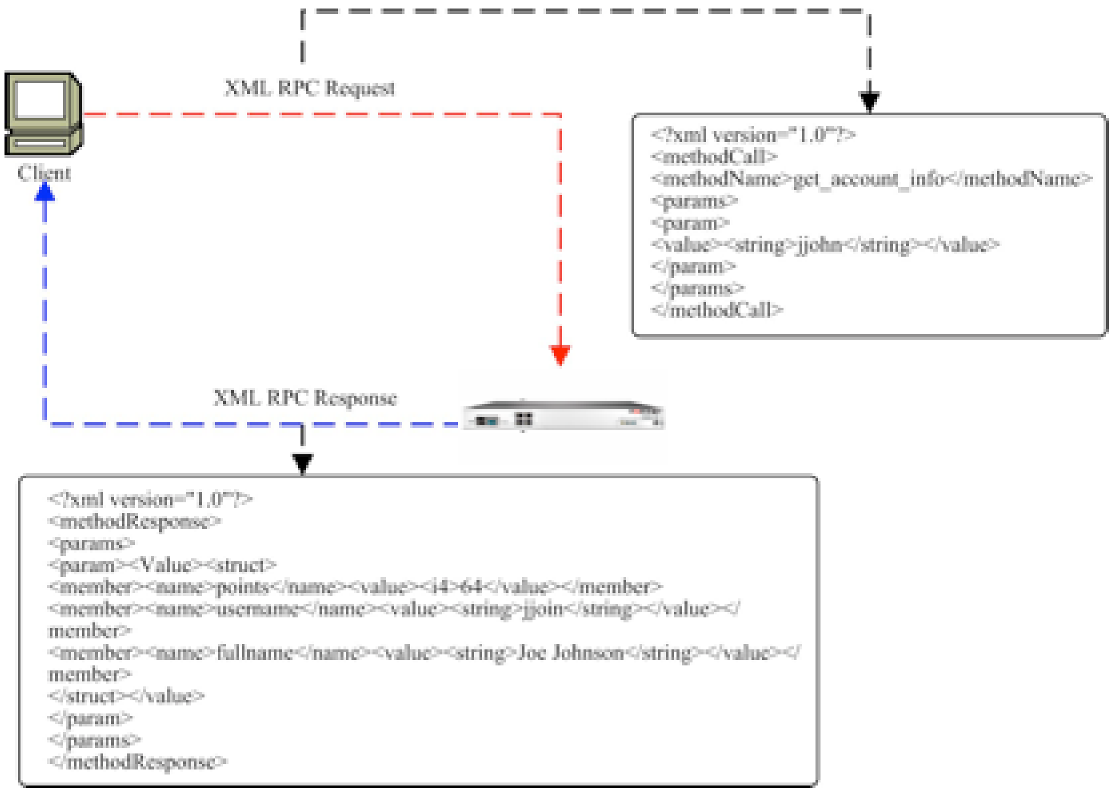
Access denied:	The number of packets denied access for any reason.
Bad length or format:	The number of packets with invalid length, format or port number.
Bad authentication:	The number of packets not verified as authentic.
Rate exceeded:	The number of packets discarded due to rate limitation.

19.1.2.2.10 XML RPC

XML RPC allows clients to run some CLI commands remotely in the OS. This enables system programmers to automate remote configuration which is difficult with web UI.

XML RPC is a Remote Procedure Calling protocol that works over the Internet, which uses HTTP as a transport mechanism and XML as an encoding.

As shown in the figure below, Client sends an HTTP POST Request to FortiBalancer. XML RPC message is the body of the HTTP Request, in which the commands to run and the commands' parameters are specified. Then, FortiBalancer decodes the XML PRC message and executes the called commands. At last it returns the results formatted in XML to Client.



```
<?xml version="1.0"?>
<methodCall>
<methodName>get_account_info</methodName>
<params>
<param>
<value><string>jjohn</string></value>
</param>
</params>
</methodCall>
```

```
<?xml version="1.0"?>
<methodResponse>
<params>
<param><Value><struct>
<member><name>points</name><value><i4>64</value></member>
<member><name>username</name><value><string>jjoin</string></value></
member>
<member><name>fullname</name><value><string>Joe Johnson</string></value></
member>
</struct></value>
</param>
</params>
</methodResponse>
```

Figure 19-1 XML RPC Working Mechanism

To realize the communication between the Client and the FortiBalancer appliance, a Perl script, called fortibalancer_xmlrpc.pl, MUST be first executed on Client. The command executed the script is:

```
fortibalancer_xmlrpc.pl –d <address> -p <port> -f <data_file>
```

In this command, <address> specifies the FortiBalancer IP address. <port> specifies the port on which the HTTP server is listening. <data_file> specifies the full path and filename of XML RPC message.

XML RPC message is formatted in XML and contains a <methodCall> tag in which <methodName> and <params> tags are embedded.

The following is an HTTP POST Request whose body is an XML RPC message:

```
POST    /cgi-bin/xmlrpc_server    HTTP/1.1
Content-Type: text/xml
Content-Length: xxx

<?xml version='1.0' ?>
<methodCall>
<methodName>slb_real</methodName>
<params>
  <param>
    <value>
      <struct>
        <member>
          <name>enable_passwd</name>
          <value>
            <string>****</string>
          </value>
        </member>
        <member>
          <name>protocol</name>
          <value>
            <string>http</string>
          </value>
        </member>
        <member>
          <name>name</name>
          <value>
            <string>fortibalancer</string>
          </value>
        </member>
        <member>
          <name>ip</name>
          <value>
            <string>10.1.1.1</string>
          </value>
        </member>
        <member>
          <name>port</name>
          <value>
            <int>80</int>
          </value>
        </member>
        <member>
          <name>maxconns</name>
          <value>
            <int>1000</int>
          </value>
        </member>
        <member>
          <name>hctype</name>
          <value>
            <string>tcp</string>
          </value>
        </member>
        <member>
          <name>hcup</name>
          <value>
            <int>1</int>
          </value>
        </member>
        <member>
          <name>hcdown</name>
          <value>
            <int>1</int>
          </value>
        </member>
      </struct>
     </value>
    </param>
  </params>
  </methodCall>
```

In this example, the first three lines (as below) constitute the HTTP Request Header, and the remaining part HTTP Request body.

```
POST    /cgi-bin/xmlrpc_server    HTTP/1.1
Content-Type: text/xml
Content-Length: xxx
```

In the first three lines of XML RPC message (as below), "slb_real" is the XML RPC method of the called command "**slb real** *<protocol> <name> <ip> [port] [maxconns] [hc_type] [hc_up] [hc_down]*". XML PRC method is embedded in a <methodName> tag (Please refer to Appendix III, in which all XML RPC methods supported by FortiBalancer are listed.).

```
<?xml version='1.0' ?>
<methodCall>
<methodName>slb_real</methodName>
```

The following part specifies the Enable mode and its password, which indicates the user will log in the Enable mode. "enable_password" is the keyword. The actual password value is embedded in a <string> tag. Enable password is included in every XML RPC message.

```
<member>
        <name>enable_passwd</name>
        <value>
          <string>****</string>
        </value>
</member>
```

This portion (as below) specifies the "protocol" parameter of the called "slb_real" method. "protocol" is the keyword， whose value is embedded in a <string> tag.

```
<member>
        <name>protocol</name>
        <value>
          <string>http</string>
        </value>
</member>
```

In this example, the parameters of the "slb_real" method include protocol, name, ip, port, maxconns, hctype, hcup and hcdown。Protocol, name and ip are required, while port, maxconns, hctype, hcup and hcdown are optional.

Note: In an HTTP Request, more than one XML RPC method can be called.

If the calling is successful, FortiBalancer will return an HTTP Response formatted in as follows:

```
<?xml version='1.0' ?>
```

```
<methodResponse>
  <params>
    <param>
      <value>
        <string>xmlrpc command successful</string>
      </value>
    </param>
  </params>
</methodResponse>
```

If the called command is a "**show**" command, its output will be displayed in the place of "xmlrpc command successful". If there is any error, the error is displayed.

To configure the XML PRC function on FortiBalancer, you need to configure two commands:

➢ Step 1 Turn on XML RPC

```
FortiBalancer1(config)#xml on https
```

➢ Step 2 Set the port for XML RPC to listen

```
FortiBalancer1(config)#xml port 9999
```

19.1.2.2.11 Remote Management

The Remote Management feature of the FortiBalancer appliance allows administrators to access remote devices via Telnet & SSH.

To use the Telnet feature on the FortiBalancer appliance, users can execute the command "**telnet** *"host port"*" as follows:

```
FortiBalancer#telnet "'172.16.2.182 -4'"
Trying 172.16.2.182...
Connected to 172.16.2.182 -4.
Escape character is '^]'.
Trying SRA secure login:
User (root): admin
Password:
[ SRA accepts you ]................succeed
```

To use the SSH feature on the FortiBalancer appliance, users can execute the command "**ssh remote** *"user@hostname"*" as follows:

```
FortiBalancer#ssh remote "root@172.16.85.240"
root@172.16.85.240's password:
Linux libh-server1 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686
GNU/Linux

Welcome to Ylmf_OS!
  * Information:    http://www.ylmf.com/

0 packages can be updated.
0 updates are security updates.

Last login: Wed Apr 20 00:39:35 2011 from 10.3.46.1
root@libh-server1:~#
```

19.1.2.2.12 FortiBalancer Flight Deck

The FortiBalancer appliance monitors a variety of useful statistics that provide a good indication of performance, user and network activity. The FortiBalancer appliance provides a graphical interface that can be used to easily monitor various statistics and get a comprehensive picture of the status of the FortiBalancer appliance. This graphical interface is called the Flight Deck.

The Flight Deck is an additional pop up browser window that, once set, can display a wide range of real time network operational data. Across the top of the browser window, you will discover readouts concerning the server health, request rate, cache hits and system usage. Moving to the left side of the window, you will find reading for the TCP, HTTP and SSL connections. The three connection figures sum up to total used "TCP pcb" displayed in the output of the "**show memory**" command. Sometimes, a pair of TCP connections is created for the same client request, e.g. an SLB client request normally will generate two connections, one is from the client to FortiBalancer appliance, and the other is from the FortiBalancer appliance to the server.

The central portion of the Flight Deck is occupied by two configurable graphs. Simply use the pull-down menu to choose the desired data you wish to track in the real time graphical output.

You can access the Flight Deck from the FortiBalancer appliance web UI by clicking the "Flight Deck" node at the bottom of the web UI Home configuration tree.

There exists two drop down menus above each graph. The first menu, called "Graph Type" contains a list of the statistics that can be displayed in the graph. Note that the list is identical for each graph. The second menu, called "Interval", is used to control the granularity of the time units shown on the horizontal axis of the graph, and how often the FortiBalancer appliance will update the graph. The default menu option is 5 seconds, which is also the smallest value that can be chosen. When the value is 5 seconds, the FortiBalancer appliance will update the graph display every 5 seconds, and the time will be shown on the horizontal axis in multiples of 5.

For some statistics, it makes sense to use a smaller interval. For example, it might be useful to see how the number of packets processed by the FortiBalancer appliance varies in 30 sec. intervals. On the other hand, you may want to view some statistics over a wider interval. For example, you may want to look at how the number of concurrent sessions varies from hour to hour, to get a feel for when most of your end users are logging in.

It is important to note that in order to view any of the statistics in the graphs, you must enable SNMP. This can be done via the web UI from the "Graph SNMP Monitoring" page under the "Admin Tools" node. Some of the statistics also require additional configuration, which will be described below.

Note: For the sake of security, it is strongly recommended to modify the default SNMP community string to avoid possible system information interception.

The following statistics are available for viewing in the graphs:

--------TCP--------

Active Opens

The number of times CLICKTCP connections have made to a direct transition to the SYN-SENT state from the CLOSED state.

Passive Opens

The number of times CLICKTCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.

Open Failures

The number of times CLICKTCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.

Established Conns

The number of CLICKTCP connections for which the current state is either ESTABLISHED or CLOSE- WAIT.

Resets Received

The number of times CLICKTCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

Resets Sent

The number of CLICKTCP segments sent containing the RST flag.

Retransmits

The total number of segments retransmitted **-** that is, the number of CLICKTCP segments transmitted containing one or more previously transmitted octets.

Packet Errors

The total number of segments received in error (e.g. bad CLICKTCP checksums)

--------IP---------

Packets Received

The number of IP packets received.

Packets Sent

The number of IP packets sent.

Bytes Received

The number of bytes received.

Bytes Sent

The number of bytes sent.

Header Errors

The number of IP packets with header errors.

Unknown Protocol

The number of IP packets with unknown IP protocol.

No Route Out

The number of IP packets with no route out.

--------UDP--------

Packets Received

The number of UDP packets received.

Packets Sent

The number of UDP packets sent.

Invalid Ports

The number of UDP packets with invalid ports.

Packet Errors

The number of UDP packets with packet error.

-------ICMP-------

Messages In

The number of ICMP message in.

Errors In

The number of ICMP errors message in.

Unreachable In

The number of unreachable ICMP messages.

Echoes In

The number of ICMP echoes in.

Echo Replies In

The number of ICMP echo replies in.

Messages Out

The number of ICMP message out.

Errors Out

The number of ICMP error message out.

Unreachable Out

The number of unreachable ICMP message out.

Echoes Out

The number of ICMP errors message out.

Echo Replies Out

The number of ICMP echo replies out.

-------CPU--------

% CPU Utilization

When this option is selected, the graph will represent what the percentage of the FortiBalancer appliance CPU that is utilized over time. For example, if the plot line on the graph is at the “75” mark on the graph, then the FortiBalancer appliance’s CPU is 75% utilized at that time, or is only 25% idle.

-------Proxy-Cache--------

% HTTP/HTTPS requests

The number of HTTP/HTTPS requests per second.

% Established Client Connections

The number of established client connections.

% Established Server Connections

The number of established server connections

-------Compression--------

% Bytes Received

The number of compression bytes received.

% Bytes Sent

The number of compression bytes sent.

19.2 Administrator Configuration and Privilege Management

19.2.1 Overview

The FortiBalancer appliance allows creating system administrators and specifying the access control level (Enable or Config) for administrators. If more flexible control over the administrator privilege is needed, the role-based privilege management can be used to control the CLI commands an administrator can execute.

19.2.2 System Administrator

The FortiBalancer appliance allows creating two types of administrator: administrator of the Enable level and administrator of the Config level. Administrators of the Enable level can execute only the commands allowed by the Enable and User levels. Administrators of the Config level can execute all the commands allowed by the Config, Enable and User levels.

19.2.3 Role-based Privilege Management

The role-based privilege management function can control the CLI commands that an administrator can execute by assigning roles to the administrator, thus realizing more flexible administrator privilege control.

One administrator can be assigned one or more roles, and the logic among the multiple roles is "OR". If any role assigned to an administrator permits the execution of a command, then the administrator can execute this command; if all roles assigned to an administrator deny (or none of the roles permits) the execution of a command, the administrator cannot execute this command. If an administrator is not assigned any role, the administrator is allowed to execute all the commands of his access control level.

Role is a set of privilege rules. A privilege rule consists two parts: rule string and operation privilege.

➢ **Rule String**

Rule string defines one or a group of command configurations. Three forms of rule string are supported:

- Complete form: slb real http 'r1' 172.16.2.250 80 1 tcp 3 3
- Incomplete form with some parameters: slb real http 'r1'
- Incomplete form with a part of the command body: slb real

To configure eligible rule strings, please pay attention to the following notes:

- The command body cannot be abbreviated.
- The parameters are case-sensitive.
- The whole rule string needs to be enclosed in double quotes. If double quotes are further needed by some parameters in the rule string, please use single quotes instead.
- If the entered string has multiple consecutive spaces, the spaces will be regarded as one space.

The system will check the command entered by the administrator against the rule strings from the first letter. If the command is identical with or comprises a rule string, the command is regarded as matching the rule string and will follow the rule.

➢ **Operation Privilege**

There are two kinds of operation privileges: "permit" and "deny". The privilege rules are consequently divided into "permit" rules and "deny" rules, which are respectively used to control that one command or a group of commands can or cannot be executed by certain roles. If a role is not configured with any "permit" rule, the role is not permitted to execute any command.

When a command matches a "permit" rule and a "deny" rule at the same time, the system will assume that the "deny" rule has a higher priority.

For example:

role1:

- "permit" rule: "no slb group"
- "deny" rule: "no slb group method"

The system will deny the execution of commands starting with "no slb group method" by role1, but the execution of other commands starting with "no slb group" is permitted.

role2:

- "permit" rule: "no slb group method"
- "deny" rule: "no slb group"

Because the "deny" rule has a higher priority, the system will deny the execution of the commands starting with "no slb group" by role2, although the "permit" rule allows the execution of the commands starting with "no slb group method".

To disallow a role from executing configuration, display and deletion operations about a feature (for example LLB), please configure the following privilege rule for this role:

- "deny" rule: "llb"
- "deny" rule: "no llb"
- "deny" rule: "show llb"
- "deny" rule: "clear llb"

Note:

➢ The system provides two pre-defined roles: "SLB" and "NETWORK". You can execute the "**show role predefined**" command to see the privilege rules of the two roles.

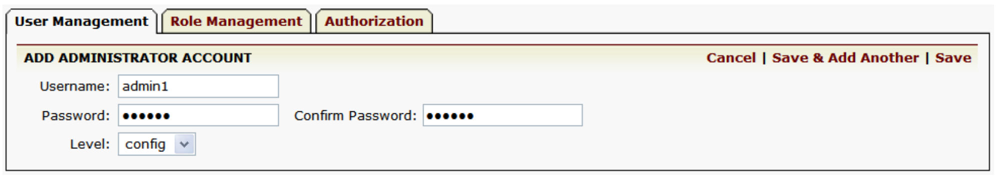
➢ Monitoring web UI requires the privilege for executing the "**show**" CLI commands. Please make sure the administrators who need to monitor a specific feature on web UI are assigned the role with the correspondent "**show**" privileges.

19.2.4 Configuration Examples

19.2.4.1 Creating System Administrator

➢ **web UI:**

Select **Admin Tools > User Management > User Management**, and click the **Add Admin** action link in the **Administrators** area. In the new configuration window, specify the required parameters in the **Add Administrator Account** area, and click the **Save** action link.



ADD ADMINISTRATOR ACCOUNT

Username:	admin1
Password:	••••••
Confirm Password:	••••••
Level:	config

➢ **CLI:**

Execute the following command to add an administrator account and specify the access control level:

```
user <user_name> <password> [enable|config]
```

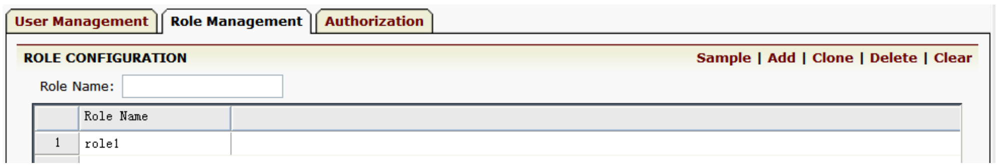
For example:

```
FortiBalancer(config)#user admin1 abcabc config
```

19.2.4.2 Configuring Role-based Privilege Management

➢ **web UI:**

1. Select **Admin Tools > User Management > Role Management**. Enter the role name in the **Role Name** text box in the **Role Configuration** area, and click the **Add** action link.

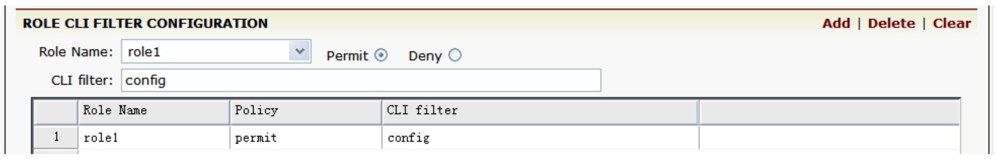


ROLE CONFIGURATION

Role Name:

	Role Name
1	role1

2. Select the **Role Management** tab, enter the rule string in the **CLI filter** text box and select the **Permit** or **Deny** radio button in the **Role CLI Filter Configuration** area, and click the **Add** action link.



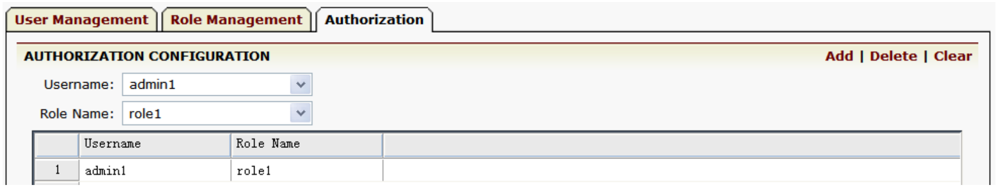
ROLE CLI FILTER CONFIGURATION

Role Name: role1 Permit [✓] Deny

CLI filter: config

	Role Name	Policy	CLI filter
1	role1	permit	config

3. Select the **Authorization** tab, specify the **Username** and **Role Name** parameters, and click the **Add** action link to assign the role to the administrator.



AUTHORIZATION CONFIGURATION

Username: admin1

Role Name: role1

	Username	Role Name
1	admin1	role1

➢ **CLI:**

1. Execute the following command to add a role:

```
role name <role_name>
```

For example:

```
FortiBalancer(config)#role name role1
```

2. Execute the following commands to configure the privilege rules for the role:

```
role deny <role_name> <filter_string>
role permit <role_name> <filter_string>
```

For example:

```
FortiBalancer(config)#role deny role1 “clear config”
FortiBalancer(config)#role permit role1 “show run config”
```

3. Execute the following command to assign the role to the administrator:

```
role user <user_name> <role_name>
```

For example:

```
FortiBalancer(config)#role user admin1 role1
```

Appendix I SNMP OID List

SNMP OID List	
.1.3.6.1.4.1.12356	This file defines the private CA SNMP MIB extensions.
.1.3.6.1.4.1.12356.3.1	Serial number of the FortiBalancer appliance.
.1.3.6.1.4.1.12356.4.1.0	Current system total available memory.
.1.3.6.1.4.1.12356.4.2.0	Current percentage of Network memory utilization.
.1.3.6.1.4.1.12356.16.1.1.0	Current status of the reverse proxy cache - on or off.
.1.3.6.1.4.1.12356.16.1.2.0	Total number of requests received by the reverse proxy cache.
.1.3.6.1.4.1.12356.16.1.3.0	Total GET requests received by the reverse proxy cache.
.1.3.6.1.4.1.12356.16.1.4.0	Total HEAD requests received by the reverse proxy cache.
.1.3.6.1.4.1.12356.16.1.5.0	Total PURGE requests received by the reverse proxy cache.
.1.3.6.1.4.1.12356.16.1.6.0	Total POST requests received by the reverse proxy cache.
.1.3.6.1.4.1.12356.16.1.7.0	Number of current client connections (e.g. from the browsers).
.1.3.6.1.4.1.12356.16.1.8.0	Number of current backend server connections.
.1.3.6.1.4.1.12356.16.1.9.0	Requests redirected to HTTPS.
.1.3.6.1.4.1.12356.16.1.10.0	Requests redirected based on regex match.
.1.3.6.1.4.1.12356.16.1.11.0	Requests forwarded with rewritten url.
.1.3.6.1.4.1.12356.16.1.12.0	Locations rewritten to HTTPS.
.1.3.6.1.4.1.12356.16.1.13.0	Locations rewritten based on regex match.
.1.3.6.1.4.1.12356.16.1.14.0	Cache skip, cache off.
.1.3.6.1.4.1.12356.16.1.15.0	We found the requested URL in the cache. The object was fresh and we did not have to revalidate. The object was served from our cache.
.1.3.6.1.4.1.12356.16.1.16.0	We got an IMS header in the request. We validated the timestamp and decided that the client's copy of this object is fresh. So we generated a 304 response and sent it out to the client.
.1.3.6.1.4.1.12356.16.1.17.0	Cache hit, reply with Precondition Failed.
.1.3.6.1.4.1.12356.16.1.18.0	The requested object was found in the cache. However, the request required revalidation (due to client generated revalidate, proxy generated revalidate or proxy generated forced miss).
.1.3.6.1.4.1.12356.16.1.19.0	The request does not result in a cache table search. Something in the request made us deem it non-cacheable (e.g. very long URL, a 'Cache-Control: no-store' header etc).
.1.3.6.1.4.1.12356.16.1.20.0	Count of times the cache table was searched, no matching entry was found and a new entry was created. However, note that sometimes, an entry is created temporarily (e.g. for an IMS request resulting in a 304) and is deleted after sending it out to the client (delayed delete).
.1.3.6.1.4.1.12356.16.1.21.0	Cache miss, create new entry, resp noncacheable.
.1.3.6.1.4.1.12356.16.1.22.0	Cache hit reply using cache + cache reply with 'not modified'.
.1.3.6.1.4.1.12356.18.1.1.0	Current maximum possible number of entries in the vrrpTable, which is 255 * (number of interfaces for which a cluster is defined). 255 is the max number of VIPs in a cluster.
.1.3.6.1.4.1.12356.18.1.2.0	Current number of entries in the vrrpTable.
.1.3.6.1.4.1.12356.18.1.3	A table containing clustering configuration.
.1.3.6.1.4.1.12356.18.1.3.1	An entry in the vrrpTable. Each entry represents a cluster VIP and not the cluster itself. If a cluster has n VIPs, then there will be n entries for the cluster in the vrrpTable (0 <= n <= 255). All the entries in the vrrpTable belonging to a single cluster will have the same values for all the fields except clusterVirIndex and clusterVirAddr.
.1.3.6.1.4.1.12356.18.1.3.1.1	The cluster virtual table index.
.1.3.6.1.4.1.12356.18.1.3.1.2	The cluster identifier.

SNMP OID List	
.1.3.6.1.4.1.12356.18.1.3.1.3	The current state of the cluster.
.1.3.6.1.4.1.12356.18.1.3.1.4	The interface name on which the cluster is defined.
.1.3.6.1.4.1.12356.18.1.3.1.5	A virtual ip address (VIP) in the cluster.
.1.3.6.1.4.1.12356.18.1.3.1.6	The IP address type of clusterVirAddress.
.1.3.6.1.4.1.12356.18.1.3.1.7	A virtual ip address (VIP) in the cluster.
.1.3.6.1.4.1.12356.18.1.3.1.8	Type of authentication being used. none(0) - no authentication simple-text-password(1) - use password specified in cluster virtual for authentication.
.1.3.6.1.4.1.12356.18.1.3.1.9	The password for authentication.
.1.3.6.1.4.1.12356.18.1.3.1.10	This is for controlling whether a higher priority Backup VRRP virtual preempts a low priority Master.
.1.3.6.1.4.1.12356.18.1.3.1.11	VRRP advertisement interval.
.1.3.6.1.4.1.12356.18.1.3.1.12	Priority of the local node in the cluster.
.1.3.6.1.4.1.12356.19.1.1.1.0	Number of real services currently configured.
.1.3.6.1.4.1.12356.19.1.1.2	A table containing the configuration of real services.
.1.3.6.1.4.1.12356.19.1.1.2.1	An rsTable entry containing the information of one real service.
.1.3.6.1.4.1.12356.19.1.1.2.1.1	The reference index for each real service.
.1.3.6.1.4.1.12356.19.1.1.2.1.2	The name of the real service.
.1.3.6.1.4.1.12356.19.1.1.2.1.3	The protocol of the real service.
.1.3.6.1.4.1.12356.19.1.1.2.1.4	The real service IP address.
.1.3.6.1.4.1.12356.19.1.1.2.1.5	The IP address type of rsIpAddress.
.1.3.6.1.4.1.12356.19.1.1.2.1.6	The real service IP address.
.1.3.6.1.4.1.12356.19.1.1.2.1.7	The port number of the real service.
.1.3.6.1.4.1.12356.19.1.1.2.1.8	Maximum number of connections per real service.
.1.3.6.1.4.1.12356.19.1.1.2.1.9	The current status of real service - up or down.
.1.3.6.1.4.1.12356.19.1.1.2.1.10	Server Average Response Time (in microseconds).
.1.3.6.1.4.1.12356.19.1.2.1.0	Number of virtual services currently configured.
.1.3.6.1.4.1.12356.19.1.2.2	A table containing the configuration of virtual services.
.1.3.6.1.4.1.12356.19.1.2.2.1	A vsTable entry containing the configuration of one virtual service.
.1.3.6.1.4.1.12356.19.1.2.2.1.1	Reference index for each virtual service.
.1.3.6.1.4.1.12356.19.1.2.2.1.2	Name of the virtual service.
.1.3.6.1.4.1.12356.19.1.2.2.1.3	The protocol of the virtual service.
.1.3.6.1.4.1.12356.19.1.2.2.1.4	The virtual service IP address.
.1.3.6.1.4.1.12356.19.1.2.2.1.5	The IP address type of vsIpAddress.
.1.3.6.1.4.1.12356.19.1.2.2.1.6	The virtual service IP address.
.1.3.6.1.4.1.12356.19.1.2.2.1.7	The port of the virtual service.
.1.3.6.1.4.1.12356.19.1.2.2.1.8	The max connection of virtual service.
.1.3.6.1.4.1.12356.19.1.3.1	Number of groups currently configured.
.1.3.6.1.4.1.12356.19.1.3.2	A table containing group members' configurations.
.1.3.6.1.4.1.12356.19.1.3.2.1	A gpTable entry containing one group member's configuration.
.1.3.6.1.4.1.12356.19.1.3.2.1.1	Reference index for each group member.
.1.3.6.1.4.1.12356.19.1.3.2.1.2	Name of the group.
.1.3.6.1.4.1.12356.19.1.3.2.1.3	Name of the real service.
.1.3.6.1.4.1.12356.19.1.3.2.1.4	Metric used to balance real services within the group.
.1.3.6.1.4.1.12356.19.2.1.1	Real service statistics table.
.1.3.6.1.4.1.12356.19.2.1.1.1	An rsStatsTable entry containing the statistics of one real service.
.1.3.6.1.4.1.12356.19.2.1.1.1.1	Reference index for each real service.
.1.3.6.1.4.1.12356.19.2.1.1.1.2	Name of the real service.
.1.3.6.1.4.1.12356.19.2.1.1.1.3	Real service IP address.
.1.3.6.1.4.1.12356.19.2.1.1.1.4	The IP address type of realAddress.
.1.3.6.1.4.1.12356.19.2.1.1.1.5	The real service IP address.
.1.3.6.1.4.1.12356.19.2.1.1.1.6	The port number of the real service.

SNMP OID List	
.1.3.6.1.4.1.12356.19.2.1.1.1.7	Number of outstanding requests to the real service.
.1.3.6.1.4.1.12356.19.2.1.1.1.8	Number of open connections to the real service.
.1.3.6.1.4.1.12356.19.2.1.1.1.9	The total number of requests sent to the real service.
.1.3.6.1.4.1.12356.19.2.1.1.1.10	The health status (up or down) of the real service.
.1.3.6.1.4.1.12356.19.2.2.1	A statistics table for virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1	A vsStatsTable entry containing the statistics of one virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.1	Reference index for each virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.2	Name of the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.3	IP address of the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.4	Port number of the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.5	Number of QoS URL policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.6	Number of QoS Hostname policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.7	Number of Persistent Cookie policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.8	Number of QoS Cookie hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.9	Number of Default policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.10	Number of Persistent URL policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.11	Number of Static policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.12	Number of QoS Network policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.13	Number of QoS URL policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.14	Number of Backup policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.15	Number of Cache hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.16	Number of Regex policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.17	Number of Rewrite Cookie policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.18	Number of Insert Cookie policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.19	Number of open connections to the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.22	Number of QoS Client Port policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.23	Number of QoS Body policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.24	Number of Header policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.25	Number of Hash URL policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.2.1.1.26	Number of Redirect policy hits for the virtual service.
.1.3.6.1.4.1.12356.19.2.3.1	A statistics table of the group.
.1.3.6.1.4.1.12356.19.2.3.1.1	A gpStatsTable entry containing the statistics of one group.
.1.3.6.1.4.1.12356.19.2.3.1.1.1	Reference index for each group.
.1.3.6.1.4.1.12356.19.2.3.1.1.2	Name of the group.
.1.3.6.1.4.1.12356.19.2.3.1.1.3	Total hits for the group.
.1.3.6.1.4.1.12356.20.1.2.0	Number of SSL hosts currently configured.
.1.3.6.1.4.1.12356.20.2.1.0	Total number of open SSL connections (all SSL hosts).
.1.3.6.1.4.1.12356.20.2.2.0	Total number of accepted SSL connections (all SSL hosts).
.1.3.6.1.4.1.12356.20.2.3.0	Total number of requested SSL connections (all SSL hosts).
.1.3.6.1.4.1.12356.20.2.4	SSL host statistics table.
.1.3.6.1.4.1.12356.20.2.4.1	sslTable entry for one SSL host.
.1.3.6.1.4.1.12356.20.2.4.1.1	The SSL table index.
.1.3.6.1.4.1.12356.20.2.4.1.2	Name of the SSL host.
.1.3.6.1.4.1.12356.20.2.4.1.3	Open SSL connections for SSL hostName.
.1.3.6.1.4.1.12356.20.2.4.1.4	Number of accepted SSL connections for SSL hostName.
.1.3.6.1.4.1.12356.20.2.4.1.5	Number of requested SSL connections for SSL hostName.
.1.3.6.1.4.1.12356.20.2.4.1.6	Number of resumed SSL sessions for SSL hostName.
.1.3.6.1.4.1.12356.20.2.4.1.7	Number of resumable SSL sessions for SSL hostName.
.1.3.6.1.4.1.12356.20.2.4.1.8	Number of session misses for SSL hostName.
.1.3.6.1.4.1.12356.22.1.0	Status of VIP statistics gathering - on or off.
.1.3.6.1.4.1.12356.22.2.0	The hostname that the VIP is representing (hostname of the appliance).
.1.3.6.1.4.1.12356.22.3.0	The current time in the format of MM/DD/YY HH:MM.

SNMP OID List	
.1.3.6.1.4.1.12356.22.4.0	Total number of ip packets received on all VIPs.
.1.3.6.1.4.1.12356.22.5.0	Total number of ip packets sent out on all VIPs.
.1.3.6.1.4.1.12356.22.6.0	Total number of IP bytes received on all VIPs.
.1.3.6.1.4.1.12356.22.7.0	Total number of IP bytes sent out on all VIPs.
.1.3.6.1.4.1.12356.22.8	A table of VIP statistics.
.1.3.6.1.4.1.12356.22.8.1	An entry in the ipStatsTable which is created for each VIP.
.1.3.6.1.4.1.12356.22.8.1.1	The VIP statistics table index.
.1.3.6.1.4.1.12356.22.8.1.2	The VIP address.
.1.3.6.1.4.1.12356.22.8.1.3	Total number of IP packets received on the VIP.
.1.3.6.1.4.1.12356.22.8.1.4	Total number of bytes received on the VIP.
.1.3.6.1.4.1.12356.22.8.1.5	Total number of packets sent out on the VIP.
.1.3.6.1.4.1.12356.22.8.1.6	Total number of bytes sent out on the VIP.
.1.3.6.1.4.1.12356.22.8.1.7	The time statistics gathering was enabled for the VIP.
.1.3.6.1.4.1.12356.23.1.0	The number of network interfaces presented on this system.
.1.3.6.1.4.1.12356.23.2.0	The total accumulated number of octets received on all the active interfaces (loopback is not included).
.1.3.6.1.4.1.12356.23.3.0	The total accumulated number of octets transmitted out on all the active interfaces (loopback is not included).
.1.3.6.1.4.1.12356.23.4	A table of interface statistics. The number of entries is given by the value of infNumber.
.1.3.6.1.4.1.12356.23.4.1	An infTable entry for one interface.
.1.3.6.1.4.1.12356.23.4.1.1	A unique value for each interface. Its value ranges between 1 and the value of infNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re- initialization.
.1.3.6.1.4.1.12356.23.4.1.2	Name of the interface.
.1.3.6.1.4.1.12356.23.4.1.3	The current operational state of the interface (up or down).
.1.3.6.1.4.1.12356.23.4.1.4	The interface's IP address.
.1.3.6.1.4.1.12356.23.4.1.5	The IP address type of infIpv4Address (should always be IPv4).
.1.3.6.1.4.1.12356.23.4.1.6	The interface's IPv4 address.
.1.3.6.1.4.1.12356.23.4.1.7	The IP address type of infIpv6Address (should always be IPv6).
.1.3.6.1.4.1.12356.23.4.1.8	The interface's IPv6 address.
.1.3.6.1.4.1.12356.23.4.1.9	The total number of octets received on the interface, including framing characters.
.1.3.6.1.4.1.12356.23.4.1.10	The number of packets, delivered by this sub-layer to a higher (sub-) layer, which were not addressed to a multicast or broadcast address at this sub-layer.
.1.3.6.1.4.1.12356.23.4.1.11	The number of packets, delivered by this sub-layer to a higher (sub-) layer, which were addressed to a multicast or broadcast address at this sub-layer.
.1.3.6.1.4.1.12356.23.4.1.12	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.
.1.3.6.1.4.1.12356.23.4.1.13	For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. For character- oriented or fixed-length interfaces, the number of inbound transmission units that contained errors preventing them from being deliverable to a higher-layer protocol.
.1.3.6.1.4.1.12356.23.4.1.14	For packet-oriented interfaces, the number of packets received via the interface which was discarded because of an unknown or unsupported protocol. For character-oriented or fixed-length

SNMP OID List	
	interfaces that support protocol multiplexing the number of transmission units received via the interface which were discarded because of an unknown or unsupported protocol. For any interface that does not support protocol multiplexing, this counter will always be 0.
.1.3.6.1.4.1.12356.23.4.1.15	The total number of octets transmitted out of the interface, including framing characters.
.1.3.6.1.4.1.12356.23.4.1.16	The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.
.1.3.6.1.4.1.12356.23.4.1.17	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.
.1.3.6.1.4.1.12356.23.4.1.18	For packet-oriented interfaces, the number of outbound packets that could not be transmitted because of errors. For character-oriented or fixed-length interfaces, the number of outbound transmission units that could not be transmitted because of errors.
.1.3.6.1.4.1.12356.23.4.1.19	The total number of octets received on the interface, including framing characters.
.1.3.6.1.4.1.12356.23.4.1.20	The number of packets, delivered by this sub-layer to a higher (sub-) layer, which were not addressed to a multicast or broadcast address at this sub-layer.
.1.3.6.1.4.1.12356.23.4.1.21	The number of packets, delivered by this sub-layer to a higher (sub-) layer, which were addressed to a multicast or broadcast address at this sub-layer.
.1.3.6.1.4.1.12356.23.4.1.22	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.
.1.3.6.1.4.1.12356.23.4.1.23	For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. For character- oriented or fixed-length interfaces, the number of inbound transmission units that contained errors preventing them from being deliverable to a higher-layer protocol.
.1.3.6.1.4.1.12356.23.4.1.24	For packet-oriented interfaces, the number of packets received via the interface which were discarded because of an unknown or unsupported protocol. For character-oriented or fixed-length interfaces that support protocol multiplexing the number of transmission units received via the interface which were discarded because of an unknown or unsupported protocol. For any interface that does not support protocol multiplexing, this counter will always be 0.
.1.3.6.1.4.1.12356.23.4.1.25	The total number of octets transmitted out of the interface, including framing characters.
.1.3.6.1.4.1.12356.23.4.1.26	The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.
.1.3.6.1.4.1.12356.23.4.1.27	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a

SNMP OID List	
	multicast or broadcast address at this sub-layer, including those that were discarded or not sent.
.1.3.6.1.4.1.12356.23.4.1.28	For packet-oriented interfaces, the number of outbound packets that could not be transmitted because of errors. For character-oriented or fixed-length interfaces, the number of outbound transmission units that could not be transmitted because of errors.
.1.3.6.1.4.1.12356.24.1.1.0	The number of syslog notifications that have been sent. This number may include notifications that were prevented from being transmitted due to reasons such as resource limitations and/or non-connectivity. If one is receiving notifications, one can periodically poll this object to determine if any notifications were missed. If so, a poll of the logHistoryTable might be appropriate.
.1.3.6.1.4.1.12356.24.1.2.0	Indicates whether logMessageGenerated notifications will or will not be sent when a syslog message is generated by the device. Disabling notifications does not prevent syslog messages from being added to the logHistoryTable.
.1.3.6.1.4.1.12356.24.1.3.0	Indicates which syslog severity levels will be processed. Any syslog message with a severity value greater than this value will be ignored by the agent. note: severity numeric values increase as their severity decreases, e.g. error(3) is more severe than debug(7).
.1.3.6.1.4.1.12356.24.2.1.0	The upper limit on the number of entries that the logHistoryTable may contain. A value of 0 will prevent any history from being retained. When this table is full, the oldest entry will be deleted and a new one will be created.
.1.3.6.1.4.1.12356.24.2.2	A table of syslog messages generated by this device. All 'interesting' syslog messages (i.e. severity <= logMaxSeverity) are entered into this table.
.1.3.6.1.4.1.12356.24.2.2.1	A syslog message that was previously generated by this device. Each entry is indexed by a message index.
.1.3.6.1.4.1.12356.24.2.2.1.1	A monotonically increasing integer for the sole purpose of indexing messages. When it reaches the maximum value the agent flushes the table and wraps the value back to 1.
.1.3.6.1.4.1.12356.24.2.2.1.2	The severity of the message.
.1.3.6.1.4.1.12356.24.2.2.1.3	The text of the message. If the text of the message exceeds 255 bytes, the message will be truncated to 254 bytes and a '*' character will be appended, indicating that the message has been truncated.
.1.3.6.1.4.1.12356.24.3.1	If the fastlog level is larger than or equal to “error” (including emerg, alert, crit, err), this trap will be sent. Some examples are shown below: 1. Execute "ifconfig em0 up/down" 2. One of the dual power supplies failed/recovered 3. Execute "log test" 4. CPU fan failed/recovered 5. CPU overheated
.1.3.6.1.4.1.12356.25.1.0	The number of times ClickTCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.
.1.3.6.1.4.1.12356.25.2.0	The number of times ClickTCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.
.1.3.6.1.4.1.12356.25.3.0	The number of times ClickTCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP

SNMP OID List	
	connections have made a direct transition to the LISTEN state from the SYN-RCVD state.
.1.3.6.1.4.1.12356.25.4.0	The number of times ClickTCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.
.1.3.6.1.4.1.12356.25.5.0	The number of ClickTCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.
.1.3.6.1.4.1.12356.25.6.0	The total number of ClickTCP segments received, including those received in error. This count includes segments received on currently established connections.
.1.3.6.1.4.1.12356.25.7.0	The total number of ClickTCP segments sent, including those on current connections but excluding those containing only retransmitted octets.
.1.3.6.1.4.1.12356.25.8.0	The total number of segments retransmitted - that is, the number of ClickTCP segments transmitted containing one or more previously transmitted octets.
.1.3.6.1.4.1.12356.25.9.0	The total number of segments received in error (e.g., bad ClickTCP checksums).
.1.3.6.1.4.1.12356.25.10.0	The number of ClickTCP segments sent containing the RST flag.
.1.3.6.1.4.1.12356.25.11	A table containing ClickTCP connection-specific information.
.1.3.6.1.4.1.12356.25.11.1	A conceptual row of the ctcpConnTable containing information about a particular current TCP connection. Each row of this table is transient, in that it ceases to exist when (or soon after) the connection makes the transition to the CLOSED state.
.1.3.6.1.4.1.12356.25.11.1.1	A unique value for each clicktcp connection.
.1.3.6.1.4.1.12356.25.11.1.2	The state of this TCP connection.
.1.3.6.1.4.1.12356.25.11.1.3	The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used.
.1.3.6.1.4.1.12356.25.11.1.4	The IP address type of ctcpConnLocalAddress.
.1.3.6.1.4.1.12356.25.11.1.5	The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0/:: is used.
.1.3.6.1.4.1.12356.25.11.1.6	The local port number for this TCP connection.
.1.3.6.1.4.1.12356.25.11.1.7	The remote IP address for this TCP connection.
.1.3.6.1.4.1.12356.25.11.1.8	The IP address type of ctcpConnRemAddress.
.1.3.6.1.4.1.12356.25.11.1.9	The remote IP address for this TCP connection.
.1.3.6.1.4.1.12356.25.11.1.10	The remote port number for this TCP connection.
.1.3.6.1.4.1.12356.27.1.1.0	The number of real services being checked.
.1.3.6.1.4.1.12356.27.1.2	Health Check statistics table.
.1.3.6.1.4.1.12356.27.1.2.1	An hcStatsTable entry containing health check statistics for one real service.
.1.3.6.1.4.1.12356.27.1.2.1.1	Reference index for each real service being checked.
.1.3.6.1.4.1.12356.27.1.2.1.2	Real service name.
.1.3.6.1.4.1.12356.27.1.2.1.3	Health Check IP address.
.1.3.6.1.4.1.12356.27.1.2.1.4	The IP address type of hcAddress.
.1.3.6.1.4.1.12356.27.1.2.1.5	The health Check IP address.
.1.3.6.1.4.1.12356.27.1.2.1.6	Health Check port.
.1.3.6.1.4.1.12356.27.1.2.1.7	The status (UP/DOWN) of the health check.
.1.3.6.1.4.1.12356.27.1.2.1.8	The reason why the health check is being marked UP/DOWN.
.1.3.6.1.4.1.12356.27.1.2.1.9	The number of times the health check is down.

SNMP OID List	
.1.3.6.1.4.1.12356.27.1.2.1.10	The number of times the health check is up.
.1.3.6.1.4.1.12356.27.1.2.1.11	The number of connections attempted.
.1.3.6.1.4.1.12356.27.1.2.1.12	The number of successful connections.
.1.3.6.1.4.1.12356.27.1.2.1.13	The number of connection failures.
.1.3.6.1.4.1.12356.28.1.0	Total number of bytes received.
.1.3.6.1.4.1.12356.28.2.0	Total number of bytes sent.
.1.3.6.1.4.1.12356.28.3.0	Number of bytes received per second.
.1.3.6.1.4.1.12356.28.4.0	Number of bytes sent per second.
.1.3.6.1.4.1.12356.28.5.0	Peak received bytes per second.
.1.3.6.1.4.1.12356.28.6.0	Peak sent bytes per second.
.1.3.6.1.4.1.12356.28.7.0	Number of currently active transaction.
.1.3.6.1.4.1.12356.30.1.0	Current percentage of CPU utilization.
.1.3.6.1.4.1.12356.30.2.0	Number of connections per second.
.1.3.6.1.4.1.12356.30.3.0	Number of requests per second.
.1.3.6.1.4.1.12356.31.1.0	Total DNS requests.
.1.3.6.1.4.1.12356.31.2.0	Total successful DNS resolvings.
.1.3.6.1.4.1.12356.31.3.0	Total failed DNS resolvings.
.1.3.6.1.4.1.12356.31.4.0	Total DNS requests in the last second.
.1.3.6.1.4.1.12356.31.5.0	Total successful DNS resolvings in the last second.
.1.3.6.1.4.1.12356.31.6.0	Total failed DNS resolvings in the last second.
.1.3.6.1.4.1.12356.31.7.0	Peak DNS requests in a second.
.1.3.6.1.4.1.12356.31.8.0	Peak successful DNS resolvings in a second.
.1.3.6.1.4.1.12356.31.9.0	Total DNS requests in the last minute.
.1.3.6.1.4.1.12356.31.10.0	Total successful DNS resolvings in the last minute.
.1.3.6.1.4.1.12356.31.11.0	Total failed DNS resolvings in the last minute.
.1.3.6.1.4.1.12356.31.12.0	Peak DNS requests in a minute.
.1.3.6.1.4.1.12356.31.13.0	Peak successful DNS resolvings in a minute.
.1.3.6.1.4.1.12356.31.14.0	Total DNS requests in the last hour.
.1.3.6.1.4.1.12356.31.15.0	Total successful DNS resolvings in the last hour.
.1.3.6.1.4.1.12356.31.16.0	Total failed DNS resolvings in the last hour.
.1.3.6.1.4.1.12356.31.17.0	Peak DNS requests in an hour.
.1.3.6.1.4.1.12356.31.18.0	Peak successful DNS resolvings in an hour.
.1.3.6.1.4.1.12356.31.19.0	Total DNS requests in the last day.
.1.3.6.1.4.1.12356.31.20.0	Total successful DNS resolvings in the last day.
.1.3.6.1.4.1.12356.31.21.0	Total failed DNS resolvings in the last day.
.1.3.6.1.4.1.12356.31.22.0	Peak DNS requests in a day.
.1.3.6.1.4.1.12356.31.23.0	Peak successful DNS resolvings in a day.
.1.3.6.1.4.1.12356.31.24.0	Total DNS requests in the last 5 seconds.
.1.3.6.1.4.1.12356.31.25.0	Total successful DNS resolvings in the last 5 seconds.
.1.3.6.1.4.1.12356.31.26.0	Total failed DNS resolvings in the last 5 seconds.
.1.3.6.1.4.1.12356.31.27.0	Peak DNS requests in 5 seconds.
.1.3.6.1.4.1.12356.31.28.0	Peak successful DNS resolvings in 5 seconds.
.1.3.6.1.4.1.12356.32.1	Current temperature of CPU and the system.
.1.3.6.1.4.1.12356.32.2	Current fan speed.
.1.3.6.1.4.1.12356.32.3	Current dual power supply state: “0” means OK and “1” means error.
.1.3.6.1.4.1.12356.251.1	Execute "snmp on" and save configuration, and then reboot the system. When the system boots up, this trap will be sent.
.1.3.6.1.4.1.12356.251.2	Execute "snmp on", and reboot the system. When the system is shut down, this trap will be sent.
.1.3.6.1.4.1.12356.251.3	License remaining days. If the current license key is going be expired in 15 days, this trap will be sent per day.
Float	A single precision floating-point number. The semantics and encoding are identical for type 'single' defined in IEEE Standard

SNMP OID List	
	for Binary Floating-Point, ANSI/IEEE Std 754-1985. The value is restricted to the BER serialization of the following ASN.1 type: FLOATTYPE ::= [120] IMPLICIT FloatType (note: the value 120 is the sum of '30'h and '48'h) The BER serialization of the length for values of this type must use the definite length, short encoding form. For example, the BER serialization of value 123 of type FLOATTYPE is '9f780442f60000'h. (The tag is '9f78'h; the length is '04'h; and the value is '42f60000'h.) The BER serialization of value '9f780442f60000'h of data type Opaque is '44079f780442f60000'h. (The tag is '44'h; the length is '07'h; and the value is '9f780442f60000'h.
Synlogseverity	The severity of a syslog message. The enumeration values are equal to the values that syslog uses + 1. For example, with syslog, emergency=0.

Appendix II Abbreviations

Acronym	Full Spelling
AAA	Authentication, Authorization & Accounting
ACL	Access Control List
ADC	Application Delivery Controller
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
CA	Certificate Authority
CDN	Content Distribution Network
CDP	CRL Distribution Point
CGI	Common Gateway Interface
CLI	Command Line Interface
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CRC	Cyclic Redundancy Check
DMZ	DeMilitarized Zone
DNS	Domain Name Service
DoS	Denial Of Service
DPS	Dynamic Proximity System
FFO	Fast Failover
FIFO	First-In First-Out
FTP	File Transfer Protocol
FTPS	FTP over SSL
GMT	Greenwich Mean Time
GRE	Generic Routing Encapsulation
GSLB	Global Server Load Balance (also known as SDNS)
HC	Health Check
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Sockets Layer
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIS	Internet Information Server
IMS	Information Management System
IP	Internet Protocol
ISP	Internet Service Provider
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LED	Light Emitting Diode
LLB	Link Load Balancing
Local DNS	Local Domain Name Service
MAC	Media Access Control
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extensions
MNET	Multi-Netting
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NIC	Network Interface Card
NMS	Network Management Station
OCSP	Online Certificate Status Protocol

Acronym	Full Spelling
OID	Object Identifier
OSI	Open System Interconnection
OSPF	Open Shortest Path First
OWA	Outlook Web Access
PCI	Peripheral Component Interface
PEM	Privacy Enhanced Mail
PHY	Physical Layer
PKI	Public Key Infrastructure
POP3	Post Office Protocol - Version 3
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
PST	Pacific Standard Time
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RAM	Random Access Memory
RDP	Remote Desktop Protocol
RFC	Request For Comments
RIP	Routing Information Protocol
RIPv2	Routing Information Protocol version 2
RTSP	Real Time Streaming Protocol
RTS	Return to Sender
SCP	Session Control Protocol
SDNS	Smart DNS (SDNS, also known as GSLB)
SIP	Session Initiation Protocol
SLB	Server Load Balancing
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer
TACACS	Terminal Access Controller Access Control System
TCI	Tag Control Information
TCP	Transmission Control Protocol
TCPS	TCP with SSL
TELNET	Terminal Emulation Protocol in a TCP/IP Environment
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security Protocol
TPID	Tag Protocol Identifier
TTL	Time to Live
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VCID	Virtual Cluster ID
VIP	Virtual IP
VLAN	Virtual Local Area Network
VOD	Video On Demand
VoIP	Voice over Internet Protocol
VRRP	Virtual Router Redundancy Protocol
Web UI	Web User Interface
WELF	WebTrends Enhanced Log Format

Appendix III XML RPC Methods

The following table lists all XML RPC methods supported by FortiBalancer. The default value of every parameter of the XML RPC methods are the same as the default value of every parameter of the corresponding called commands.

Generic XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
fortibalanceros_cli_enable	All commands in Enable mode	{num, int}, {cli_string0, string}, {cli_string1, string}, {cli_string2, string}, {cli_string3, string} ……	num	If no "num" value is given, it defaults to 1 and "cli_string0" should be configured. The names of CLI strings should start from "cli_string0" and end at "cli_string{n-1}". If some intermediate CLI strings are missing, the XML RPC system will just ignore and not complain.
fortibalanceros_cli_config	All commands in Config mode			
fortibalanceros_cli_config_with_input	All commands in Config mode	{cli_string, string}, {num, int}, {input_string0, string}, {input_string1, string} ...	num, input_string0, input_string1, ...	If no "num" value is given, it defaults to 1. "input_string" starts from "input_string0" to "input_string(num-1)". If "input_string(n)" is not included, it defaults to null. However, if the CLI must require a valid input at this place, the invocation may be hung or an error may be returned. You can use this method to execute only one command once.

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
slb_real	slb real	{protocol, string}, {name, string}, {ip, string}, {port, int}, {maxconns, int}, {hctype, string}, {hcup, int}, {hcdown, int}, {sess_timeout, int}	port maxconns hctype hcup hcdown、sess_timeout	"sess_timeout" parameter is valid for UDP real services only. "name" is the name of real service.
no_slb_real	no slb real	{protocol, string}, {name, string}		
slb_virtual	slb virtual	{protocol, string}, {name, string}, {ip, string}, {port, int}, {noarp,		"noarp" is of type integer. "name" is the name of virtual service.

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
		int}		
no_slb_virtual	no slb virtual	{protocol, string}, {name, string}		
slb_group_method	slb group method	{name, string}, {method, string}, {threshold, int}, {threshold_rr, int}, {cookie, string}, {path_attr, int}, {offset, int}, {header, string}, {sess_timeout, int}	Method threshold threshold_rr cookie path_attr offset header sess_timeout	"threshold" and "threshold_rr" are optional parameters for "lc" and "sr" methods. "cookie" and "path_attr" are optional parameters for "ic" method. "cookie" is required and "offset" is optional for "rc" method. "header" is a required parameter for "hh" method. "sess_timeout" is an optional parameter for "sslsid" method. Parameters in the XML RPC message which do not apply to the given method are ignored.
no_slb_group_method	no slb group method	{name, string}		
slb_group_member	slb group member	{name, string}, {real_name, string} {value, int}	value	"name" is the name of the group and "real_name" is the name of real service. "value" is an optional parameter and applies only to certain methods.
no_slb_group_member	no slb group member	{name, string}, {real_name, string}		"name" is the name of group and "real_name" is the name of real_service.
slb_policy	slb policy	{policy, string}, {name, string}, {virtual_name, string}, {group_name, string}, {real_name, string}{regex, string}, {policy_string, string}, {precedence, int}, {ip,string}, {netmask, string}		"policy" is the type of policy. The values of "policy" can be one of "static", "default", "regex", "icookie", "rcookie", "persistent url", "persistent cookie", "qos cookie", "qos hostname", "qos url", and "qos network". "policy" and "virtual_name" are required for all policies. "group_name" is required for all policies except "static" policy. "name" is required for all policies except "static" and "default". "real_name" is required only for "static" policy. "precedence" is required for "regex", "icookie",

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
				"rcookie", "persistent url", "persistent cookie", "qos cookie", and "qos hostname", "qos url" and "qos network". "regex" is required for "regex" policy. "policy_string" is required for "persistent url", "persistent cookie", "qos url", "qos cookie", and "qos hostname". "ip" and "netmask" are required for "qos network" policy.
no_slb_poli cy	no slb policy	{policy, string}, {name, string}, {virtual_name, string}		policy" is the type of policy, "name" is the name of policy and "virtual_name" is the name of "virtual_service". "virtual_name" is required for "default" and "static" policies and "name" is required for the rest of them.
cache_evict	cache evict	{hostname, string}, {regex, string}		
cluster_virt ual_on	cluster virtual on	{vcid, int}, {interface, string}	Vcid interface	
cluster_virt ual_off	cluster virtual off	{vcid, int}, {interface, string}	Vcid interface	
cluster_virt ual_ifname	cluster virtual ifname	{vcid, int}, {interface, string}		
cluster_virt ual_vip	cluster virtual vip	{vcid, int}, {interface, string}, {vip, string}		
no_cluster_ virtual_vip	no cluster virtual vip	{vcid, int}, {interface, string}, {vip, string}		
cluster_virt ual_prio	cluster virtual prio	{vcid, int}, {interface, string}, {vprio, int}, {node_id, int}	node_id	
no_cluster_ virtual_prio	no cluster virtual prio	{vcid, int}, {interface, string}, {vprio, int}, {node_id, int}	node_id	
cluster_virt ual_preemp t	cluster virtual preempt	{vcid, int}, {interface, string}, {preempt, int}		"preempt" value should be 0 or 1.
no_cluster_ virtual_pree mpt	no cluster virtual preempt	{vcid, int}, {interface, string}		
cluster_virt ual_interval	cluster virtual interval	{vcid, int}, {interface, string},	interval	"interval" value should be between 3 and 10 and the

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
		{interval, int}		default value is 5.
no_cluster_virtual_interval	no cluster virtual interval	{vcid, int}, {interface, string}		
cluster_virtual_auth	cluster virtual auth	{vcid, int}, {interface, string}, {auth, int}, {auth_passwd, string}		“auth” value should be 0 or 1. If it is set to 1, “auth_passwd” parameter is required.
no_cluster_virtual_auth	no cluster virtual auth	{vcid, int}, {interface, string}		