FBLOS FortiBalancer 8.1.1

User Guide

All Rights Reserved

Warning: Modifications made to the Fortinet unit, unless expressly approved by Fortinet, Inc., could void the user's authority to operate the equipment.

Declaration of Conformity

We, Fortinet, Inc., 1090 Kifer Road Sunnyvale, CA 94086; declare under our sole responsibility that the product(s) Fortinet, Inc., FortiBalancer appliance complies with Part 15 of FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Warning: This is a Class A digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy, and if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. In a residential area, operation of this equipment is likely to cause harmful interference in which case the user may be required to take adequate measures or product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Contacting Fortinet

Please use the following information to contact us at Fortinet:

Website:	http://www.fortinet.com
Email:	Please go to https://support.fortinet.com
Tel:	Please go to https://support.fortinet.com

Telephone access to Fortinet, Inc. is available Monday through Friday, 6 A.M. to 6 P.M. PST.

Address: Fortinet, Inc.

1090 Kifer Road Sunnyvale, CA 94086

Table of Contents

Chapter 1 Initial System Setup & Configuration

1.1 Overview

This section will outline the basic set up and configuration of the FortiBalancer appliance. For the physical installation of the FortiBalancer appliance and the emulation settings of the console connected to this Appliance, the easy to follow setup steps are outlined below.

The following is a breakdown of the initial set up and configuration features for the Fortinet Appliance.

1.1.1 Console Connection

Please make certain that your console is set up as follows:

Table 1-1 Console Specifications

Emulation to VT100
Baud at 9600
No. of Bits to 8
No Parity
Stop Bits to 1
No Flow Control

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

Over the next several pages, we will discuss the initial deployment of the FortiBalancer and basic configuration strategies to seamlessly integrate your network architecture with the additional power of the FortiBalancer appliance from Fortinet.

1.1.1.1 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 using SSH connection. Below gives an example.

If you require SSH software for Windows, MacOS or UNIX, it is available on-line at http://www.openssh.com. Freeware and links to other support sites can be found here.

To establish an SSH connection:

→ Step1 Run the SSH program on you workstation

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

10.3.55.251 is the FortiBalancer appliance’s IP address.

→ Step2 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 a login and a password. The default/first time login is “**admin**”. The first time password is “**admin**”.

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

1.1.2 Reading the LED

The FortiBalancer appliance possesses 3 LEDs: one yellow, one green and one blue. The following is the usage description of each LED.

Table 1-2 LED Usage Description

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: 1. FortiBalancer appliance’s CPU fan stops working. 2. FortiBalancer appliance’s CPU is overheated. 3. FortiBalancer Appliance system is overheated. 4. 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.

Attention: If the yellow LED turns on, please contact Customer Support. You can visit syslog server to get more information about the problem.

1.1.3 The FortiBalancer 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 1-3 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 1-4 FortiBalancer CLI Style Conventions

Style	Convention
Bold typeface	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**".
{ x | y | … }	Alternative items are grouped in braces and separated by vertical bars. One should be selected.
[x | y | …]	Optional alternative items are grouped in square brackets and separated by vertical bars. One or none is selected.

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.

For example:

[no|show|clear] ip address *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> <netmask>*

1.1.3.2 Levels of Access Control

The FortiBalancer appliance's Command Line Interface offers three levels of configuration and access to the FBLOS. 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 such as **ping** and **traceroute**. Here is how the **User Level** prompt appears in the CLI.

```
FBL>
```

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 FBL> to FBL#, which means the user is granted access to the Enable Level.

```
FBL>enable
Enable password:
FBL#
```

The final access level for the FBLOS is the **Configure Level**. It is with this level of authority that the user can make changes to the configuration of the box one session at a time. No two users can access the **Configure 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:

```
FBL#config terminal
```

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

```
FBL(config)#
```

In the event that Config Mode is not available due to the fact that another configure level session has been opened, the administrator may deploy the following command to gain the **Configure Level**:

```
FBL#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":
```

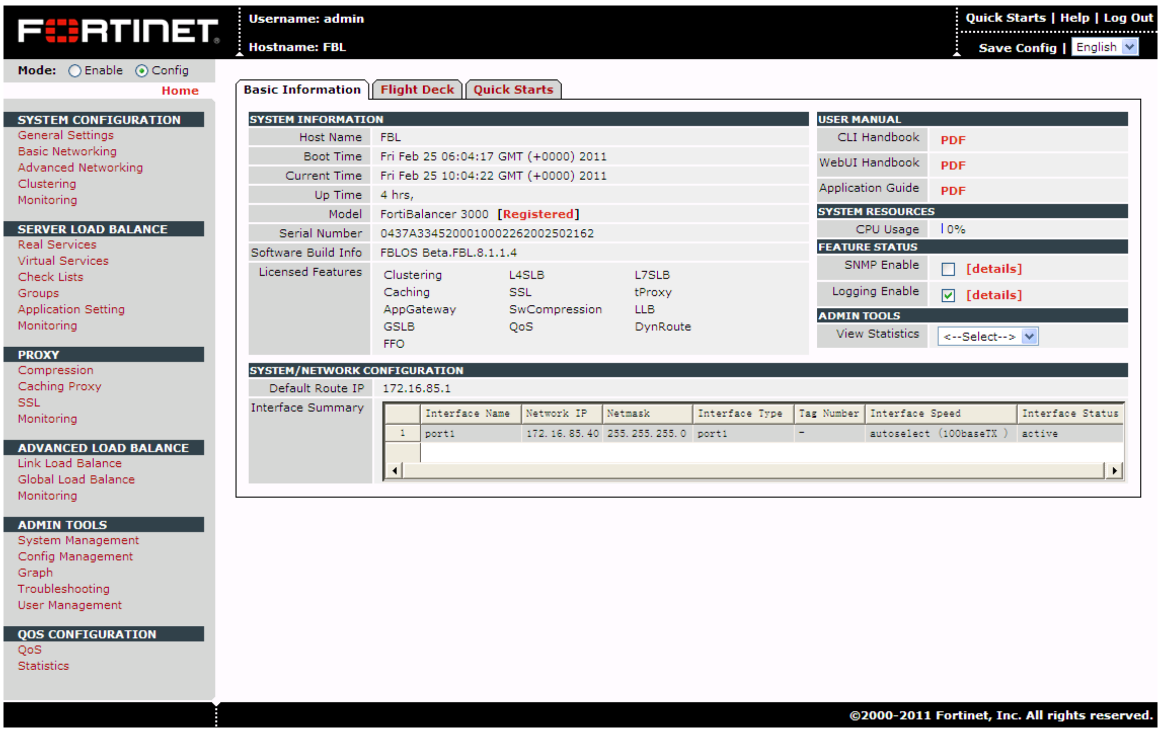
For each level the user can type “?” for available commands. For example, entering “FBL(config)#**slb real ?**” will prompt the FortiBalancer to list the possible parameters or protocols the CLI will accept with the “**slb real**” command.

```
FBL(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
…
```

1.1.4 FortiBalancer Web User’s Interface (WebUI)

This section introduces FortiBalancer Web User’s Interface (WebUI). The FortiBalancer Web User’s Interface (WebUI) can:

- Improve user experience with fast response time
- Maximize the functionality and performance of the FortiBalancer appliance
- Simplify system management
- Support three language: English, Chinese and Japanese



SYSTEM INFORMATION

Host Name	FBL
Boot Time	Fri Feb 25 06:04:17 GMT (+0000) 2011
Current Time	Fri Feb 25 10:04:22 GMT (+0000) 2011
Up Time	4 hrs,
Model	FortiBalancer 3000 [Registered]
Serial Number	0437A3345200010002262002502162
Software Build Info	FBLOS Beta.FBL.8.1.1.4
Licensed Features	Clustering, L4SLB, L7SLB, Caching, SSL, tProxy, AppGateway, SwCompression, LLB, GSLB, QoS, DynRoute, FFO

USER MANUAL

CLI Handbook	PDF
WebUI Handbook	PDF
Application Guide	PDF

SYSTEM RESOURCES

CPU Usage	0%

FEATURE STATUS

SNMP Enable	☐ [details]
Logging Enable	✓ [details]

ADMIN TOOLS

View Statistics	<--Select-->

SYSTEM/NETWORK CONFIGURATION

Default Route IP 172.16.85.1

Interface Summary

	Interface Name	Network IP	Netmask	Interface Type	Tag Number	Interface Speed	Interface Status
1	port1	172.16.85.40	255.255.255.0	port1	-	autoselect (100baseTX)	active

©2000-2011 Fortinet, Inc. All rights reserved.

Figure 1-1

1.2 General Settings Configuration

Now that you are in the **configure mode**, it’s time to assign Outside, Inside 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 basic network architecture.

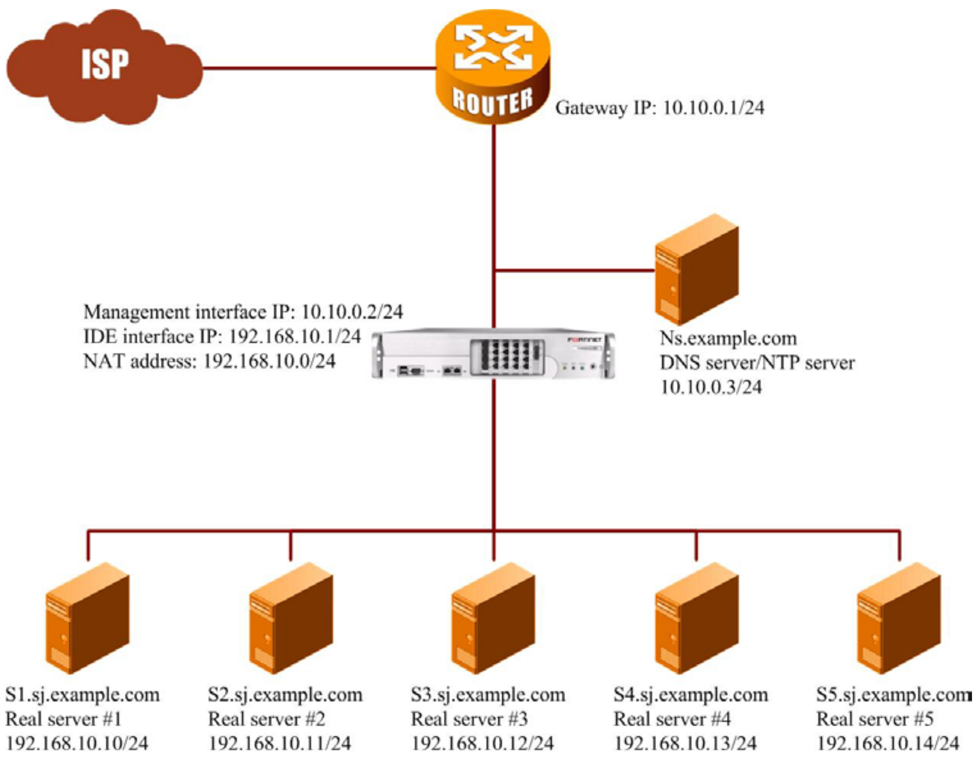


Figure 1-2 Basic Network Architecture

The most critical pieces of information from the figure above:

IP ADDRESS	DESCRIPTION
10.10.0.1/24	Gateway IP Address
10.10.0.2/24	Management IP Address
192.168.10.1/24	Inside 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 1-5 General Settings for FortiBalancer Appliance

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname}* *<ip_address> <netmask>*
Configure gateway IP address	**ip route default** <*gateway_ip*>
View IP configurations	**ping** *{ip|hostname}* **show ip address** **show ip route**
Set up WebUI	**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 Sample Configuration for General Settings via CLI

1.2.2.1 Assigning the IP Address for Interfaces

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

```
FBL(config)#ip address outside 10.10.0.2 255.255.255.0
FBL(config)#ip address inside 192.168.10.1 255.255.255.0
```

The outside interface and the inside 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.

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. Please refer to the chapter Link Load Balancing for details about LLB methods and policies.

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:

```
FBL(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:

```
FBL(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:

```
FBL(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:

```
FBL(config)#show ip address
ip address "outside" 10.10.0.2 255.255.255.0
ip address "inside" 192.168.10.1 255.255.255.0

FBL(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 outside**", will remove the outside IP address for you to then reenter the correct information.

1.2.2.4 Setting up the WebUI

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

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

```
FBL(config)#webui on
```

It's 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, you can designate the WebUI 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 outside interface and WebUI port.

The FortiBalancer appliance WebUI supports the following browsers:

- IE (Recommended)
- Chrome
- Opera
- Netscape
- Firefox

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

1.2.2.5 Assigning the Host Name

With Fortinet' clustering technology, more than one FortiBalancer appliance may be used within a single network server farm. With this in mind, the FBLOS 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 "admin" to the newly assigned name:

```
FBL(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 calls upon at reboot.

Chapter 2 Advanced Network Configuration

2.1 Overview

This section focuses on intrducing the advanced network configurations, including VLAN, MNET, Port Forwarding, NAT and Dynamic Routing functionalities and configurations on FortiBalancer appliance. The FortiBalancer appliance supports **Tag VLAN** (Virtual Local Area Network) on all interfaces (port1, port2, port3, port4, …, port14).

2.2 Understanding VLAN, MNET, Port Forwarding, Dynamic Routing and NAT

2.2.1 VLAN

A VLAN 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 (port1, port2, port3, port4, …, port14). 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.

2.2.2 MNET (Multi-Netting)

MNET 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.

2.2.3 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.

For example, say that you are running SSH on a real server on the inside 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'll have to add a port *forward* on the FortiBalancer appliance.

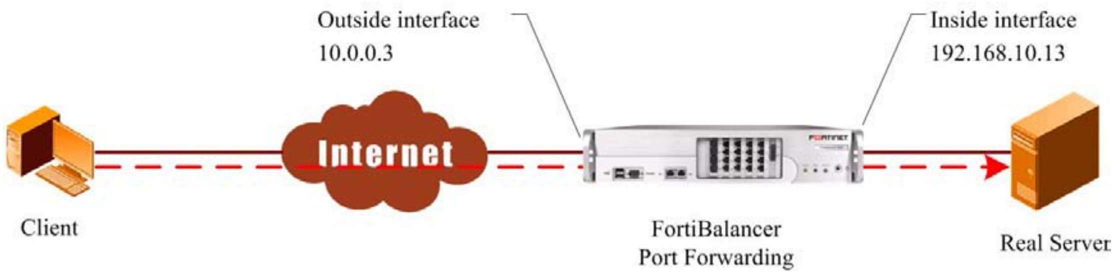


Figure 2-1 Port Forwarding

2.2.4 NAT (Network Address Translation)

Network Address Translation (NAT) is the translation of an Internet Protocol address (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.

2.2.4.1 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.

2.2.4.2 FortiBalancer Supports Two Types of NAT

Static NAT: Mapping an IP address on a 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 outside 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:

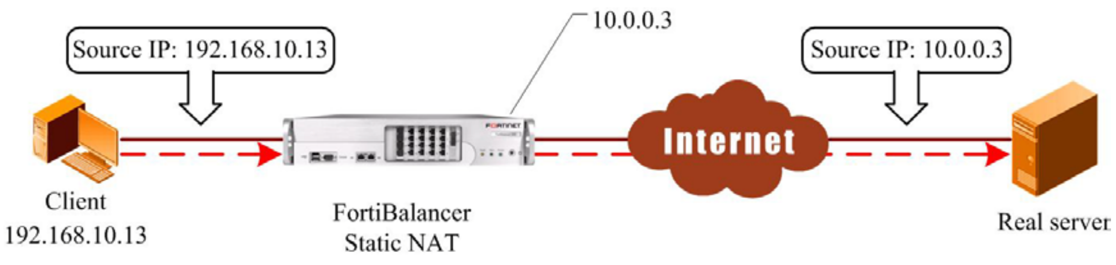


Figure 2-2 Static NAT

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

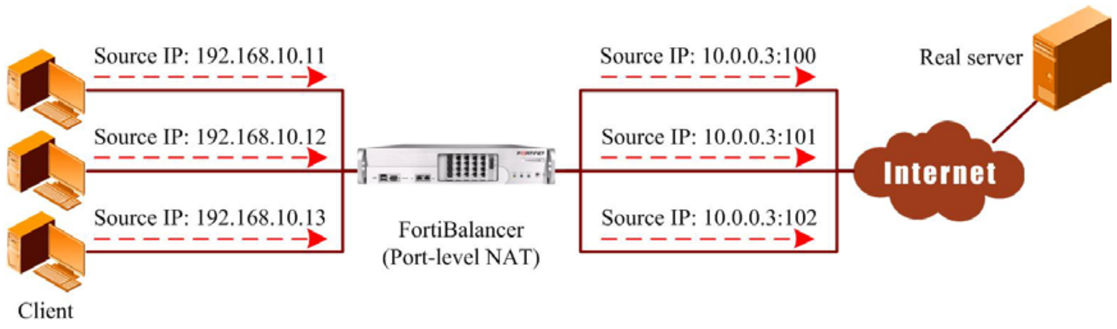


Figure 2-3 Port-level NAT

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.

2.2.5 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).

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.

2.3 Advanced Network Configuration

2.3.1 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.

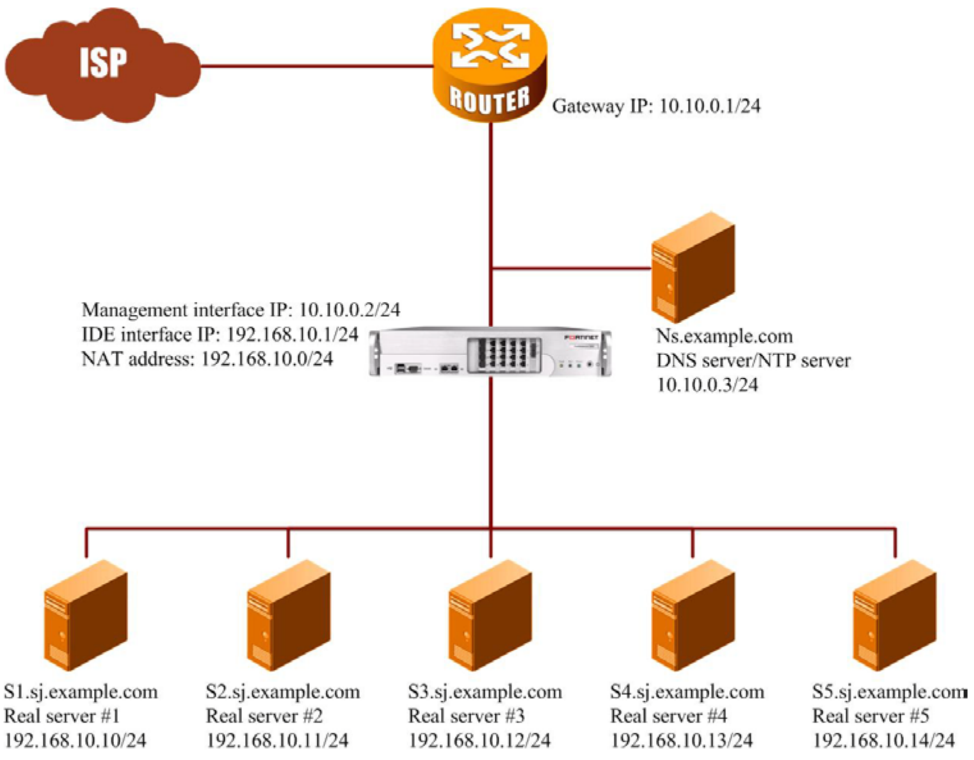


Figure 2-4 Network Architecture for Advanced Network Configuration

The most critical pieces of information from the figure above:

IP ADDRESS	DESCRIPTION
10.10.0.1/24	Gateway IP Address
10.10.0.2/24	Management IP Address
192.168.10.1/24	Inside 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 2-1 General Settings of VLAN, MNET, Port Forwarding, NAT and Dynamic Routing

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	**fwd tcp** *<local_ip> <local_port> <remote_ip> <remote_port> [timeout]*
Forwarding	**fwd udp** *<local_ip> <local_port> <remote_ip> <remote_port> [timeout]*
Configure NAT	**nat port** *<vip> <network_ip> <netmask> [timeout] [gateway]*

Operation	Command
	nat static *<vip> <network_ip> [timeout] [gateway]*

Configure Dynamic Routing	**rip {on|off}**
	rip network *<ip_address> <netmask>*
	ospf {on|off}
	ospf network *<ip_address> <netmask> <area_id>*

2.3.2 Sample Configuration for VLAN, MNET, Port Forwarding, NAT and Dynamic Routing via CLI

2.3.2.1 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

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

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

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

Since the FortiBalancer appliance currently only has one inside interface, this interface will be connected to a switch or router that must have Tag VLAN or Trunking turned on for that interface. See your switch vendors' documentation on how to setup Tag VLAN.

Note: If a system interface has been configured with VLAN, you cannot define an IP address or configure MNET on this interface.

2.3.2.2 MNET Configuration

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

→ Step 1 Define our mnet interfaces by using the "mnet" command

```
FBL(config)#mnet inside mnet1
FBL(config)#mnet inside mnet2
```

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

```
FBL(config)#ip address mnet1 192.168.1.1 255.255.255.0
FBL(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.

2.3.2.3 Port Forwarding Configuration

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

```
FBL(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 (SSL) and 80 (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.

2.3.2.4 NAT Configuration

For our sample configuration strategy, use the command as:

```
FBL(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 sample, 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 "nat port" command 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 "nat port" command 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:

```
FBL(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:

```
FBL(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:

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

2.3.2.5 Dynamic Routing Configuration

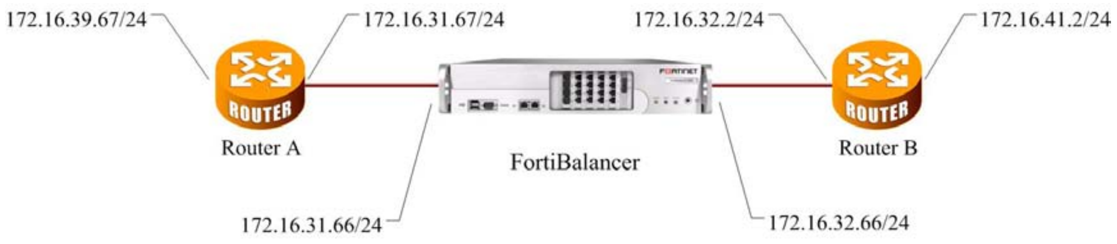


Figure 2-5 Dynamic Routing Configuration

→ Step 1 RIP Configurations

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

→ Step 2 OSPF Configurations

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

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

```
FBL(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.

Chapter 3 WebWall

3.1 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. 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 Fortinet WebWall works. For FBLOS, the WebWall feature can independently control each interface.

FortiBalancer WebWall permits TCP and UDP health check traffic, but cannot permit ICMP health check traffic automatically.

3.2 Understanding FortiBalancer WebWall

FortiBalancer WebWall is a full-fledged stateful Firewall. It bridges the gap between speed and security. Fortinet 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.

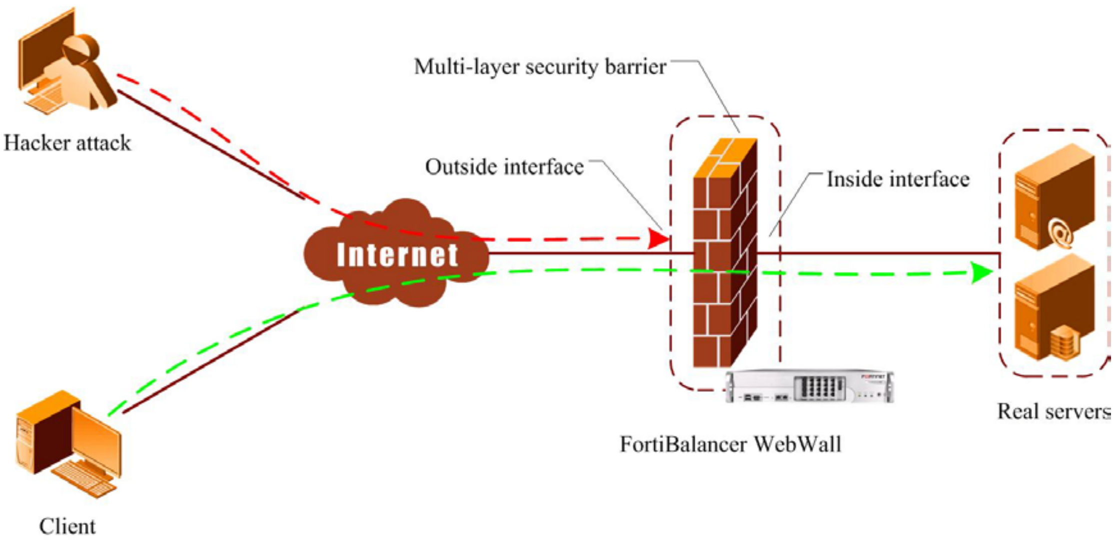


Figure 3-1 FortiBalancer WebWall

FortiBalancer WebWall contains several security mechanisms to protect Web servers from attack, including:

→ Access Control Lists Filtering

- → Protection against Syn-Flood, Fragmentation and DoS attacks
- → Stateful packet inspection
- → Single packet attack prevention

Access Control Lists 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 a FortiBalancer system's 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 don't 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 FortiBalancer's WebWall function will remain disabled until it is activated via the "**webwall on**" command.

3.3 WebWall Configuration

3.3.1 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.

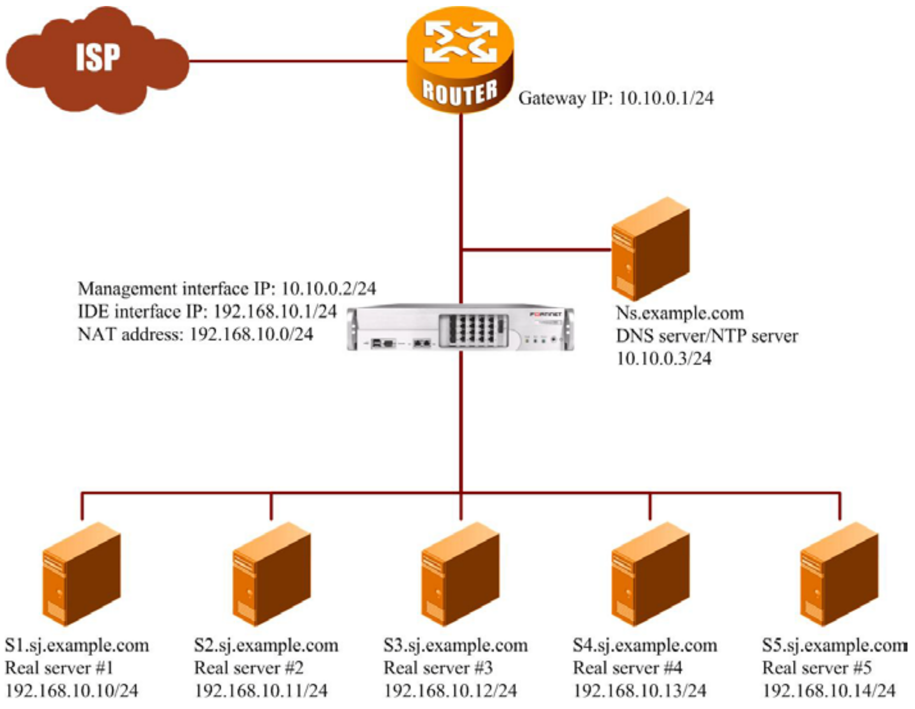


Figure 3-2 FortiBalancer WebWall Configuration

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 WebUI 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 inside interface IP.

Initially we'll define our access groups as follows:

- 50 All miscellaneous rules
- 100 All Management IP related rules
- 150 All VIP (Virtual IP) related rules

Table 3-1 General Settings of WebWall

Operation	Command
Configure access group	**accessgroup** *<accesslist_id> {interface}*
Configure ACL rules	**accesslist permit icmp echoreply** *<source_ip> <source_mask> <destination_ip> <destination_mask> <accesslist_id>* **accesslist permit icmp echorequest** *<source_ip> <source_mask> <destination_ip> <destination_mask> <accesslist_id>* **accesslist permit tcp** *<source_ip> <source_mask> <source_port> <destination_ip> <destination_mask> <destination_port> <accesslist_id>* **accesslist permit udp** *<source_ip> <source_mask> <source_port> <destination_ip> <destination_mask> <destination_port> <accesslist_id>* **accesslist deny icmp echoreply** *<source_ip> <source_mask> <destination_ip> <destination_mask> <accesslist_id>* **accesslist deny icmp echorequest** *<source_ip> <source_mask> <destination_ip> <destination_mask> <accesslist_id>* **accesslist deny tcp** *<source_ip> <source_mask> <source_port> <destination_ip> <destination_mask> <destination_port> <accesslist_id>* **accesslist deny udp** *<source_ip> <source_mask> <source_port> <destination_ip> <destination_mask> <destination_port> <accesslist_id>*
Enable/Disable WebWall	**webwall** *<interface>* **on** *[mode]* **webwall** *<interface >* **off**
View WebWall configurations	**show interface** **show accesslist** **show accessgroup**

3.3.2 Sample Configuration for WebWall via CLI

3.3.2.1 Configuring Access Groups

We may define any number of access groups and apply multiple groups to a designated interface. Let's go ahead and define our groups through the CLI based on these assumptions. Pertaining to our example model, the command should be executed as such:

```
FBL(config)#accessgroup 100 outside
FBL(config)#accessgroup 150 outside
FBL(config)#accessgroup 50 outside
```

You might have noticed that we also have specified what interfaces these access groups will be applied to.

3.3.2.2 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:

```
FBL(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.

```
FBL(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.

```
FBL(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 doesn't 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'll make another group.

```
FBL(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 inside interface when the WebWall function is **on**.

```
FBL(config)#accesslist permit icmp echorequest 192.168.10.0 255.255.255.0 192.168.10.1
255.255.255.255 50
FBL(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:

```
FBL(config)#accesslist permit tcp 10.10.10.30 255.255.255.255 0 10.10.10.10 255.255.255.255 22 100
FBL(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.

3.3.2.3 Configuring WebWall

At last once you are comfortable with the configuration of the other features of the FortiBalancer Appliance, and you should turn the WebWall feature back **on** by issuing the command:

```
FBL(config)#webwall inside on
FBL(config)#webwall outside on
```

Notes:
1. You should exercise 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 WebUI 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 FortiBalancer machines (FortiBalancer appliance and the sync peers).

3.3.2.4 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.

```
FBL(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
```

```
FBL(config)#show accessgroup
accessgroup 50 outside
accessgroup 100 outside
accessgroup 150 outside
```

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:

```
FBL(config)#show interface
port1(port1): flags=8843<UP,BROADCAST,RUNNING,SIMPLEX> mtu 1500
        inet 172.16.85.40 netmask 0xffffff00 broadcast 172.16.85.255
        ether 00:15:17:b9:64:4d
        media: autoselect (100baseTX <full-duplex>)
        status: active
        webwall status: OFF
        Hardware is i82571eb_quad_copper_lp
        Input queue: 534/4096 (size/max)
                total: 8726 packets, good: 8726 packets, 919262 bytes
                broadcasts: 4416, multicasts: 0
                3885 64 bytes, 4099 65-127 bytes,201 128-255 bytes
                113 255-511 bytes,399 512-1023 bytes,29 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: 0/4096 (size/max)
                total: 5696 packets, good:   5696 packets, 5696470 bytes
                broadcasts: 1, multicasts: 0
                821 64 bytes, 474 65-127 bytes,140 128-255 bytes
                452 255-511 bytes,358 512-1023 bytes,3451 1024-1522 bytes
                0 output errors
                0 Collisions, 0 Late collisions, 0 Deferred
                0 Single Collisions, 0 Multiple Collisions, 0 Excessive collisions
        0 lost carrier, 0 WDT reset
        packet drop (not permit): 0
                tcp  0          udp  0          icmp  0
        packet drop (deny): 0
                tcp  0          udp  0          icmp  0
        5 minute input rate 1400 bits/sec, 1 packets/sec
        5 minute output rate 6336 bits/sec, 1 packets/sec
```

```
port2(port2): flags=8842<BROADCAST,RUNNING,SIMPLEX> mtu 1500
        ether 00:15:17:b9:64:4c
        media: autoselect
        status: no carrier
        webwall status: OFF
        Hardware is i82571eb_quad_copper_lp
        Input queue: 0/4096 (size/max)
                total: 0 packets, good: 0 packets, 0 bytes
                broadcasts: 0, multicasts: 0
                0 64 bytes, 0 65-127 bytes,0 128-255 bytes
                0 255-511 bytes,0 512-1023 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: 0/4096 (size/max)
                total: 0 packets, good:   0 packets, 0 bytes
                broadcasts: 0, multicasts: 0
                0 64 bytes, 0 65-127 bytes,0 128-255 bytes
                0 255-511 bytes,0 512-1023 bytes,0 1024-1522 bytes
                0 output errors
                0 Collisions, 0 Late collisions, 0 Deferred
                0 Single Collisions, 0 Multiple Collisions, 0 Excessive collisions
        0 lost carrier, 0 WDT reset
        packet drop (not permit): 0
                tcp  0          udp  0          icmp  0
        packet drop (deny): 0
                tcp  0          udp  0          icmp  0
        5 minute input rate 0 bits/sec, 0 packets/sec
                5 minute output rate 0 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.

Chapter 4 Link Aggregation

4.1 Overview

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

4.2 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.

4.3 Link Aggregation Configuration

4.3.1 Configuration Guidelines

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

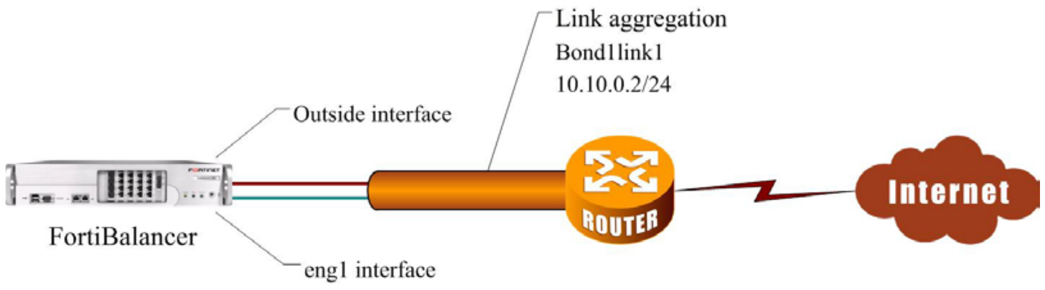


Figure 4-1 Link Aggregation Configuration

Table 4-1 General Settings of Link Aggregation

Operation	Command
Bond system interfaces	**bond interface** *<bond name> <interface name> [1|0]*
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>*
Assign default route	**ip route default** *<gateway_ip>*

4.3.2 Sample Configuration for Link Aggregation via CLI

→ **Step 1 Bind system interfaces with bond interface**

In our example, we bind the “outside” interface and the “eng1” interface with the bond interface bond1, and further set the “outside” interface as the primary, and “eng1” as the backup in the bond interface.

```
FBL(config)#bond interface bond1 outside 1
FBL(config)#bond interface bond1 eng1 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.

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

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

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

→ **Step 4 Set the gateway IP address**

```
FBL(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.

```
FBL(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 5 Clustering

5.1 Overview

The FortiBalancer Clustering Technology allows you to maintain high availability within a local site. With other options you can also distribute load across multiple boxes within a cluster. We will cover Virtual Clustering of the outside and inside interfaces in this chapter.

5.2 Understanding FortiBalancer Clustering

FortiBalancer Clustering Technology allows two or more FortiBalancer devices to be grouped together to form a logical device, which provides scalability and high availability within a local site.

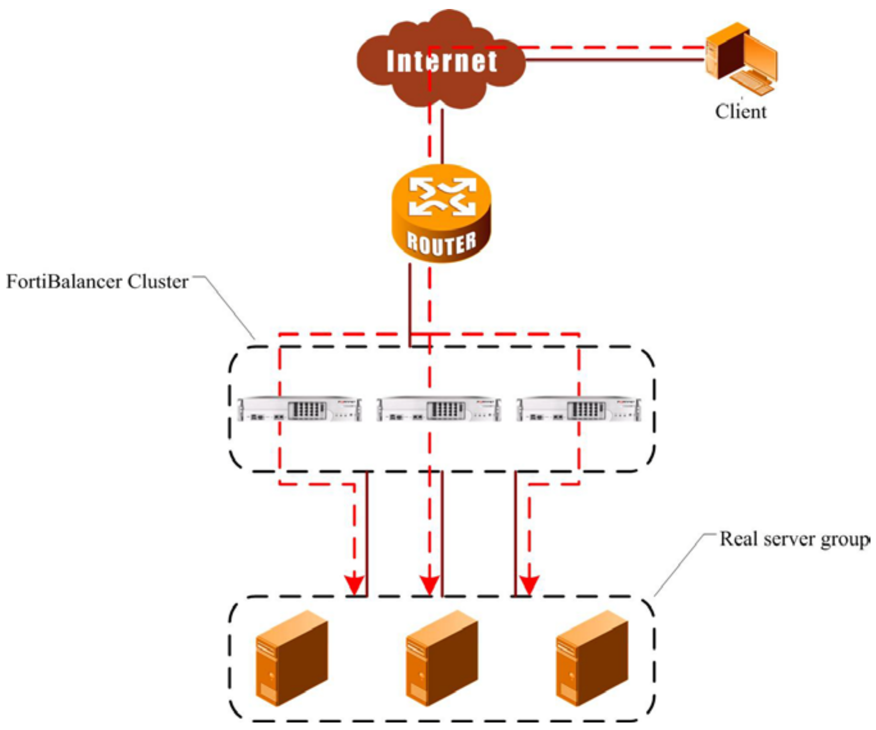


Figure 5-1 FortiBalancer Clustering

Clustering can be configured in Active-Standby or Active-Active 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.

5.2.1 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 FortiBalancer cluster. When one system powers off, panics, reboots or its interface losses carrier (link disconnection), all the traffic will be immediately switched to the other. FortiBalancer clustering technology with fast failover mechanism provides higher availability and much faster response time than the typical FortiBalancer clustering technology.

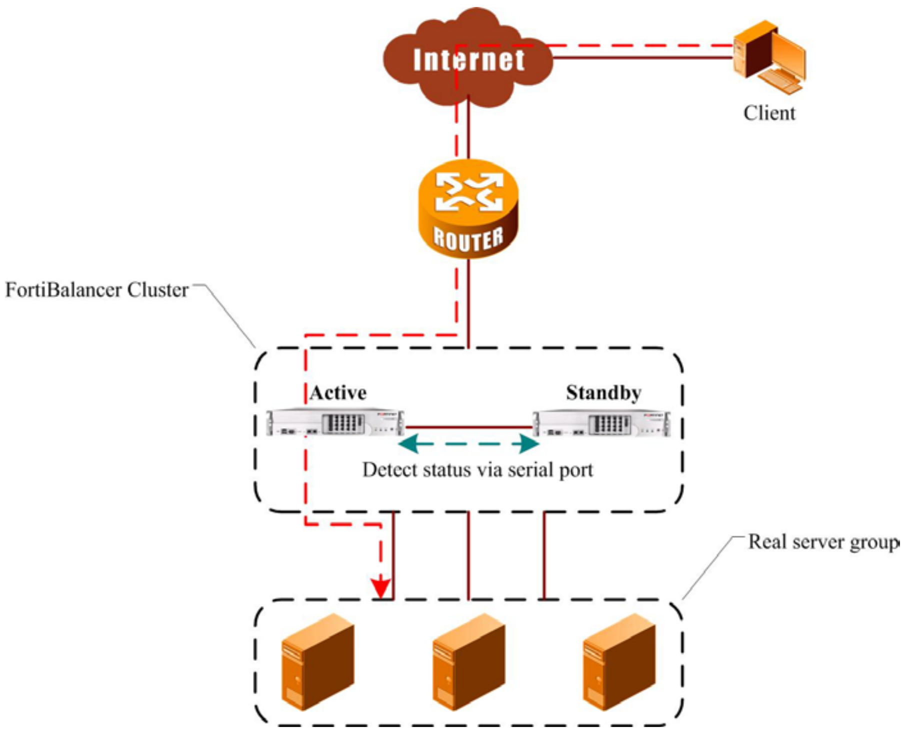


Figure 5-2 FortiBalancer Clustering FFO Mode

5.2.2 Discreet Backup Mode

For traditional clustering, a backup and a master communicate each other's state information through the network. If the backup doesn't receive the VRRP 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 figure shows how the Discreet Backup mode works.

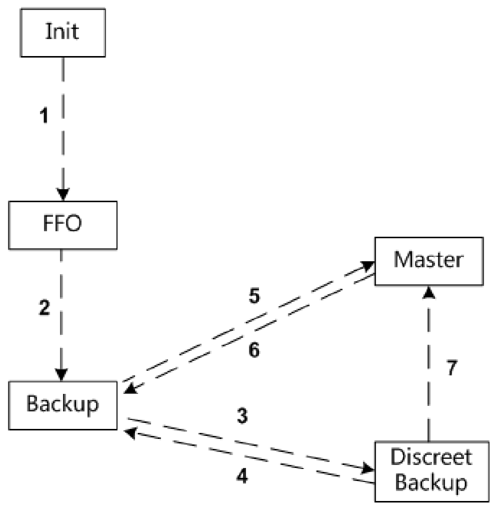


Figure 5-3 Discreet Backup Mode Working Mechanism

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.

   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.

3. If the backup receives a higher priority VRRP packet, it will switch to Discreet Backup state.

4. In the following events, the discreet backup will switch to Backup state:

   - 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.).

   - 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.

5. In the following events, the backup will switch to Master state:

   - The backup receives a lower priority VRRP packet (in Preemption mode).

   - In three continuous broadcast intervals (the default interval is 5 seconds, three intervals are 15 seconds), the backup doesn't receive the VRRP packet from the master.

6. If the master receives a higher priority VRRP packet, it will switch to Backup state.

7. 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.

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

5.3 Clustering Configuration

Virtual Clustering provides high availability to SLB VIPs for the outside interface and for redundant gateways via the inside interface. We'll cover SLB clustering first and then discuss how to handle the inside interface.

5.3.1 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.

5.3.1.1 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 following figure for a typical layout of Active-Standby architecture.

FBL1 is the current master, and handles SLB traffic for VIP.
FBL2 is the backup, and listens for advertisements from the master. It will resume master status if FBL1 stops sending advertisements (i.e. FBL1 fails).

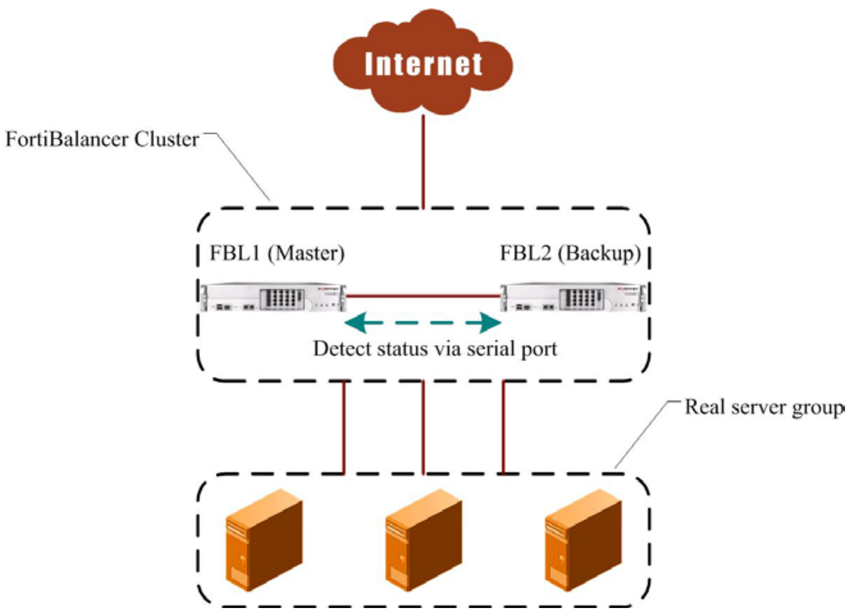


Figure 5-4 Active-Standby Two-Node Architecture

Table 5-1 General Settings for Active-Standby Two-Node Clustering

Operation	Command
Configure SLB	Refer to Server Load Balancing Configuration in chapter Server Load Balancing.
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> {0|1}*
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|all]*
Enable fast failover feature	**cluster virtual ffo {on|off}** **cluster virtual ffo interface carrier loss timout** *<interface_timeout>*

Sample Configuration for Active-Standby SLB Clustering via CLI

Now let's start to configure FBL1 and FBL2:

→ Step 1 Configure SLB for both FBL1 and FBL2

```
FBL1(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
FBL1(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
FBL1(config)#slb group method "group1" rr
FBL1(config)#slb group member "group1" "server1" 1
FBL1(config)#slb group member "group1" "server2" 1
```

```
FBL1(config)#slb virtual http “vip1” 192.168.2.100 80
FBL1(config)#slb policy default “vip1” “group1”

FBL2(config)#slb real http “server1” 192.168.1.50 80 1000 tcp 1 1
FBL2(config)#slb real http “server2” 192.168.1.51 80 1000 tcp 1 1
FBL2(config)#slb group method “group1” rr
FBL2(config)#slb group member “group1” “server1” 1
FBL2(config)#slb group member “group1” “server2” 1
FBL2(config)#slb virtual http “vip1” 192.168.2.100 80
FBL2(config)#slb policy default “vip1” “group1”
```

→ Step 2 Configure a virtual interface name

```
FBL1(config)#cluster virtual ifname “outside” 100

FBL2(config)#cluster virtual ifname “outside” 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.

```
FBL1(config)#cluster virtual auth outside 100 0

FBL2(config)#cluster virtual auth outside 100 0
```

→ Step 4 Configure virtual cluster preemption

Now we configure FBL1 to preempt the VIP when the initial master returns online. For FBL2, it will not preempt the VIP from the master node, but will take over if the master ceases operations

```
FBL1(config)#cluster virtual preempt outside 1 1

FBL2(config)#cluster virtual preempt outside 1 0
```

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

```
FBL1(config)#cluster virtual vip “outside” 100 192.168.2.100

FBL2(config)#cluster virtual vip “outside” 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 FBL1 to always be master of the VIP, we will set its priority to 255. For FBL2, 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:

```
FBL1(config)#cluster virtual priority outside 100 255
```

```
FBL2(config)#cluster virtual priority outside 100 100
```

Note: The state is the backup on FBL2. This is expected since it is of lower priority than the master.

→ **Step 7 Turn on the clustering**

```
FBL1(config)#cluster virtual on

FBL2(config)#cluster virtual on
```

→ **Step 8 Turn on fast failover**

```
FBL1(config)#cluster virtual ffo on
FBL1(config)#cluster virtual ffo interface carrier loss timeout 1000

FBL2(config)#cluster virtual ffo on
FBL2(config)#cluster virtual ffo interface carrier loss timeout 1000
```

5.3.1.2 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.

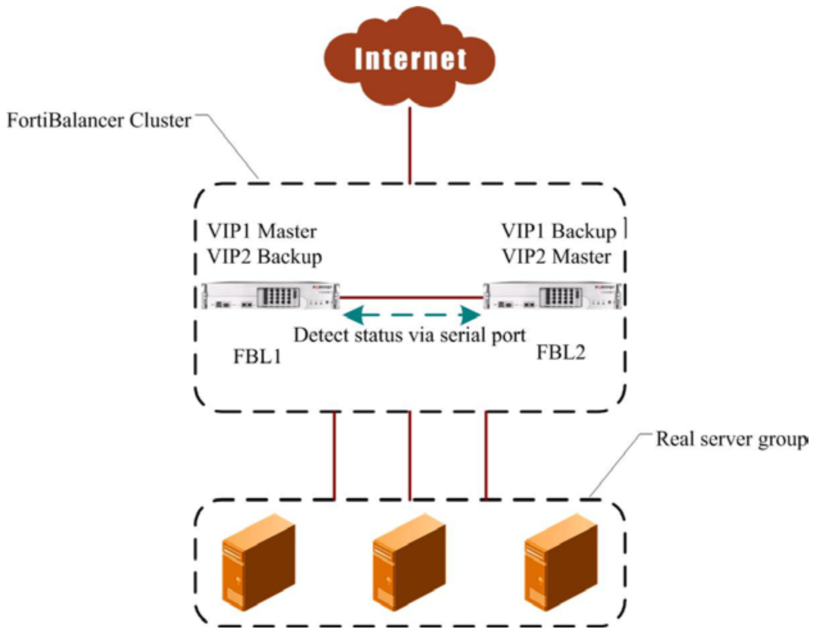


Figure 5-5 Active-Active Two-Node Architecture

Referring to the above figure, FBL1 is the master for VIP1 and the backup for VIP2 and FBL2 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 5-2 General Settings for Active-Active Two-Node Clustering

Operation	Command
Configure SLB	Refer to Server Load Balancing Configuration in chapter Server Load Balancing.
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> {0|1}*
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|all]*

Sample Configuration 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 FBL1 and FBL2

```
FBL1(config)#slb real http “server1” 192.168.1.50 80 1000 tcp 1 1
FBL1(config)#slb real http “server2” 192.168.1.51 80 1000 tcp 1 1
FBL1(config)#slb group method “group1” rr
FBL1(config)#slb group member “group1” “server1” 1
FBL1(config)#slb group member “group1” “server2” 1
FBL1(config)#slb virtual http “vip1” 192.168.2.100 80
FBL1(config)#slb virtual http “vip2” 192.168.2.101 80
FBL1(config)#slb policy default “vip1” “group1”

FBL2(config)#slb real http “server1” 192.168.1.50 80 1000 tcp 1 1
FBL2(config)#slb real http “server2” 192.168.1.51 80 1000 tcp 1 1
FBL2(config)#slb group method “group1” rr
FBL2(config)#slb group member “group1” “server1” 1
FBL2(config)#slb group member “group1” “server2” 1
FBL2(config)#slb virtual http “vip1” 192.168.2.100 80
FBL1(config)#slb virtual http “vip2” 192.168.2.101 80
FBL2(config)#slb policy default “vip1” “group1”
```

→ Step 2 Configure a virtual interface name

```
FBL1(config)#cluster virtual ifname “outside” 100
```

```
FBL1(config)#cluster virtual ifname “outside” 101

FBL2(config)#cluster virtual ifname “outside” 100
FBL2(config)#cluster virtual ifname “outside” 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.

```
FBL1(config)#cluster virtual auth outside 100 0
FBL1(config)#cluster virtual auth outside 101 0

FBL2(config)#cluster virtual auth outside 100 0
FBL2(config)#cluster virtual auth outside 101 0
```

→ Step 4 Configure virtual cluster preemption

```
FBL1(config)#cluster virtual preempt outside 100 1
FBL1(config)#cluster virtual preempt outside 101 0

FBL2(config)#cluster virtual preempt outside 100 0
FBL2(config)#cluster virtual preempt outside 101 1
```

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

```
FBL1(config)#cluster virtual vip “outside” 100 192.168.2.100
FBL1(config)#cluster virtual vip “outside” 101 192.168.2.101

FBL2(config)#cluster virtual vip “outside” 100 192.168.2.100
FBL2(config)#cluster virtual vip “outside” 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.

```
FBL1(config)#cluster virtual priority outside 100 255
FBL1(config)#cluster virtual priority outside 101 100

FBL2(config)#cluster virtual priority outside 100 100
FBL2(config)#cluster virtual priority outside 101 255
```

→ Step 7 Turn on the clustering

```
FBL1(config)#cluster virtual on

FBL2(config)#cluster virtual on
```

5.3.2 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.

5.3.2.1 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.

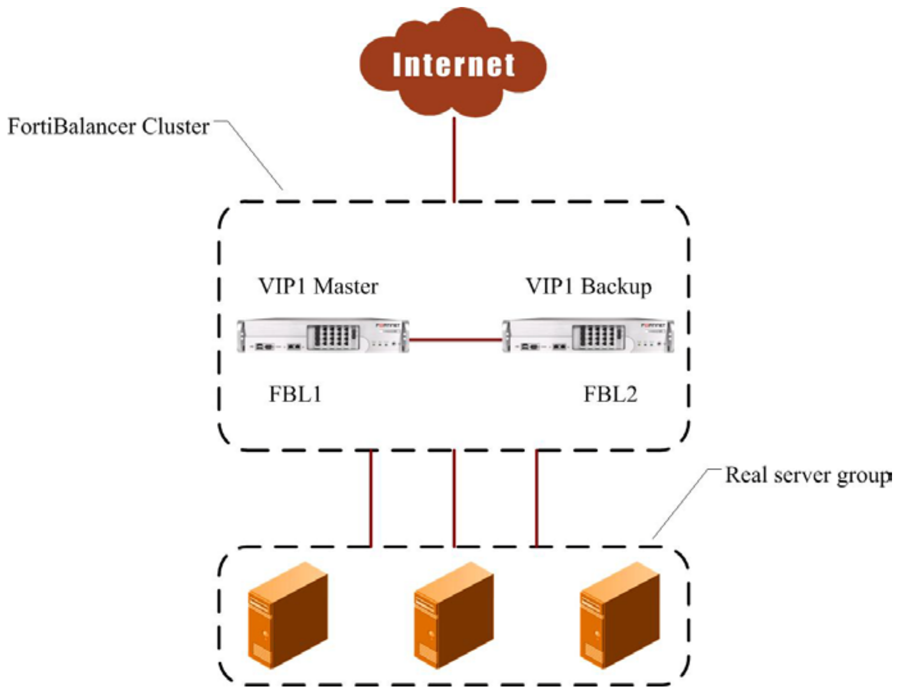


Figure 5-6 Inside Interface Active-Standby Mode

Table 5-3 General Settings for 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]*

Operation	Command
Enable the virtual cluster	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*

Sample Configuration for Active-Standby Clustering Inside Interface via CLI

→ Step 1 Configure a virtual interface and its cluster ID

```
FBL1(config)#cluster virtual ifname “inside” 100

FBL2(config)#cluster virtual ifname “inside” 100
```

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

```
FBL1(config)#cluster virtual vip “inside” 100 192.168.1.3

FBL2(config)#cluster virtual vip “inside” 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.

```
FBL1(config)#cluster virtual priority inside 100 255

FBL2(config)#cluster virtual priority inside 100 100
```

→ Step 4 Turn on the clustering

```
FBL1(config)#cluster virtual on

FBL2(config)#cluster virtual on
```

5.3.2.2 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.

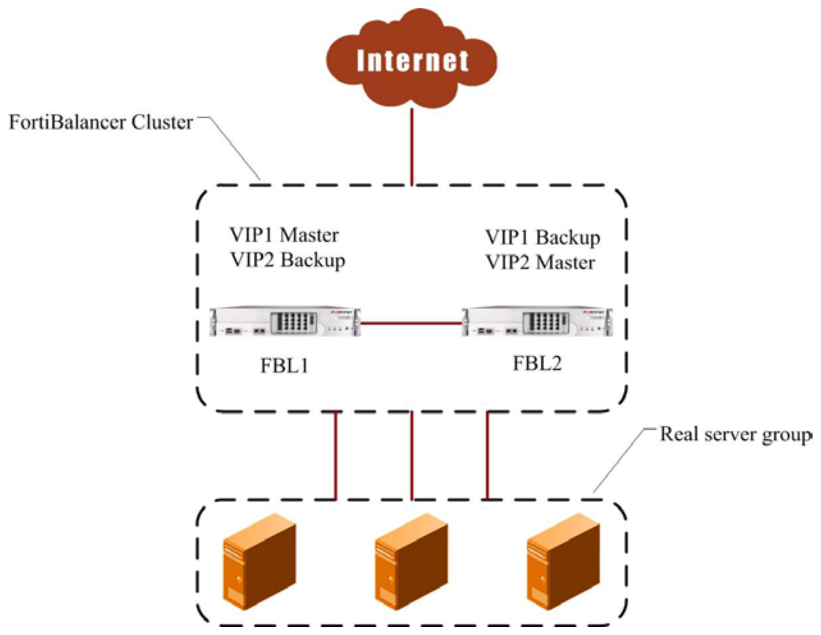


Figure 5-7 Inside Interface Active- Active Mode

Table 5-4 General Settings for 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]*

Sample Configuration 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

```
FBL1(config)#cluster virtual ifname “inside” 100
FBL1(config)#cluster virtual ifname “inside” 101

FBL2(config)#cluster virtual ifname “inside” 100
FBL2(config)#cluster virtual ifname “inside” 101
```

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

```
FBL1(config)#cluster virtual vip “inside” 100 192.168.1.3
FBL1(config)#cluster virtual vip “inside” 101 192.168.1.4

FBL2(config)#cluster virtual vip “inside” 100 192.168.1.3
FBL2(config)#cluster virtual vip “inside” 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.

```
FBL1(config)#cluster virtual priority inside 100 255
FBL1(config)#cluster virtual priority inside 101 100

FBL2(config)#cluster virtual priority inside 100 100
FBL2(config)#cluster virtual priority inside 101 255
```

→ Step 4 Turn on the clustering

```
FBL1(config)#cluster virtual on

FBL2(config)#cluster virtual on
```

Chapter 6 Server Load Balancing

6.1 Overview

Server Load Balancing (SLB) 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 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. We actually will setup the appliance working from the bottom of this figure to the top. The bottom of the figure shows the physical servers, also known as real servers. The lines that connect real servers to server groups denote memberships in that group. A real server can belong to more than one group. The group circles denote definitions of groups and methods of load balancing. The lines from the VIP ovals to groups show policies or rules that bind requests coming to virtual IPs to get directed to a load balancing group.

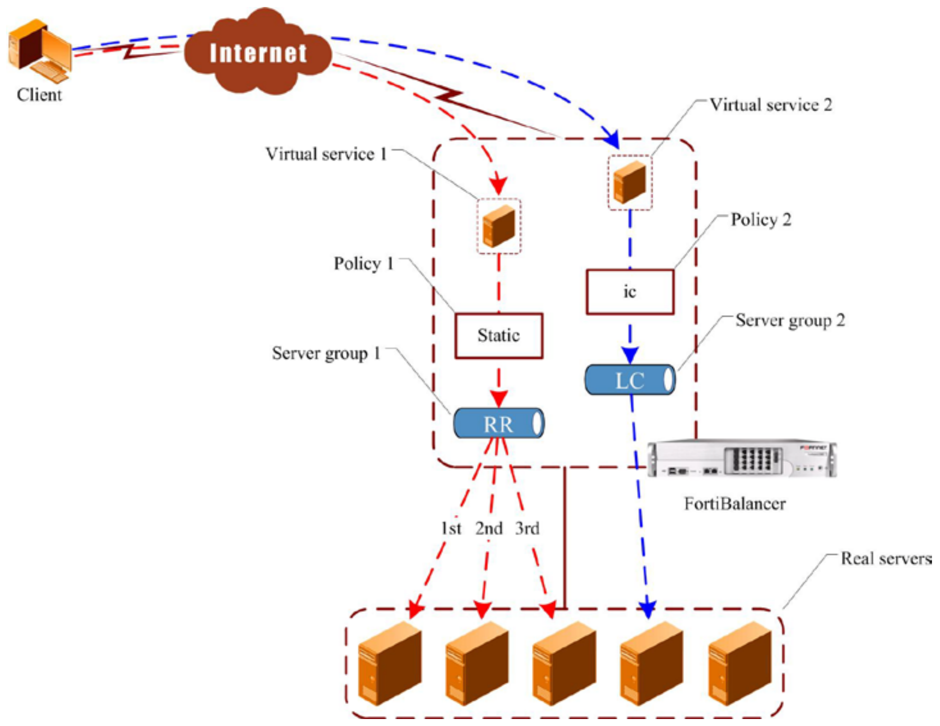


Figure 6-1 SLB Architecture

6.2 Understanding Server Load Balancing

6.2.1 Methods of Load Balancing

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

SLB Methods	Description
	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 cookies in the HTTP header.
Insert Cookie (ic)	Dynamically inserts cookies to allow FBLOS™ 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 GSLB 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, Persistent Hostname, Hash Cookie, Hash Header, SSL SID, Hash IP** and **Consistent Hash IP**. For more information on these additional SLB methods, please consult the CLI Handbook.

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 (L2~L7). 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:

Table 6-2 SLB Policies

Basic Policy	Persistent Policy	Qos Policy
Redirect	Persistent URL	QoS Cookie
Static	Persistent Cookie	QoS Hostname
Default	Rewrite Cookie	QoS URL
Backup	Insert Cookie	QoS Network
	Hash URL	QoS Clientport
		Regular Expression

Basic Policy	Persistent Policy	Qos Policy
		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 new policy flow that FortiBalancer appliance SLB will follow:

a. redirect
b. static
c. (Customized SLB Policy Precedence)
   qos-clientport - qos client port
   qos-network - qos network
   pu - persistent url
   rc - rewrite cookie
   ic - insert cookie
   pc - persistent cookie
   qos-cookie - qos cookie
   qos-hostname - qos hostname
   qos-url - qos url
   regex - regex url
   header - header
   hu - hash url
d. default
e. backup

For L4 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

FortiBalancer 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 (virtual link)" 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.

Therefore, will be distributed to a group associated to a vlink only when they match two or more nested policies.

Limitations of SLB Policy Nesting:

1. Some policies can't be nested, such as static, redirect, filetype and external policies.
2. 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.
3. 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.
4. Only HTTP and HTTPS protocols are supported.

6.2.3 SLB Health Check

FortiBalancer appliance provides three types of health check:

→ Basic health check
→ Additional health check
→ Script health check

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, UDP, 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. See "**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 FortiBalancer.

6.2.3.1 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 requests.

ARP Health Check

ARP health check is to support Layer 2 SLB real service health check.

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 t 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).

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 a 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 checks are provided for checking the availability of the RADIUS 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”.

6.2.3.2 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> <ip> <port> <list name> [frequency] [hc_localip] [hc_localport]*

6.2.3.3 HTTP Requests and Responses

By default FBLOS 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:

```
FBL(config)#health on
FBL(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
FBL(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 *<server_name> <request_index> <response_index> [ip] [port]*

For our network example:

→ Step 1 Define your own HTTP request

```
FBL(config)#health request 1 “GET /cgi-bin/dbstatus.pl”
```

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

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

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

The health check provided by the FBLOS 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 **admin**

→ Step 1 Add a real service with HTTP health check

```
FBL(config)#slb real http rs 10.3.16.188 80 1000 http 3 3
```

→ Step 2 Configure HTTP health check

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

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

```
FBL(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).

6.2.3.4 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 our 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.

A new CLI command is introduced to turn on/turn off the function:

slb virtual health {on|off}

Note: If WebWall is turned on for the interface ("inside" for most cases) that SLB real services are using, you will need to add an access list rule to allow traffic between the FortiBalancer 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.

6.2.3.5 Health Failover

If all real servers configured in a 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 a FortiBalancer appliance is marked as **UP** by Health Check, the FortiBalancer appliance will take over the traffic again if its mode is preemptive.

6.2.4 Triangle Transmission

FortiBalancer's 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 transmission, 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.

Packet Flow

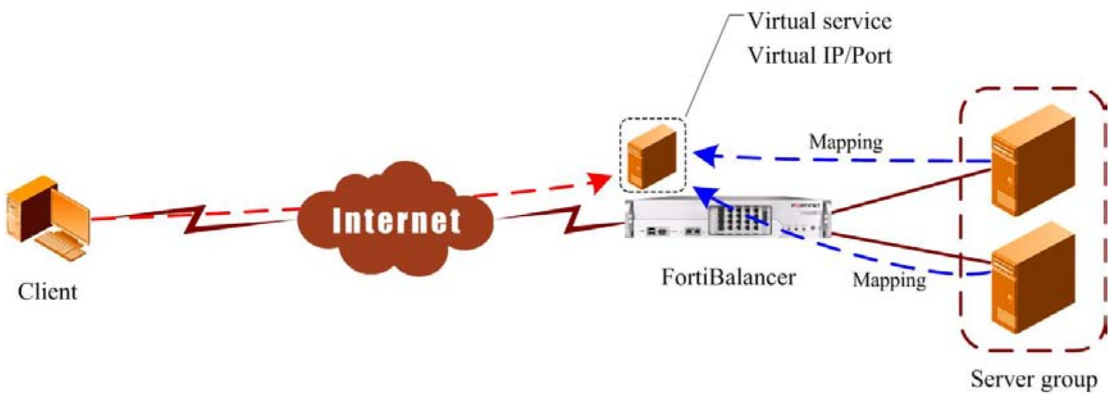


Figure 6-2 Triangle Transmission

1. Client sends a request to a virtual IP on FortiBalancer appliance via the router.
2. 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 inside 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 FortiBalancer appliance, but the response will be sent from the real server to the client directly without hitting 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.

6.2.5 Session Initiation Protocol (SIP) Load Balancing

What is FortiBalancer SIP 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, FortiBalancer 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.

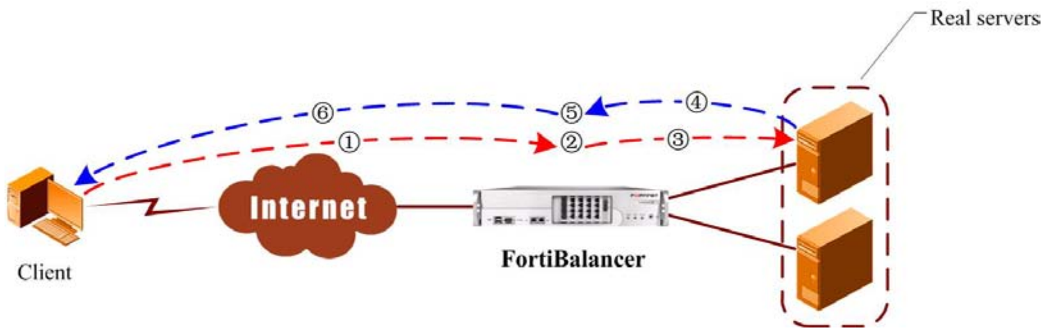


Figure 6-3 SIP Load Balancing

FortiBalancer 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 don't 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, FortiBalancer SIP SLB supports the broadcasting of SIP registration requests to all the SIP register servers in the same SLB group.

6.2.6 Real Time Streaming Protocol (RTSP) Load Balancing

What is RTSP

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 FortiBalancer 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 SLB 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.

6.2.7 L2 IP/MAC-based Load Balancing

L2 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 (L4 and L7 SLB), the traffic to be balanced in layer 2 doesn't need to access a particular IP address or "IP address + Port" pair defined in the interface (normally the outside interface) on FortiBalancer appliance for balancing purpose. As long as the incoming traffic can be routed to FortiBalancer appliance's interface with defined L2 virtual services (e.g. virtual services are defined on FortiBalancer appliance's outside interface and FortiBalancer appliance outside interface's system IP address is set as the gateway of client machines), it will be balanced among a pool of L2 real services according to configured load balance algorithms.

L2 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;

→ L2 SLB only supports default policy;

→ L2 SLB real services only supports two types of health checks: ARP request check and ICMP check (ping);

→ The supported groups methods are rr (Round Robin) and hi (Hash IP).

6.2.8 L3 IP-based Load Balancing

L3 IP based SLB balances all the TCP and UDP traffic from one global virtual IP address to multiple real servers. Unlike L4 SLB, both the virtual service and real service in L3 SLB are defined by single IP address without port number. L3 SLB real services only support ICMP health check. Without specifying port number, L3 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.

6.2.9 Port Range Load Balancing

Port range SLB allows us to define a virtual service with a range of ports so that FBLOS 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 L4 and L7 virtual services. That means, if an input request hits both a L4/L7 virtual service and a port range virtual service, the L4/L7 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 doesn't 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:

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

6.2.10 DirectFWD

DirectFWD is a new L4 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 L4 SLB, DirectFWD provides remarkably better L4 SLB performance. This function is controlled by the command "**slb direcfwd {on|off}**".

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

1. It is only used for TCP and UDP SLB, and temporarily only supports static, default and backup policies. All the L4 SLB methods, except Shortest Response Time, are supported. (Please refer to the policy-method supporting matrix to check the details about the L4 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.

6.2.10.1 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.

6.3 Server Load Balancing Configuration

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

This section covers more than one SLB configuration sample. At first, we will give a sample with basic SLB configuration. Then more configuration samples 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 network:

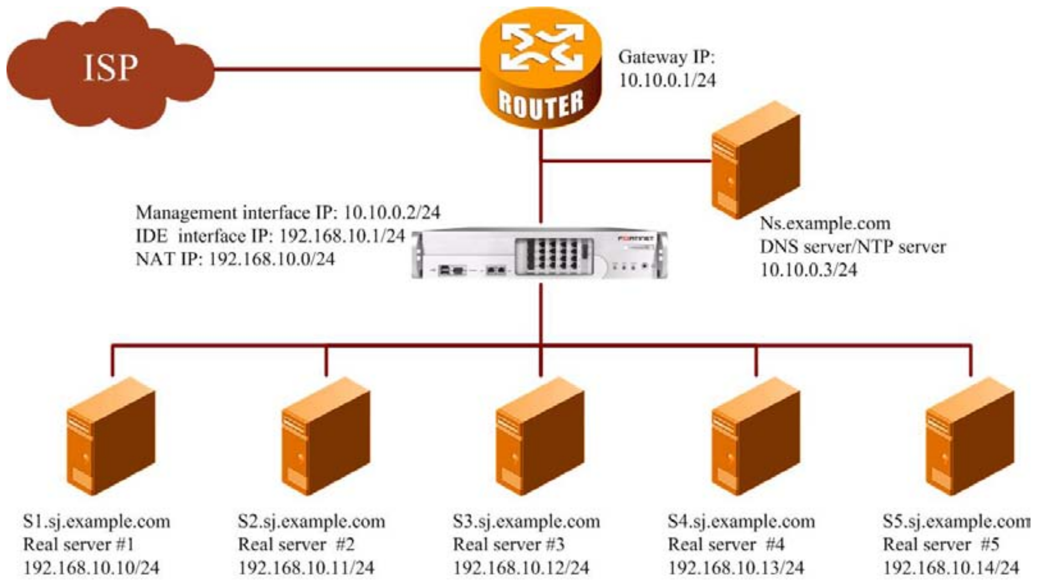


Figure 6-4 SLB Netwok Architetcure

Table 6-3 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} [hc_up] [hc_down]* **slb real ftp** *<real_name> <ip> [port] [max_conn] [tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp] [hc_up] [hc_down]*

Operation	Command
	slb real http *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp] [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]* **slb real https** *<real_name> <ip> [port] [max_conn] [https|tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp] [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] [hc_up] [hc_down]* **slb real dns** *<real_name> <ip> <port> [max_conn] [dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp] [hc_up] [hc_down] [timeout]* **slb real health** *<real_name> <ip> <port> [http|https|tcp|icmp|dns|script-tcp|script-udp|script-tcps|radius-auth|radius-acct|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** *[granularity] [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]* **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** *<virtual_name> <vip> [vport] [max_conn]* **slb virtual ip** *<virtual_name> <vip>* **slb virtual l2ip** *<virtual_name> <virtual_ip> [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>*

Operation	Command
	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 Sample Configuration for Basic SLB 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:

```
FBL(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.

```
FBL(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.

```
FBL(config)#slb real http service3http 192.168.10.12
FBL(config)#slb real http service4http 192.168.10.13
FBL(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:

```
FBL(config)#slb real health service1http 192.168.10.10 80 http
FBL(config)#slb real health service1http 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:

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

Now verify our configuration change:

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

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

```
FBL(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.

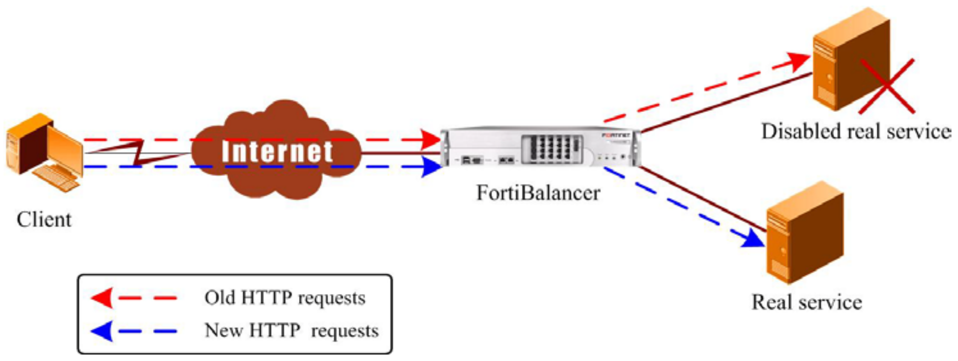


Figure 6-5 Graceful Shutdown of Real Servive

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

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

```
FBL(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:

For our model:

```
FBL(config)#slb group member rrgroup server1http
FBL(config)#slb group member rrgroup server2http
FBL(config)#slb group member rrgroup server3http
FBL(config)#slb group member rrgroup server4http
FBL(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 doesn't 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:

```
FBL(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”.

```
FBL(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 Sample

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 don’t have to redefine the real servers and virtual service so we will leave them as they were originally defined in the basic SLB configuration sample above.

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

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

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

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

```
FBL(config)#slb policy qos url url_pol_1 virtual1http group1 “.jpg ”1
FBL(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 doesn’t 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.

```
FBL(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.

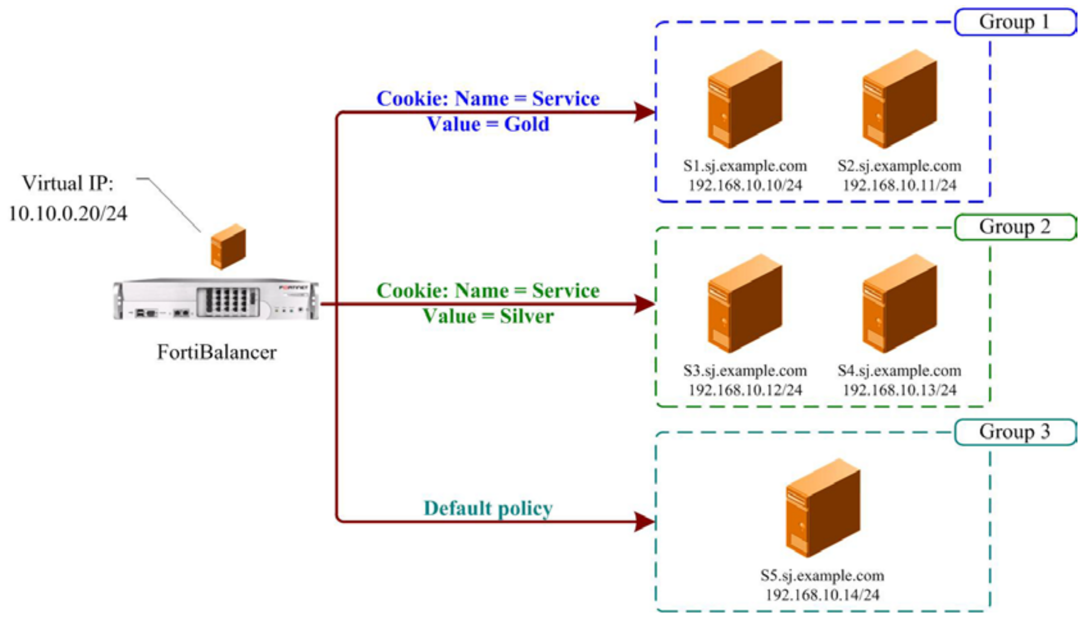


Figure 6-6 QoS Cookie Policy

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 don’t 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 don’t have to redefine the real services so we will leave them as they have been originally defined in the basic SLB configuration sample.

→ Step1 Define Group 1, 2, 3

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

→ Step2 Add the real services into the groups

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

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

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

→ Step 3 Set the default policy (Group 3) where cookies will be set.

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

→ Step 4 Set the QoS Cookie policy for the group 1 and group 2

For our configuration model:

```
FBL(config)#slb policy qos cookie gold_policy virtual1http gold_group “service=gold” 1
FBL(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 group 1 and group 2

```
FBL(config)#slb group method group1 pc
FBL(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

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

→ Step 3 Set the default policy (Group 2) where cookies will be set.

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

→ Step 4 Set the persistent cookie policy for Group 1

For our configuration model:

```
FBL(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.

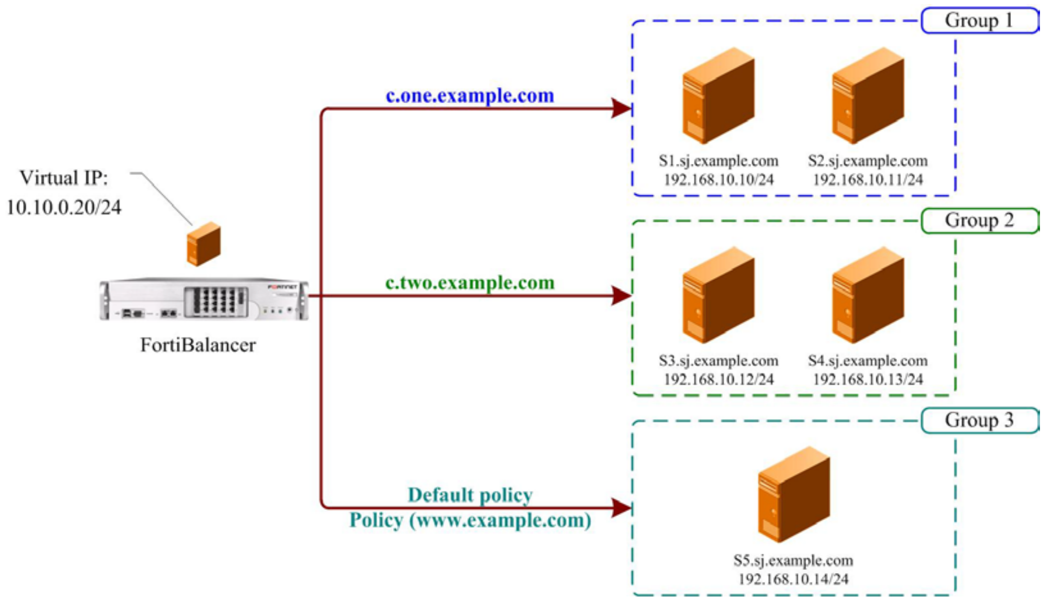


Figure 6-7 QoS Hostname Policy

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 Setup the groups

```
FBL(config)#slb group method c_one_group rr
FBL(config)#slb group method c_two_group rr
FBL(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

```
FBL(config)#slb group member c_one_group service1http
FBL(config)#slb group member c_one_group service2http
FBL(config)#slb group member c_two_group service3http
FBL(config)#slb group member c_two_group service4http
FBL(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 don't have c-one.example.com or c-two.example.com as their host names.

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

→ Step 4 Define the QoS Hostname policy for c_one_group and c_two_group

```
FBL(config)#slb policy QoS hostname c_one_pol virtual1http c_one_group
c_one.example.com 1
FBL(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 layer7 policy and direct the request to server1. The following configuration example shows you how to setup a persistent URL policy.

→ Step 1 Create the group for Persistent URL policy

```
FBL(config)#slb group method pu_group pu
FBL(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

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

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

→ Step 3 Setup the policies

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

Insert Cookie

Insert cookie is a method that allows us to insert a cookie into the response from the server in order to maintain persistency.

→ **Step 1 Create the group for Insert Cookie policy**

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

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

→ **Step 2 Create the group for Insert Cookie policy**

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

→ **Step 3 Setup the policies**

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

By default you do not need to give a cookie name. The FBLOS will generate a random cookie name and store it in the configuration file for you. You can see what the cookie name is by running the "**show slb group**" command:

```
FBL(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 to the **slb group method** command:

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

Now responses from these services will get the cookie: CookieExampleName inserted into the stream. The value is used by the FBLOS to determine which service to persist to.

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 FBLOS has made. So the backend server will see the modified cookie.

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

→ **Step 1 Create the group for Rewrite Cookie policy**

```
FBL(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**

```
FBL(config)#slb group member rc_group service1http
FBL(config)#slb group member rc_group service2http
FBL(config)#slb group member rc_group service3http
FBL(config)#slb group member rc_group service4http
FBL(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:

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

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

→ Step 2 Create a group

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

→ Step 3 Add the real services into the group

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

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

Currently, redirect policy supports rr (round robin), lc (least TCP 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. It has two parts: configuration guidelines and configuration sample via CLI.

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.

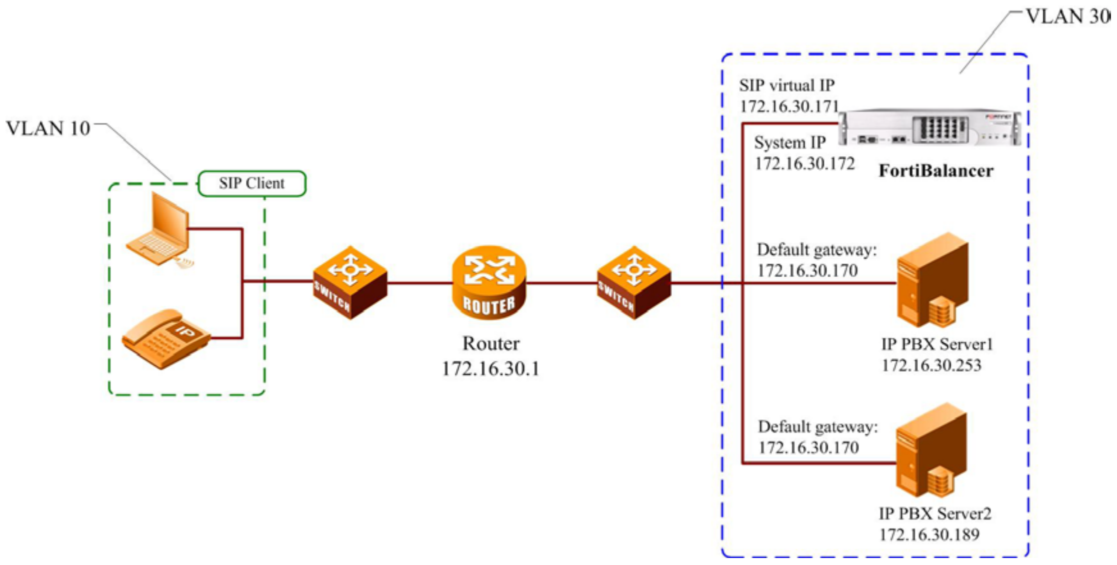


Figure 6-8 SIP Load Balancing

Table 6-4 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} [hc_up] [hc_down] [timeout]* **slb real sipudp***<real_name> <ip> <port> [max_conn] {icmp/script-tcp/script-udp/radius-auth/radius-acct/sip-tcp/sip-udp } [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name>* **sipcid|sipuid** *[first choice method][threshold]*

Operation	Command
	slb group method *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** *[granularity] [{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|tcp|https|tcps|ftp|udp|dns|sip-tcp|sip-udp|rtsp}** *<virtual_name> <vip> [vport][max_conn]*
Bind the group to the virtual service	**slb policy default** *<virtual_name> <group_name>* **slb policy static** *<virtual_name> <real_name>* **slb policy backup** *<virtual_name> <group_name>*

6.3.2.2 Sample Configuration for SIP Load Balancing via CLI

→ Step 1 Define SIPUDP real services

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

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

→ Step 3 Add SIPUDP real services into the group

```
FBL(config)#slb group member "g1" "r1" 1
FBL(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.

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

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

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

→ Step 6 Configure SIP Multi-register

If the backend servers don't share database, turn on the multi-register function.

```
FBL(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.

```
FBL(config)#sip nat 172.16.30.171 5060 172.16.32.253 5060 udp 60 callid
FBL(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.fortinet.com:554/test.mp3”. FortiBalancer appliance and the client disconnect, and the client begins to communicate with the real server “audio2.fortinet.com:554” In this mode, all the real servers should have public IP addresses which can be accessible from Internet clients.

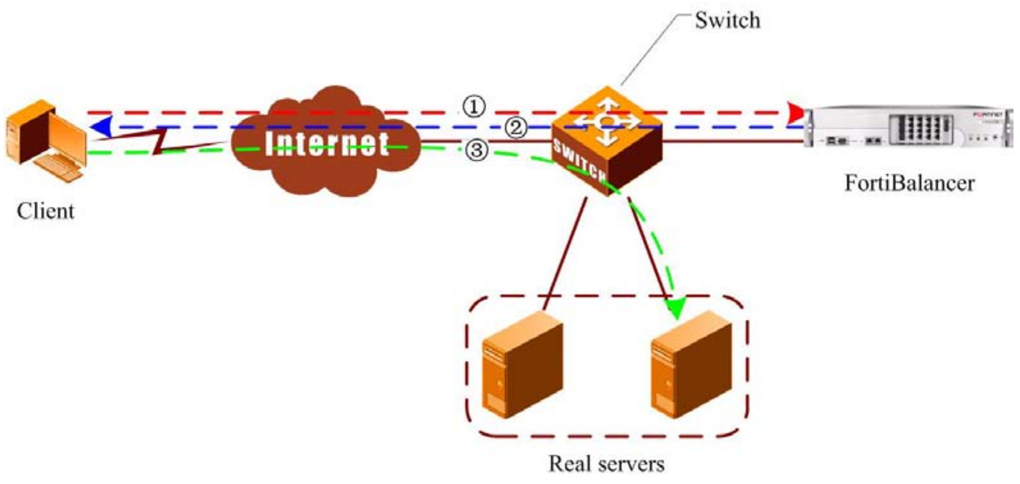


Figure 6-9 RTSP Load Balancing - Redirect Mode

Table 6-5 General Settings of RTSP SLB

Operation	Command
Configure real services	**slb real rtsp** *<real_name> <ip> [port] [max_conn] [rtsp-tcp|tcp|icmp|script-tcp|script-udp|none] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name>* **[rr|pu|sr]** **slb group method** *<group_name>* **lc** *[granularity] [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]*

Operation	Command
	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** *<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 Sample 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]"

```
FBL(config)#slb real rtsp "10.5.1.90" 10.5.1.90 554 1000 rtsp-tcp 3 3
FBL(config)#slb real rtsp "10.5.1.91:554" 10.5.1.91 554
FBL(config)#slb real rtsp "audio1.fortinet.com" 10.5.1.92 554
FBL(config)#slb real rtsp "audio2.fortinet.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 (Constant hash IP), snmp method to choose RTSP real service in one group.

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

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

→ Step 3 Add the real services into the groups

```
FBL(config)#slb group member "mp3_group" "10.5.1.90"
FBL(config)#slb group member "mp3_group" "10.5.1.91:554"
FBL(config)#slb group member "song" "audio1.fortinet.com"
FBL(config)#slb group member "song" "audio2.fortinet.com:554"
```

→ Step 4 Define an RTSP virtual service

The RTSP virtual service default mode is "Redirect", port is 554.

```
FBL(config)#slb virtual rtsp "vs_rtsp1" 10.5.1.80
FBL(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

```
FBL(config)#slb policy filetype "p1" "vs_rtsp1" "mp3_group" "mp3"
FBL(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 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 don’t have to use public IP addresses. The internal private IP addresses will be translated into global IP address on FortiBalancer appliance.

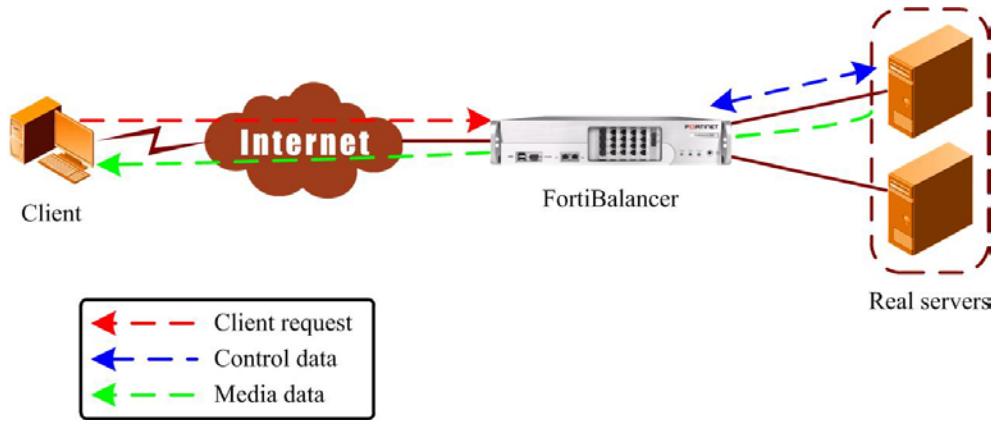


Figure 6-10 RTSP Load Balancing - Dynamic NAT Mode

Table 6-6 General Settings of RTSP SLB

Operation	Command
Configure real services	**slb real rtsp** *<real_name> <ip> [port] [max_conn] [rtsp-tcp|tcp|icmp|script-tcp|script-udp|none] [hc_up] [hc_down] [timeout]*]
Define group methods	**slb group method** *<group_name>* **[rr|sr]** **slb group method** *<group_name>* **lc** *[granularity] [yes|no]* **slb group method** *<group_name>* **pi** *[hash_bits] [rr|sr|lc] [threshold]*

Operation	Command
	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** *<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 Sample for Dynamic NAT Mode via CLI

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

```
FBL(config)#slb real rtsp "rs_rtsp1" 10.5.14.90 554 1000 rtsp-tcp 3 3 60
FBL(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 (Constant hash IP), and snmp method to choose RTSP real service in one group.

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

→ Step 3 Add the real services into the group

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

→ Step 4 Define RTSP virtual services

```
FBL(config)#slb virtual rtsp "vs_rtsp1" 10.3.14.90 554 nat
FBL(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.

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

6.3.4 L2 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 L2SLB 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 FortiBalancer appliance.

For example, the following routes can be added in 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.

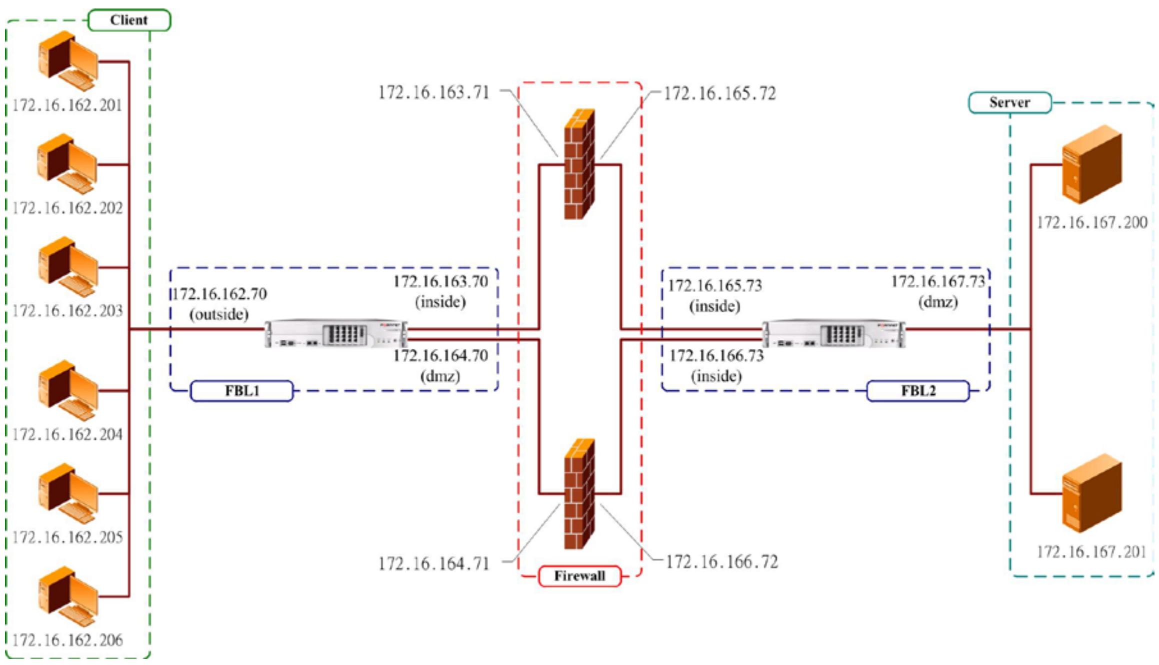


Figure 6-11 L2 IP/MAC-based Load Balancing - Two-arm Network

Note: One real service can only be included by one real service group. L3 real service and L2 real service can't be on the same interface.

Table 6-7 General Settings of L2 IP/MAC SLB

Operation	Command
Configure real services	**slb real l2ip** *<real_name> <real_ip> [hc_up] [hc_down]* **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> <virtual_ip> [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** *<real_name> <ip> <port> [http|https|tcp|icmp|dns|script-tcp|script-udp|script-tcps|radius-auth|radius-acct|sip-tcp|sip-udp|rtsp-tcp] [hc_up] [hc_down]*

Sample Configuration via CLI

Then we begin to configure the left box FBL1 in the above figure.

→ Step 1 Assign IP address to relative interfaces:

```
FBL(config)#ip address outside 172.16.162.70 255.255.255.0
FBL(config)#ip address inside 172.16.163.70 255.255.255.0
FBL(config)#ip address dmz 172.16.164.70 255.255.255.0
```

→ Step 2 Define L2 real services

We can use IP address to define a real service.

```
FBL(config)#slb real l2ip rs1 172.16.163.71 3 3
FBL(config)#slb real l2ip rs2 172.16.164.71 3 3
```

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

```
FBL(config)#slb real l2mac rs1 00:e0:81:03:36:e4 inside
FBL(config)#slb real l2mac rs2 00:30:48:81:54:9c dmz
```

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

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

Note: L2 SLB supports three methods for the group: rr, hi and chi.

When the "**slb group method**" command is used to define an L2 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 L2 real service. Possible values for route mode are: direct and route. "direct" the data packet from a L2 real service will be sent out from the related L2 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

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

→ Step 5 Define L2 virtual service

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

→ Step 6 Define the SLB policy

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

Note: L2 SLB only supports default policy.

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

```
FBL(config)#slb real health rs1 172.16.165.73 0
FBL(config)#slb real health rs2 172.16.166.73 0
```

Here, we have finished the configuration on FBL1 for the L2 IP/MAC based SLB. Now we will begin to configure FBL2:

→ Step 1 Assign IP address to relative interfaces

```
FBL(config)#ip address outside 172.16.165.73 255.255.255.0
FBL(config)#ip address inside 172.16.166.73 255.255.255.0
FBL(config)#ip address dmz 172.16.164.73 255.255.255.0
```

→ Step 2 Define L2 real services

```
FBL(config)#slb real l2ip rs1 172.16.165.72 3 3
FBL(config)#slb real l2ip rs2 172.16.166.72 3 3
Or
FBL(config)#slb real l2mac rs1 00:e0:81:03:36:e5 outside
FBL(config)#slb real l2mac rs2 00:30:48:81:54:9d inside
```

→ Step 3 Define the group for the real service

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

→ Step 4 Add its members to the group

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

→ Step 5 Define L2 virtual service

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

→ Step 6 Define the SLB policy

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

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

```
FBL(config)#slb real health rs1 172.16.163.70 0
FBL(config)#slb real health rs2 172.16.164.70 0
```

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
```

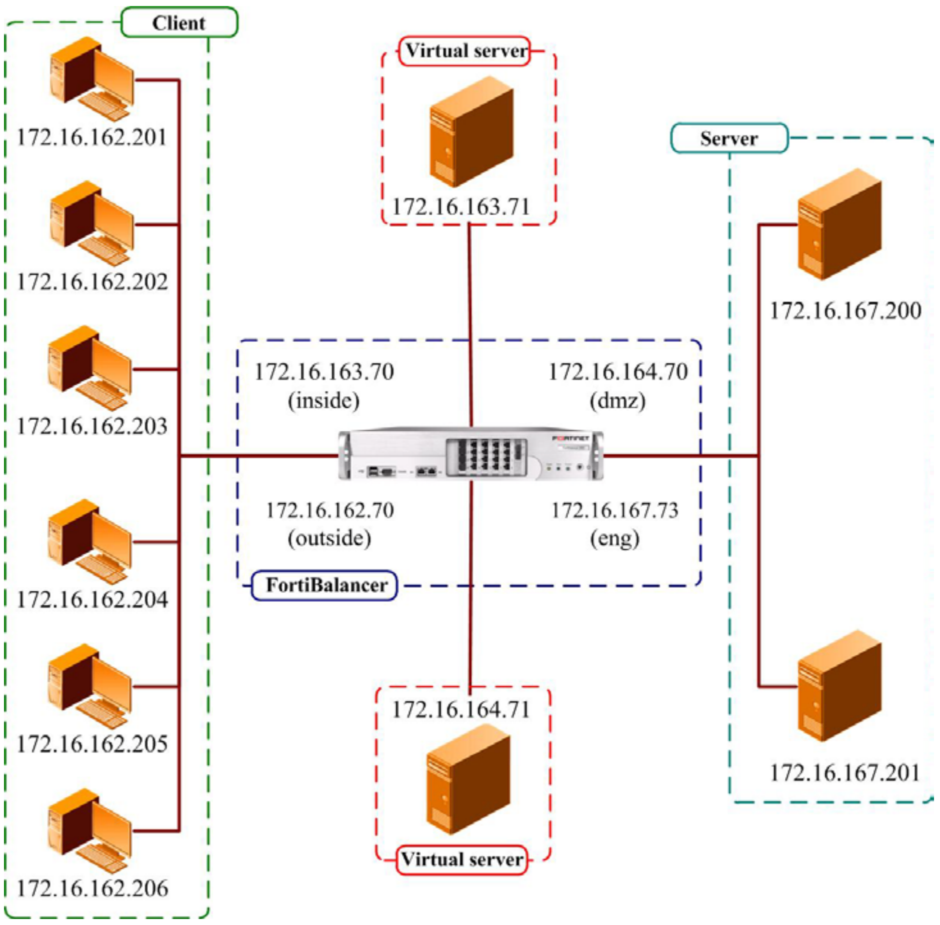


Figure 6-12 IP/MAC-based Load Balancing - Single-arm Network

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

Sample Configuration via CLI

Then we begin to configure FBL1 in the above figure.

→ Step 1 Assign IP address to relative interfaces

```
FBL(config)#ip address "outside" 172.16.162.70 255.255.255.0
FBL(config)#ip address "inside" 172.16.163.70 255.255.255.0
FBL(config)#ip address "dmz" 172.16.164.70 255.255.255.0
FBL(config)#ip address "eng" 172.16.167.73 255.255.255.0
```

→ Step 2 Define L2 real services

```
FBL(config)#slb real l2ip "r1" 172.16.163.71 3 3
FBL(config)#slb real l2ip "r2" 172.16.164.71 3 3
or
FBL(config)#slb real l2mac r1 00:e0:81:03:36:e4 inside
FBL(config)#slb real l2mac r2 00:30:48:81:54:9c dmz
```

→ Step 3 Define the group for the real service

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

→ Step 4 Add the real services into the group

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

→ Step 5 Define L2 virtual service

```
FBL(config)#slb virtual l2ip "v1" 172.16.162.70
FBL(config)#slb virtual l2ip "v2" 172.16.167.73
```

→ Step 6 Define the SLB policy

```
FBL(config)#slb policy default "v1" "g1"
FBL(config)#slb policy default "v2" "g1"
```

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

```
FBL(config)#slb real health r1 172.16.163.71 0
FBL(config)#slb real health r2 172.16.164.71 0
or
FBL(config)#slb real health r1 172.16.163.70 0
FBL(config)#slb real health r2 172.16.164.70 0
```

6.3.5 L3 IP-based Load Balancing

6.3.5.1 Configuration Guidelines

Table 6-8 General Settings of L3 SLB

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** *[granularity] [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 Sample Configuration for L3 IP-based Load Balancing via CLI

→ Step 1 Create real services

```
FBL(config)#slb real ip rip0 10.3.14.10
FBL(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 L3 load balancing by using the "slb group method" command

```
FBL(config)#slb group method gip0
```

→ Step 3 Add the real services into the group

```
FBL(config)#slb group member "gip0" "rip0" 1
FBL(config)#slb group member "gip0" "rip1" 1
```

→ Step 4 Create a virtual service

```
FBL(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 L3 load balancing by using the command "**slb policy default**" or associate the real service to the virtual service by the command "**slb policy static**".

```
FBL(config)#slb policy default vip0 gip0
```

```
Or
FBL(config)#slb policy static vip0 rip0
```

6.3.6 Port Range Load Balancing

6.3.6.1 Configuration Guidelines

Table 6-9 General Settings of Port Range 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} [hc_up] [hc_down]* **slb real http** *<real_name> <ip> [port] [max_conn] [http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp] [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]* **slb real https** *<real_name> <ip> [port] [max_conn] [https|tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp] [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] [hc_up] [hc_down]* **slb real dns** *<real_name> <ip> <port> [max_conn] [dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp] [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]* **slb virtual rtsp** *<virtual_name> <vip> [vport] [mode] [arp|noarp] [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 Sample Configuration for Port Range Load Balancing 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:

```
FBL(config)#slb real http rhttp0 10.3.14.10
```

Rhttp0’s port is defaulted to 80 for HTTP type

```
FBL(config)#slb real http rhttp1 10.3.14.11 90
```

Rhttp1’s port is specified to 90.

When the real services are port range services, health check can only be “icmp” or “none”:

```
FBL(config)#slb real http rhttp0 10.3.14.10 0 1000 icmp
FBL(config)#slb real http rhttp1 10.3.14.11 0 1000 none
```

→ Step 2 Define a group

```
FBL(config)#slb group method ghttp1
```

→ Step 3 Add the real services into the group

```
FBL(config)#slb group member ghttp1 rhttp0
FBL(config)#slb group member ghttp1 rhttp1
```

→ Step 4 Create a virtual service

```
FBL(config)#slb virtual http vhttp0 10.3.14.50 0
```

→ Step 5 Define the port range for the virtual service

An SLB virtual service at most can define three port ranges.

```
FBL(config)#slb virtual portrange vhttp0 80 90
FBL(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 to 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**”.

```
FBL(config)#slb policy default vhttp0 ghttp1
Or
FBL(config)#slb policy static vhttp0 rhttp1
```

6.3.7 Policy Nesting

6.3.7.1 Configuration Guidelines

Table 6-10 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] [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] [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** *<virtual_name> <vip> [vport] [max_conn]*
Create the virtual link	**slb vlink** *<vlink_name>*
Bind the group to the virtual service or the virtual link	**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>*

6.3.7.2 Sample Configuration for Policy Nesting via CLI

To help you better understand the example, the following graphic shows the logical relationship among VIPs, vlinks and groups.

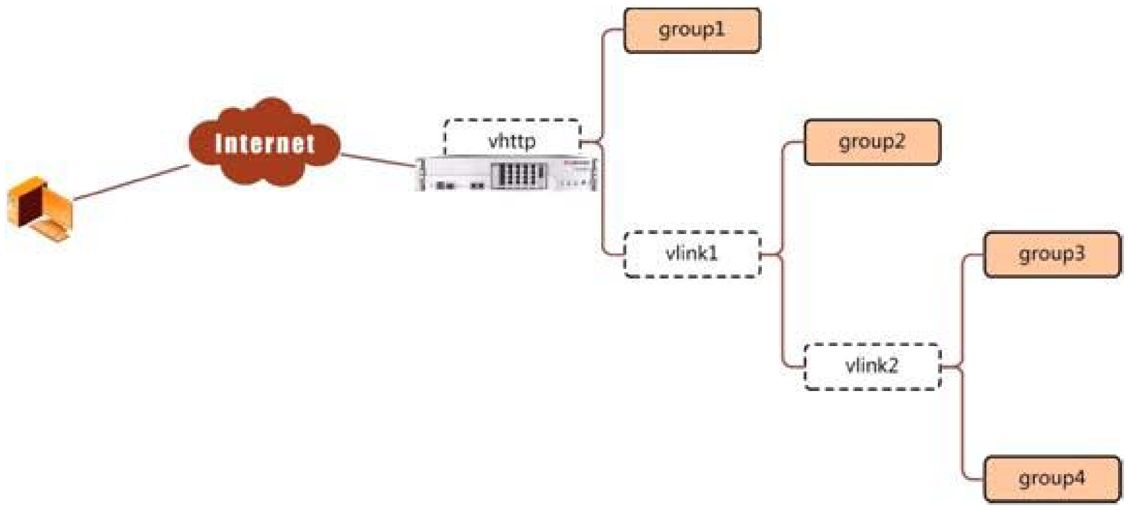


Figure 6-13 Policy Nesting

→ Step 1 Define the real services

Six HTTP real services are configured in this example:

```
FBL(config)#slb real http "rhttp1" 172.16.85.109 80 1000 http 3 3
FBL(config)#slb real http "rhttp2" 172.16.85.110 8081 1000 http 3 3
FBL(config)#slb real http "rhttp3" 172.16.85.111 8080 1000 http 3 3
FBL(config)#slb real http "rhttp4" 172.16.85.112 112 1000 tcp 3 3
FBL(config)#slb real http "rhttp5" 172.16.85.113 113 1000 tcp 3 3
FBL(config)#slb real http "rhttp6" 172.16.85.114 114 1000 tcp 3 3
```

→ Step 2 Create the groups

```
FBL(config)#slb group method group1 rr
FBL(config)#slb group method group2 rr
FBL(config)#slb group method group3 rr
FBL(config)#slb group method group4 rr
```

→ Step 3 Add the real services into the groups

```
FBL(config)#slb group member group1 rhttp1 1
FBL(config)#slb group member group1 rhttp2 2
FBL(config)#slb group member group2 rhttp3 1
FBL(config)#slb group member group2 rhttp4 2
FBL(config)#slb group member group3 rhttp5 1
FBL(config)#slb group member group4 rhttp6 1
```

→ Step 4 Create a virtual service

```
FBL(config)# slb virtual http vhttp 172.16.63.109 80 arp 0
```

→ Step 5 Create the virtual links

```
FBL(config)# slb vlink vlink1
FBL(config)# slb vlink vlink2
```

→ Step 6 Associate a group to a virtual service or a virtual link

```
FBL(config)#slb policy qos network policy1 vhttp vlink1 192.168.2.3 255.255.255.255 1
FBL(config)#slb policy default vhttp group1
FBL(config)#slb policy qos url policy2 vlink1 vlink2 “news” 1
FBL(config)#slb policy default vlink1 group2
FBL(config)#slb policy qos url policy3 vlink2 group3 “sport” 1
FBL(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.

6.4 SLB Summary

SLB Type	Priority (1 is the highest)	Virtual Service	Real Service	Health check	Scenarios
L7 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
L7 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
L7 FTP	2	IP + Port + proto (FTP)	IP + Port + proto (FTP)	None TCP ICMP Additional Script	FTP traffic
L7 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
L7 RTSP	2	IP + Port + proto (RTSP)	IP + Port + proto (RTSP)	None TCP ICMP Additional Script RTSP-TCP	Balance real time media traffic
L4	2	IP + port	IP + Port	None TCP TCPS ICMP	1. Balance traffic according to TCP/UDP headers.

SLB Type	Priority (1 is the highest)	Virtual Service	Real Service	Health check	Scenarios
				Additional Script	2.TCP port or UDP port is specified to determine a particular service
Port range (for L7)	3	L7 VS + Port range	L7 RS L7 RS (0 port)	Non-zero port RS: L7 health check Zero port RS: ICMP Additional	In addition to L7 SLB, cross-port and dynamic port application traffic balance is supported
Port range (for L4)	3	L4 VS + Port range	L4 RS L4 RS (0 port)	Non-zero port RS: L4 health check Zero port RS: ICMP Additional	In addition to L4 SLB, cross-port and dynamic port application traffic balance is supported
L3	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
L2	1	IP + port ranges	IP, MAC	ARP Additional (only ICMP)	1.The backend real services don’t have usable IP addresses so that the traffic can’t 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).

Group Method / Policy		Basic					IP-Based			Header/Request-Based								Cookie-Based				
		rr - Round Robin lc - Least Connections snmp - SNMP sr - Shortest Response Time prox - Proximity					pi - Persistent 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 sipuid - SIP UserID			rc - Rewrite Cookie ec - Embed Cookie ic - Insert Cookie pc - Persistent Cookie hc - Hash Cookie				
		rr	lc	snmp	sr	prox	pi	hi	chi	hh	ph	chh	pu	hq	sslsid	sipcid	sipuid	rc	ec	ic	pc	hc
Basic Policy Types	Static	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Default	OK	OK	OK	OK[1]	OK	OK	OK	OK	OK	OK	OK	N/A	OK	OK	OK	OK	OK	OK	OK	N/A	OK
	Backup	OK	OK	OK	OK[1]	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	OK	OK	OK	OK	OK
Persistent Policy Types	Persistent URL	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	OK	OK	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Persistent Cookie	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	OK	OK
	Rewrite Cookie	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	OK	OK	N/A	N/A	N/A
	Insert Cookie	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	OK	N/A	N/A
QoS Policy Types	QoS Cookie	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	OK	OK
	QoS Hostname	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	N/A	N/A
	QoS URL	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	N/A	N/A
	QoS Network	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	N/A	N/A
	QoS Clientport	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	N/A	N/A
	Regular Expression	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	N/A	N/A
	Header	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	N/A	N/A
Other Policies	Redirect	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	N/A	N/A	N/A	N/A	N/A	N/A	OK	N/A	N/A	N/A

[1] DirectFWD doesn't support Shortest Response Time (sr).

	Filetype	OK	OK	OK	OK	N/A	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Hashurl	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	N/A	N/A	N/A	N/A	N/A	OK	OK	OK	N/A	N/A

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.

7.2 Understanding Reverse Proxy Cache

7.2.1 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.

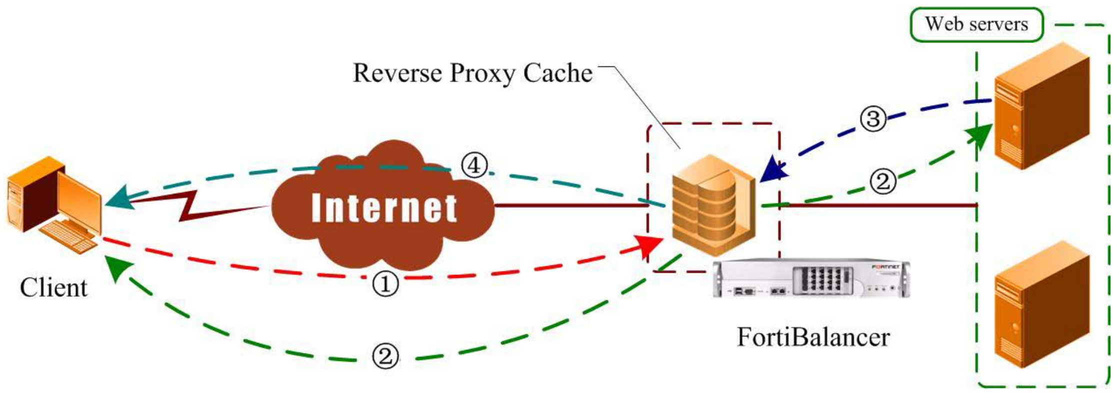


Figure 7-1 Reverse Proxy Cache Working Mechanism

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:

System Memory	Max Size of Cache Object
4GB	10240KB (10MB)
8GB	20480KB (20MB)
16GB	40960KB (40MB)

7.2.2 Advantages of FortiBalancer Reverse Proxy Cache

Compared with traditional cache functions, the FortiBalancer 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 FortiBalancer Reverse Proxy Cache function is mainly featured with the following advantages:

1. **Improved performance**
   - The cache function is an independent module in the FBLOS. 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.
2. **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.
3. **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.

4. **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.

7.2.3 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.

7.2.2.1 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.

7.2.2.2 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.

7.2.4 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.

- 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.

7.2.5 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** *<hostname> <url> {force_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.

If the “ttl” parameter is not specified, the global expiration time specified by “**cache settings expire**” command will apply.

2. For the cache content that does not match any cache filter rule, the expiration time defined in the HTTP header will be applied.

3. If no “Expires” field is available in the HTTP header to define the expiration time, just follow the configuration of “**cache settings expire**”.

7.3 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.

7.3.1 Configuration Guidelines

Table 7-1 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> {force_cache|urlquery|ttl}*
View cache filter configuration	**show cache filter status**
View all cache filters	**show cache filter hostname** *<host name>*

Operation	Command
about the specified host name	
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** *[hostname|all]*

7.3.2 Sample Configuration for Reverse Proxy Cache 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

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".

```
FBL(config)#cache on virtual_MOSS
```

The current status of cache can be viewed by using the "**show cache status**" command.

```
FBL(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.

```
FBL#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 set to 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.

```
FBL(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.

```
FBL(config)#cache settings objectsize 1000
```

Now we use using the "**show cache settings**" command to view current cache settings:

```
FBL(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** *<{on|off}> <virtual_service>*". By default, the cache filter function is disabled.

```
FBL(config)#cache filter on
```

Then, define cache filter rules by using the command "**cache filter rule** *<host name> <url> <{force_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:

```
FBL(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**".

```
FBL(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" "sample" "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 FBLOS™ will allow you to view the running status of the appliance as it pertains to the caching requirements you've configured.

```
FBL(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 DNS Cache

8.1 Overview

The DNS SLB mechanism used by FortiBalancer appliance supports DNS cache feature. Type "A" DNS response, a mapping from host name to an IP address, will be saved in SLB DNS cache for a configurable amount of time. If FortiBalancer appliance receives any DNS type "A" request from clients, the appliance will lookup the response from the DNS cache and send back the cached results to the clients when there is a cache hit. Only when there is a DNS "cache miss" does the FortiBalancer appliance communicate with the remote DNS server(s) directly.

8.2 DNS Cache Configuration

8.2.1 Configuration Guidelines

Table 8-1 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] [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>*

8.2.2 Sample Configuration for DNS Cache via CLI

→ Step 1 Configure necessary SLB component

Since DNS cache is interdependent with SLB configuration strategies, please first refer to the chapter Server Load Balancing for SLB configurations. Below is a sample configuration for DNS cache deployment. First, the SLB component needs to be established.

```
FBL(config)#slb real dns "RS_DNS_1" 10.1.1.10 53 1000 icmp 1 1 20
FBL(config)#slb virtual dns "VS_DNS_1" 10.1.61.100 53
FBL(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 Handbook.

→ 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.

```
FBL(config)#dns cache on
```

→ Step 3 Configure the DNS cache expiration time

```
FBL(config)#dns cache expire 1 36000
```

→ Step 4 Establish hosts for the DNS cache

```
FBL(config)#dns cache host "sting" 10.1.61.200
FBL(config)#dns cache host "gunrose" 10.1.61.100
FBL(config)#dns cache host "roxxette" 10.1.61.2
FBL(config)#dns cache host "queens" 10.1.61.47
```

Chapter 9 HTTP Compression

9.1 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 Data Compression capabilities which are part of FBLOS platform. Configuration of HTTP Data 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.

9.2 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 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 FortiBalancer appliance for certain browsers via "**http compression policy useragent**" 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 don't 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.

9.3 HTTP Compression Configuration

This section covers enabling HTTP Data Compression functionality with default settings and configuring the advanced HTTP compression.

9.3.1 Configuration Guideline

Table 9-1 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>*

9.3.2 Sample Configuration for HTTP Compression via CLI

→ Step 1 Enable HTTP data compression

```
FBL(config)#http compression on
```

This command enables the HTTP Data 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 data compression

```
FBL(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:

```
FBL(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:

```
FBL(config)#http compression advanced useragent on
```

It activates the compression of additional MIME types for a best-known-working-set of browsers summarized in the following table:

	IE Support	**Mozilla Support**
Java Script (JS)	IE 6.0 and above	Mozilla 5.0
Style Sheet (CSS)	IE 6.0 and above	Mozilla 5.0

→ Step 4 Configure url-exclude HTTP compression rule

```
FBL(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.

```
FBL(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.

```
FBL(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.

```
FBL(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 10 Secure Sockets Layer (SSL)

10.1 Overview

Now that the basic SLB and Caching are setup on the FortiBalancer Appliance, we can set up the Secure Sockets Layer (SSL) 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 origin 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.

10.2 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.

10.2.1Cryptography

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.

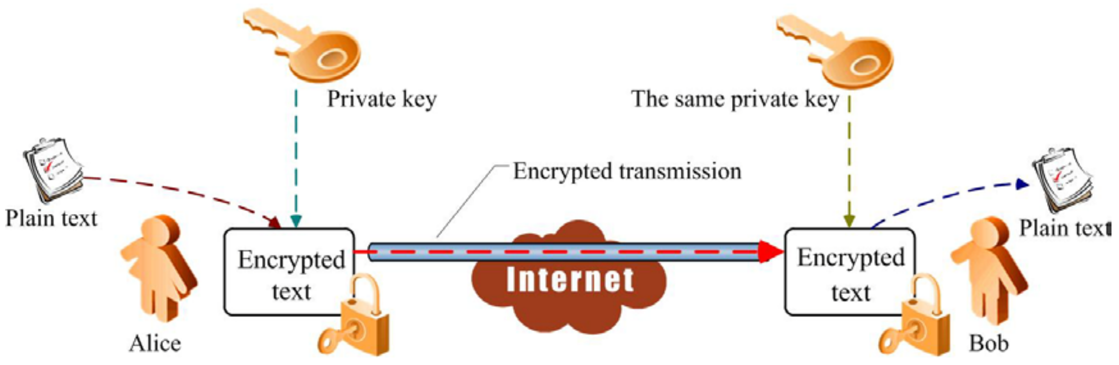


Figure 10-1 Secret Key Encryption/Decryption

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.

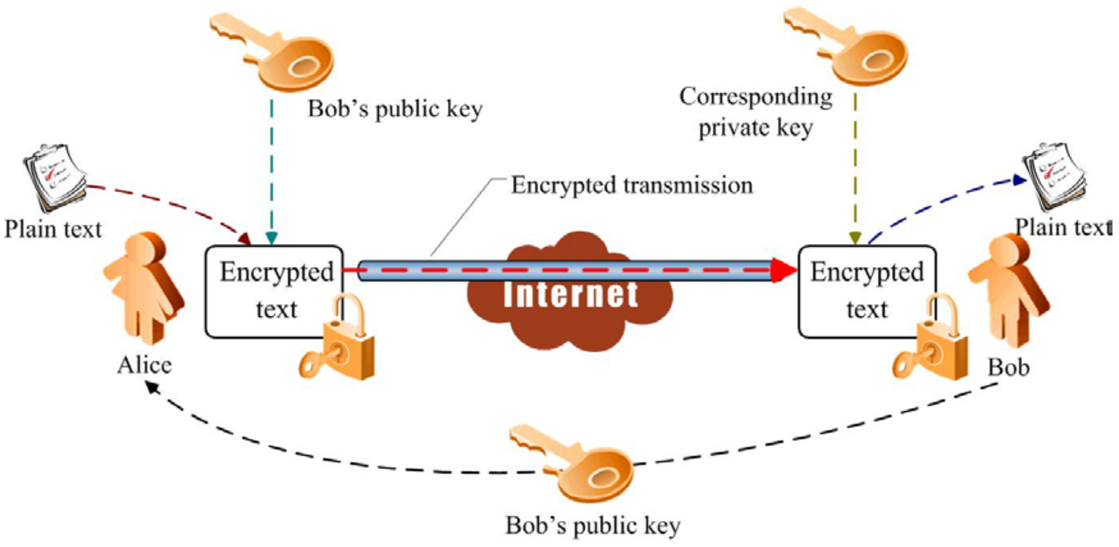


Figure 10-2 Public Key Encryption/Decryption

10.2.2Digital 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.

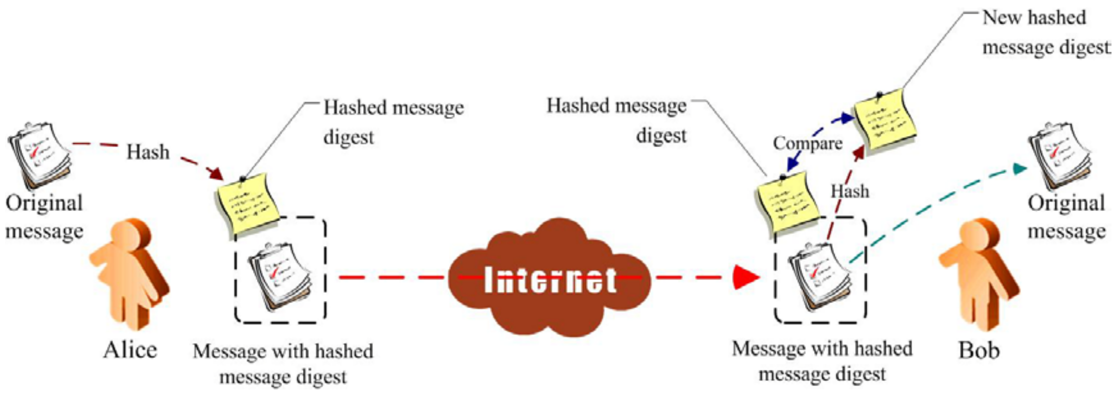


Figure 10-3 Digital Signatures

10.2.3Certificates

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 10-1 X.509 Certificate

Algorithm Identifier
Serial Number
Version
Issuer
Period of Validity
Subject
Subject's Public Key
Issue Unique ID
Subject Unique ID
Extensions
Signature

10.2.3.1 Client Certificate Parse

A backend real service needs information of a client certificate before processing the client requests. But the backend server itself can't 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.

FortiBalancer appliance supports two ways to parse an X.509 certificate: "standard" and "fast". "standard" method means FBLOS will use general ASN.1(Abstract Syntax Notation One) parser to verify the certificate, while "fast" method is to use Fortinet certificate parser to verify the client certificate.

10.3 SSL Acceleration Configuration

10.3.1Configuration Guidelines

Before we get started, let's explain the terminologies used extensively throughout this chapter.

Virtual Host: An SSL host that is 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 that is 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 this is port 443.

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

To setup SSL acceleration there are two methods. The first Method applies if you have never setup 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 10-2 General Settings for 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***<host_name> [key_length]* **ssl import certificate** *<host_name> [tftp_ip] [filename]* **ssl import key** *<host_name> [tftp_ip] [filename]*
Create an SSL real	**slb real https** *<real_name> <ip> [port] [max_conn]*

Operation	Command
host	*[https|tcp|tcps|icmp|script-tcp|script-udp|script-tcps|sip-tcp|sip-udp] [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] [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** *<vhost_name> <cipher_string>* **ssl settings protocol** *<host_name> <version>* **ssl settings reuse** *<host>* **ssl settings clientauth** *<host_name> [subject_filter]* **ssl import rootca** *[host_name] [tftp_ip] [filename]* **ssl settings crl offline** *<host> <Crldp_name> <CRLDistribution_point> [time_interval] [delay_time]* **ssl settings crl online** *<host>* **ssl settings ocsp** *<host> <ocsp server>* **ssl settings minimum** *<host> <key_size> <url>* **ssl start** *<host_name>* **ssl import error** *<error_code> <url>* **ssl load error** *<error_code>*
Advanced configuration for an SSL real host	**ssl settings ciphersuite** *<vhost_name> <cipher_string>* **ssl settings protocol** *<host_name> <version>* **ssl settings reuse** *<host>* **ssl settings clientauth** *<host_name> [subject_filter]* **ssl import rootca** *[host_name] [tftp_ip] [filename]* **ssl settings servername** *<host> <ssl_server_common_name>*

10.3.2Sample Configuration for SSL Acceleration via CLI

10.3.2.1 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**" command to define our SSL virtual host.

```
FBL(config)#slb virtual https virtual1https 10.10.0.10 443
FBL(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.

Note: The command "**ssl host virtual**" creates a private and public key-pair for the newly configured SSL virtual host. The size of this key pair is 1024 bits.

10.3.2.2 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** <*vhost*>" 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 *sample answers*.)

```
FBL(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
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-----

MIIB5TCANU4ANQAwgaQxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJDQTERMA8GA1UEBxMIU2FuI
Epvc2UxFDASBgNVBAoTC0V4YW1wbGUuY29tMRQwEgYDVQQLEwtFeGFtcGxlLmNvbTEnMCUGA1UE
AxMec3NsLXRlc3QucHBiLmFycmF5bmV0d29ya3MubmV0MSAwHgYJKoZIhvcNAQkBFhFhZG1pbkBleGFt
cGxlLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEApk18ozLXGEpJS69BvtfNLcBEjoO82+Q
WRtH4CtIVJYCEOIAlQXQPWsNNN2A74AOW2wkm3f7leSEf2fPI/U6ScHYm8dz2OT523XdDZ/yqmQNwRw
Tz3NC0sNtXKRg9WD9fPMgr6grdBCEH2eVcRdDK8EIXIFrlhXmz+UTxA9y92gMANwEAAaAAMA0GCSqG
SIb3DQEBBAUAA4GBAJCZiGnJ3AHcpuapkjbr31Qr9+1eHl/V6TOesQS/gOlxbOug00T7HndIo32dZ9vnyGZqN
d4CVg9rfFfQWuk09XfDSXdvEFU9ZzedNEr1d5ujbQv8pCsrNIlkHDPDF4Hs2re1ZJeSDpnEEj1EFAFaEyW452
C8v4uGjCe2nrgrksgN

-----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**. FBLOS 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:

```
FBL(config)#ssl start www.example.com
```

The FortiBalancer appliance is configured to take full advantage of the SSL functionality within the FBLOS. 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-----
  MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzET
  MBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVB
  AoTE0NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMw
  IQYDVQQDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJA
  RYaZGV2ZWxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMD
  MwMjA4MTgwMTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQw
  wCgYDVQQHEwNET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1
  UEAxMKMTAuMTIuMC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8w
  DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1M
  HgTPTpVxdtxYhSTSOZwYIXgRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD
  25rCT1rzmalfkTea3V8jHJYP6YinDTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkf
  AgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9
  XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm
  3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnEaiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cR
  QLU=
-----END CERTIFICATE-----
```

<u>Warning</u>: 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.

```
FBL(config)#ssl import certificate www.example.com
            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-----
            MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMBE
            GA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0Nsa
            WNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQQDExp
            kZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2ZWxvcG1lb
            nRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgwMTI5WjB0
            MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEwNET08xCzAJB
            gNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIuMC4xNDEaMBg
            GCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoG
            BAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIXgRqBEdJvs2/ua1XZRz
            LOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6YinDTWKFKztxeUclkzukzP
            UZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAodV5O0LKUr/
            O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4pQc+5KLydK1ZYcTkbxJq41
            K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnEaiYWXmsYGfxPkwdhKlQCLQ
            gN+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”.

```
FBL(config)#ssl import certificate www.example.com 10.10.13.82
            You may overwrite an existing certificate file, type "YES" without quotes to continue:
            YES
```

When successful, you will get a response from the CLI stating so and you can startup the SSL:

```
FBL(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: FBLOS 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** <*vhost name*>” 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 already existing certificate and key for newly created SSL virtual host

```
FBL(config)#ssl import key www.example.com
            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".

```
FBL(config)#ssl import key www.example.com 10.10.13.82
            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:

```
FBL(config)#ssl import certificate www.example.com
            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-----
            MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMBE
            GA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0Nsa
            WNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQQDExp
            kZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2ZWxvcG1lb
            nRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgwMTI5WjB0
            MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEwNET08xCzAJB
            gNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIuMC4xNDEaMBg
            GCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoG
            BAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIXgRqBEdJvs2/ua1XZRz
            LOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6YinDTWKFKztxeUclkzukzP
            UZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAodV5O0LKUr/
            O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4pQc+5KLydK1ZYcTkbxJq41
            K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnEaiYWXmsYGfxPkwdhKlQCLQ
            gN+G3IKu2cRQLU=
            -----END CERTIFICATE-----
```

Note: You must import the key and then import the certificate. The FortiBalancer appliance supposes that the key is imported first.

Then we can start the SSL subsystem:

```
FBL(config)#ssl start www.example.com
```

Now the FortiBalancer appliance is configured to take full advantage of the SSL functionality within the FBLOS. 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 formatted certificate and key through a cut and paste method via the CLI or WebUI. 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 <vhostname>.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.

```
FBL(config)#ssl import certificate www.example.com 10.10.0.3
```

This command will download a file that is named <vhostname>.crt. In our case it is "www.example.com.crt" from the TFTP server (10.10.0.3).

Once the certificate and key import is successful through TFTP server, you need to start the SSL service with the "**ssl start**" command.

```
FBL(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.

```
FBL(config)#ssl import key www.example.com 10.10.0.3
```

This command imports the key from 10.10.0.3 with the filename "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.

```
FBL(config)#ssl import certificate www.example.com 10.10.0.3
```

This command imports the certificate from 10.10.0.3 with the filename "www.example.com.crt".

Once the key is imported, the FBLOS will ask you for a password. This password is the one you use for the database password on the iPlanet server.

Then we can start the SSL subsystem:

```
FBL(config)#ssl start www.example.com
```

Now the FortiBalancer appliance is configured to take full advantage of the SSL functionality within the FBLOS.

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 first refer to the chapter Server Load Balancing for SLB configurations.

10.3.2.3 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 FBLOS 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 chapter of FortiBalancer CLI Handbook.

```
FBL(config)#slb real https real1https 192.168.1.20 443 tcps
FBL(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:

```
FBL(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 first refer to the chapter Server Load Balancing for SLB configurations.

10.3.2.4 Advanced SSL Configuration for an SSL Virtual Host

You can configure different SSL settings for your SSL virtual host.

→ Step 1 Stop SSL virtual host

Before changing any of the configurations, you need to stop the SSL virtual host by using the following command:

```
FBL(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

```
FBL(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:

DES-CBC3-SHA
DES-CBC-SHA
RC4-SHA
RC4-MD5
EXP-DES-CBC-SHA
EXP-RC4-MD5

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:

```
FBL(config)#ssl settings ciphersuite “www.example.com”
“des-cbc3-sha:des-cbc-sha:rc4-sha:rc4-md5:exp-des-cbc-sha:exp-rc4-md5”
```

→ Step 3 Configure protocol version for SSL virtual host

```
FBL(config)#ssl settings protocol "www.example.com" "SSLv3 "
```

The FortiBalancer appliance supports the protocols SSLv3 and TLSv1. You may use one or either of the two protocols.

→ Step 4 Configure session reuse for SSL virtual host

```
FBL(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.

```
FBL(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.

```
FBL(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.

In addition to basic client certificate validation, client certificate authentication is extended to filter the client certificate “Subject” fields as well. A client certificate will be checked against the configured filter information. If no match is made, the client access will be rejected.

The filter rules can be configured with any of the supported RDNs on the FortiBalancer appliances:

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
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:

```
FBL(config)# ssl settings clientauth vhost “/C=US/O=Fortinet/OU=Certificate
```

```
Authority/emailAddress=support@fortinet.com"
```

In this situation, the FortiBalancer appliance will authenticate the client's certificate by using the trusted root certificate. Then the configured subject filter rule will be used to permit (if matching the filter rule) or deny the client's access to the SSL virtual host. For this example, all client certificates with the "C" entry "US", the "O" entry "Fortinet", the "OU"entry "Certificate Authority", and the "emailAddress" entry "support@fortinet.com" will pass the subject filter. Otherwise, the client will not pass 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.

```
FBL(config)#ssl settings cerparse www.example.com
FBL(config)#ssl settings verifymethod www.example.com fast
```

→ Step 7 Configure CRL for SSL virtual host

The FortiBalancer supports the CRL functionality. You can configure the FortiBalancer appliance to fetch the CRL file periodically from a CRL distribution point by using HTTP or FTP.

For our example, let's consider a case when you have put your CRL file (FortiBalancer.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:

```
FBL(config)#ssl settings crl offline www.example.com
"http://www.crldp.com/FortiBalancer.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.

```
FBL(config)#ssl settings crl offline www.example.com "ftp://ftp.crldp.com/FortiBalancer.crl"
1
```

You may also specify an LDAP URL to download the CRL file.

```
FBL(config)#ssl settings crl offline www.example.com
"ldap://ldap.crldp.com/cn=support,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:

```
FBL(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:

```
FBL(config)#ssl settings minimum www.example.com 168 “http://www.example2.com”
```

→ Step 9 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.

```
FBL(config)# ssl start www.example.com
```

10.3.2.5 Advanced SSL Configuration for an SSL Real Host

You can configure different SSL settings for your SSL real host.
→ Step 1 Stop SSL real host

Before changing any of the configurations, you need to stop the SSL real host by using the following command:

```
FBL(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

```
FBL(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

→ Step 3 Configure protocol version for SSL real host

```
FBL(config)#ssl settings protocol "www.myreal.com" "SSLv3 "
```

The FortiBalancer appliance supports the protocols SSLv3 and TLSv1. You may use one or either 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.

```
FBL(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.

```
FBL(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 deploying 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:

```
FBL(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, the SSL subsystem 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.

```
FBL(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.

```
FBL(config)#ssl start www.myreal.com
```

Chapter 11 QoS (Quality of Service)

11.1 Overview

This chapter introduces how to setup the QoS function in 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.

11.2 Understanding QoS

Quality of Service (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, FortiBalancer QoS supports both bandwidth management and priority control.

11.2.1Queuing 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. FortiBalancer 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.

11.2.2Packet 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 5 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.

11.2.3Bandwidth 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.

11.2.4Priority 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.

11.3 QoS Configuration

11.3.1Configuration Guidelines

Table 11-1 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]*

Operation	Command
Enable QoS	**qos enable** *<interface_name> [direction]*

11.3.2 Sample Configuration for QoS via CLI

→ Step 1 Define QoS interfaces

```
FBL(config)# qos interface outside OUT 5Mb
FBL(config)# qos interface outside IN 5Mb
```

→ Step 2 Define outgoing QoS queues

```
FBL(config)# qos queue root qr_oall outside OUT 5Mb 3
FBL(config)# qos queue sub qs_ossh qr_oall 2Mb 3 UNBORROW NONDEFAULT
FBL(config)# qos queue sub qs_oftp qr_oall 512kb 2 UNBORROW NONDEFAULT
FBL(config)# qos queue sub qs_odeflt qr_oall 8kb 3 UNBORROW DEFAULT
```

Default queue is for all the other packets which can't hit any defined queues.

→ Step 3 Define incoming QoS queues

```
FBL(config)# qos queue root qr_iall outside IN 5Mb 3
FBL(config)# qos queue sub qs_issh qr_iall 2Mb 3 BORROW NONDEFAULT
FBL(config)# qos queue sub qs_iftp qr_iall 2Mb 2 BORROW NONDEFAULT
FBL(config)# qos queue sub qs_ideflt qr_iall 8kb 3 BORROW DEFAULT
```

→ Step 4 Define QoS filter rules

```
FBL(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
FBL(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
FBL(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
FBL(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

```
FBL(config)# qos enable outside OUT
FBL(config)# qos enable outside IN
```

Chapter 12 Link Load Balancing

12.1 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

12.2 Understanding Link Load Balancing (LLB)

12.2.1Link Load Balancing

Link Load Balancing (LLB) 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 round robin (rr), weighted round robin (wrr), shortest response (sr) and dynamic detecting (dd). LLB also includes ISP/link failure detection through default gateway and link patch health checking.

12.2.1.1 Outbound Link Load Balancing

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.

12.2.1.2 Inbound Link Load Balancing

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.

12.2.2Link Load Balancing Health Check

Link Health Check verifies that the path between the FortiBalancer interface and some upstream device is available. This can be accomplished by two ways: broadcasting ARP requests at regular intervals and pinging a user-defined upstream IP address. Broadcasting ARP requests at regular intervals can verify the path between FortiBalancer interface and the upstream ISP router is available. Pinging a user-defined upstream IP address can not only verify the path between FortiBalancer interface and the upstream ISP router is available, but also verify the path between upstream ISP router and user-defined upstream IP address is available. 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.

12.2.3Link Load Balancing Methods

Outbound Link Load Balancing supports four load balancing methods:

- Round Robin
- Weighted Round Robin
- Shortest Response Time
- Dynamic Detecting

Inbound Link Load Balancing supports two load balancing methods:

- Round Robin
- Weighted Round Robin

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.

Dynamic Detect 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.

Note: The LLB dd method should work with NAT configuration. If there is no NAT configuration related with LLB link route, the DD method could not work normally.

12.2.4Link Load Balancing Policies

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:

```
FBL(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
```

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.

12.2.5Route 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 droutes (Direct Route) and will have priority 2000.

The following table shows the priority of different types of routes:

Table 12-1 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
DYNAMIC ROUTE	101-132
LLB LINK ROUTE	2
DEFAULT ROUTE	1

12.3 Link Load Balancing Configuration

12.3.1Outbound 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 12.3.2.

12.3.1.1 Configuration Guidelines

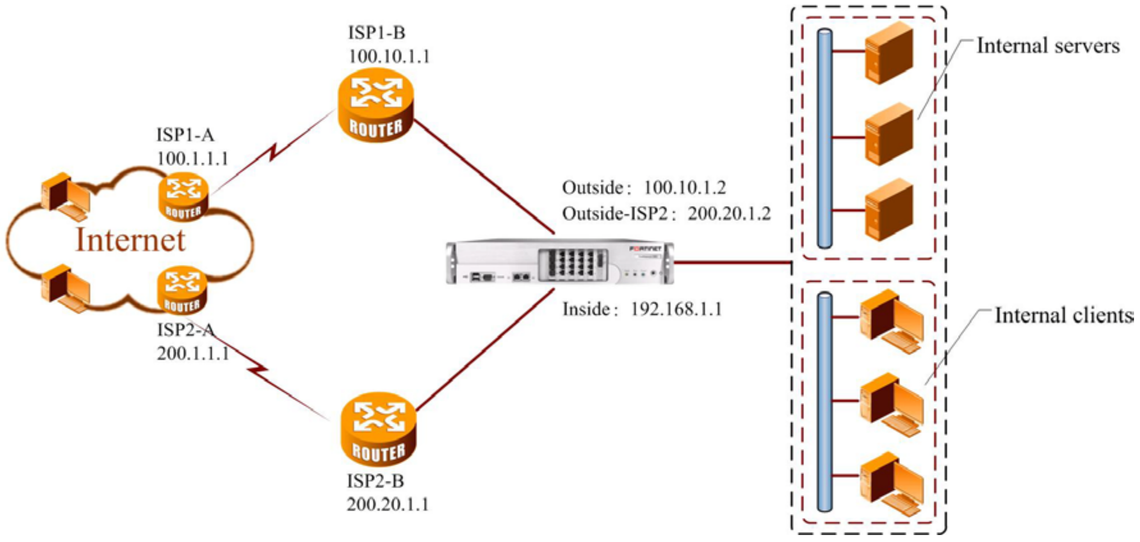


Figure 12-1 Outbound Link Load Balancing (One FortiBalancer Appliance)

Table 12-2 General Settings of Outbound Link Load Balancing

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> <netmask>*
Configure MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
Configure LLB health check	**llb link route** *<link_name> <route_ip> <hc_ip> [hc_interval] [weight] [hc_srcip]* **llb link health {on|off}** **llb link health checker** *<link_name> <hc_ip>*
Configure outbound LLB method	**llb method outbound {rr|wrr|sr|dd}** *[time_interval] [count_interval]*
Configure NAT	**nat port** *<vip> <network_ip> <netmask> [timeout] [gateway]*

Enable IPflow and RTS	**ip ipflow {on|off}**
	ip rts {on|off}

12.3.1.2 Sample Configurations via the CLI

→ Step 1 Configure interface IP addresses

The Outside interface IP will have an address from ISP1's address range. In order to assign an additional IP address on the Outside 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.

```
FBL(config)#ip address outside 100.10.1.2 255.255.255.0
FBL(config)#mnet outside outside_isp2
FBL(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

Then, configure the IP address of the Inside interface.

```
FBL(config)#ip address inside 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:

```
FBL(config)#llb link route ISP1 100.10.1.1 100.1.1.1 10 1 100.10.1.2
```

Perform health check on an IP address on the other side of ISP2's WAN router:

```
FBL(config)#llb link route ISP2 200.20.1.1 200.1.1.1 10 2 200.20.1.2
```

Enter the following command to enable link health check:

```
FBL(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.

```
FBL(config)#llb link health checker ISP1 100.1.1.2 10
FBL(config)#llb link health checker ISP1 100.1.1.3 10
FBL(config)#llb link health checker 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 four methods:

- Round Robin (rr)
- Weighted Round Robin (wrr)
- Shortest Response (sr)
- Dynamic Detecting (dd)

The default method is "rr".

In this example, we use the "wrr" method.

```
FBL(config)#llb method outbound wrr
```

→ Step 5 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:

```
FBL(config)#nat port 100.10.1.10 192.168.1.0 255.255.0.0
```

NAT for ISP2:

```
FBL(config)#nat port 200.20.1.10 192.168.1.0 255.255.0.0
```

→ 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, this function is turned on.

```
FBL(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, this function is turned on.

```
FBL(config)#ip rts on
```

12.3.2Outbound 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.

12.3.2.1 Configuration Guidelines

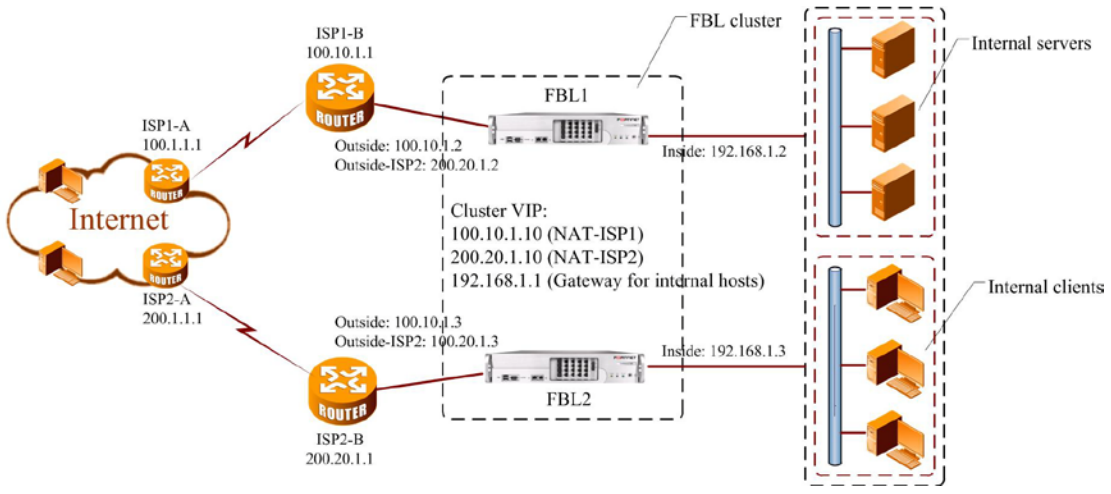


Figure 12-2 Outbound Link Load Balancing (Two FortiBalancer Appliances)

Table 12-3 General Settings of Outbound Link Load Balancing

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> <netmask>*
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> <hc_ip> [hc_interval] [weight] [hc_srcip]* **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 NAT	**nat port** *<vip> <network_ip> <netmask> [timeout] [gateway]*

Enable IPflow and RTS	**ip ipflow {on|off}**
	ip rts {on|off}

12.3.2.2 Sample Configurations 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.

(FBL#1) Outside and Inside IP address configuration:

```
FBL1(config)#ip address outside 100.10.1.2 255.255.255.0
FBL1(config)#mnet outside outside_isp2
FBL1(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
FBL1(config)#ip address inside 192.168.1.2 255.255.255.0
```

(FBL#2) Outside and Inside IP address configuration:

```
FBL2(config)#ip address outside 100.10.1.3 255.255.255.0
FBL2(config)#mnet outside outside_isp2
FBL2(config)#ip address outside_isp2 200.20.1.3 255.255.255.0
FBL2(config)#ip address inside 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.

(FBL#1) 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.

```
FBL1(config)#cluster virtual ifname inside 1
FBL1(config)#cluster virtual vip inside 1 192.168.1.1
FBL1(config)#cluster virtual priority inside 1 200
FBL1(config)#cluster virtual on 1 inside
```

(FBL#2) 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.

```
FBL2(config)#cluster virtual ifname inside 1
FBL2(config)#cluster virtual vip inside 1 192.168.1.1
FBL2(config)#cluster virtual priority inside 1 100
FBL2(config)#cluster virtual on 1 inside
```

→ 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:

```
FBL1(config)#llb link route ISP1 100.10.1.1 100.1.1.1 10
FBL1(config)#llb link route ISP2 200.20.1.1 200.1.1.1 10
FBL1(config)#llb link health on

FBL2(config)#llb link route ISP1 100.10.1.1 100.1.1.1 10
FBL2(config)#llb link route ISP2 200.20.1.1 200.1.1.1 10
FBL2(config)#llb link health on
```

→ Step 4 Configure cluster Virtual IPs for NATing traffic

(FBL#1) Cluster VIPs for NAT on each ISP. Assign a higher priority than the secondary FortiBalancer appliance.

```
FBL1(config)#cluster virtual ifname outside 10
FBL1(config)#cluster virtual vip outside 10 100.10.1.10
FBL1(config)#cluster virtual prio outside 10 200
FBL1(config)#cluster virtual on 10 outside
FBL1(config)#cluster virtual ifname outside-isp2 20
FBL1(config)#cluster virtual vip outside-isp2 20 200.20.1.10
FBL1(config)#cluster virtual prio outside-isp2 20 200
FBL1(config)#cluster virtual on 20 outside-isp2
```

(FBL#2) Cluster VIPs for NAT on each ISP. Assign them a lower priority than the primary FortiBalancer appliance.

```
FBL2(config)#cluster virtual ifname outside 10
```

```
FBL2(config)#cluster virtual vip outside 10 100.10.1.10
FBL2(config)#cluster virtual prio outside 10 100
FBL2(config)#cluster virtual on 10 outside
FBL2(config)#cluster virtual ifname outside-isp2 20
FBL2(config)#cluster virtual vip outside-isp2 20 200.20.1.10
FBL2(config)#cluster virtual prio outside-isp2 20 100
FBL2(config)#cluster virtual on 20 outside-isp2
```

→ Step 5 Configure NAT for outbound LLB sessions

(Both FortiBalancers) NAT rules for ISP1 and ISP2:

```
FBL1(config)#nat port 100.10.1.10 192.168.1.0 255.255.0.0
FBL1(config)#nat port 200.20.1.10 192.168.1.0 255.255.0.0

FBL2(config)#nat port 100.10.1.10 192.168.1.0 255.255.0.0
FBL2(config)#nat port 200.20.1.10 192.168.1.0 255.255.0.0
```

→ 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, this function is turned on.

```
FBL_(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, this function is turned on.

```
FBL_(config)#ip rts on
```

12.3.3Inbound LLB Configuration

In this implementation example, a single FortiBalancer appliance will be configured to load balance inbound traffic.

12.3.3.1 Configuration Guidelines

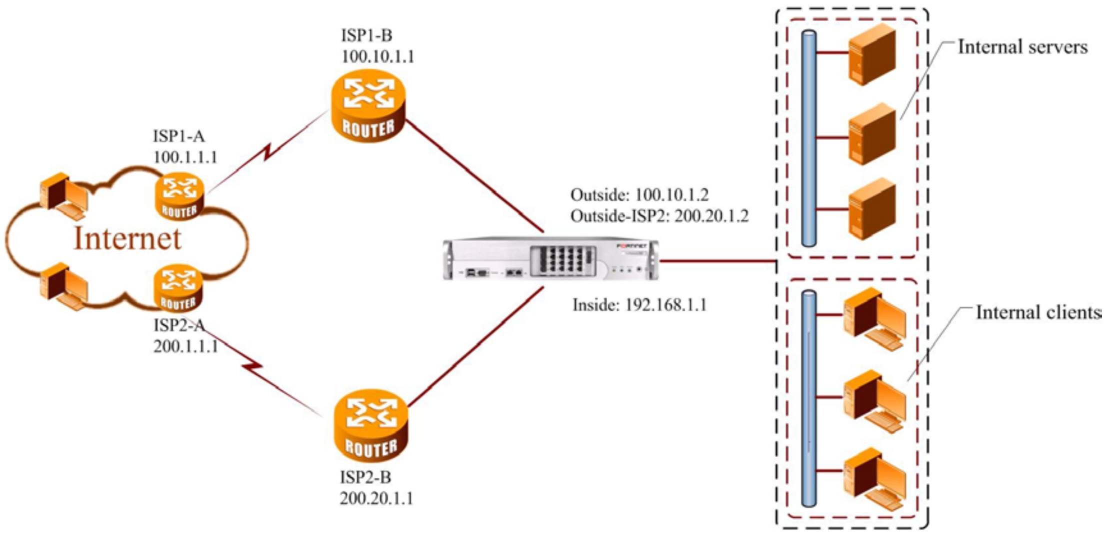


Figure 12-3 Inbound Link Load Balancing

Table 12-4 General Settings of Inbound Link Load Balancing

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> <netmask>*
Configure MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
Configure LLB health check	**llb link route** *<link_name> <route_ip> <hc_ip> [hc_interval] [weight] [hc_srcip]* **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] [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] [linkname]* **llb dns ttl** *<host name> [seconds]*
Configure load balancing method	**llb method inbound {rr|wrr}**

Enable IPflow and RTS	**ip ipflow {on|off}**
	ip rts {on|off}

12.3.3.1 Sample Configurations via the CLI

Follow these steps to configure Inbound Link Load Balancing with a single FortiBalancer appliance.

→ Step 1 Configure interface IP addresses

The Outside interface IP will have an address from ISP1's address range. In order to assign an additional IP address on the Outside 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.

```
FBL(config)#ip address outside 100.10.1.2 255.255.255.0
FBL(config)#mnet outside outside_isp2
FBL(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

Then, configure the IP address of the Inside interface.

```
FBL(config)#ip address inside 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:

```
FBL(config)#llb link route ISP1 100.10.1.1 100.1.1.1 10
```

Perform health check on an interface on the other side of ISP2's WAN router:

```
FBL(config)#llb link route ISP2 200.20.1.1 200.1.1.1 10
```

Enter the following command to enable link health check:

```
FBL(config)#llb link health on
```

→ Step 3 Configure Server Load Balance

```
FBL(config)#slb virtual http vip1 100.10.1.10
FBL(config)#slb virtual http vip2 200.20.1.10
FBL(config)#slb real http server1192.168.1.100
FBL(config)#slb policy static vip1 server1
FBL(config)#slb policy static vip2 server1
```

→ Step 4 Configure LLB DNS host and TTL for inbound

```
FBL(config)# llb dns host llb.fortinet.com 100.10.1.10 2
FBL(config)# llb dns host llb.fortinet.com 200.20.1.10 1
FBL(config)# llb dns ttl llb.fortinet.com 60
```

→ Step 5 Configure inbound load balancing method

```
FBL(config)#llb method inbound wrr
```

→ 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, this function is turned on.

```
FBL(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, this function is turned on.

```
FBL(config)#ip rts on
```

Chapter 13 Global Service Load Balance (GSLB)

13.1 Overview

Global Server Load Balance (GSLB) 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.

13.2 Understanding Global Server Load balance

In FortiBalancer GSLB solution, 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.

Smart DNS 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 a FortiBalancer appliance joins the Smart DNS 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 Smart DNS process works as follows:

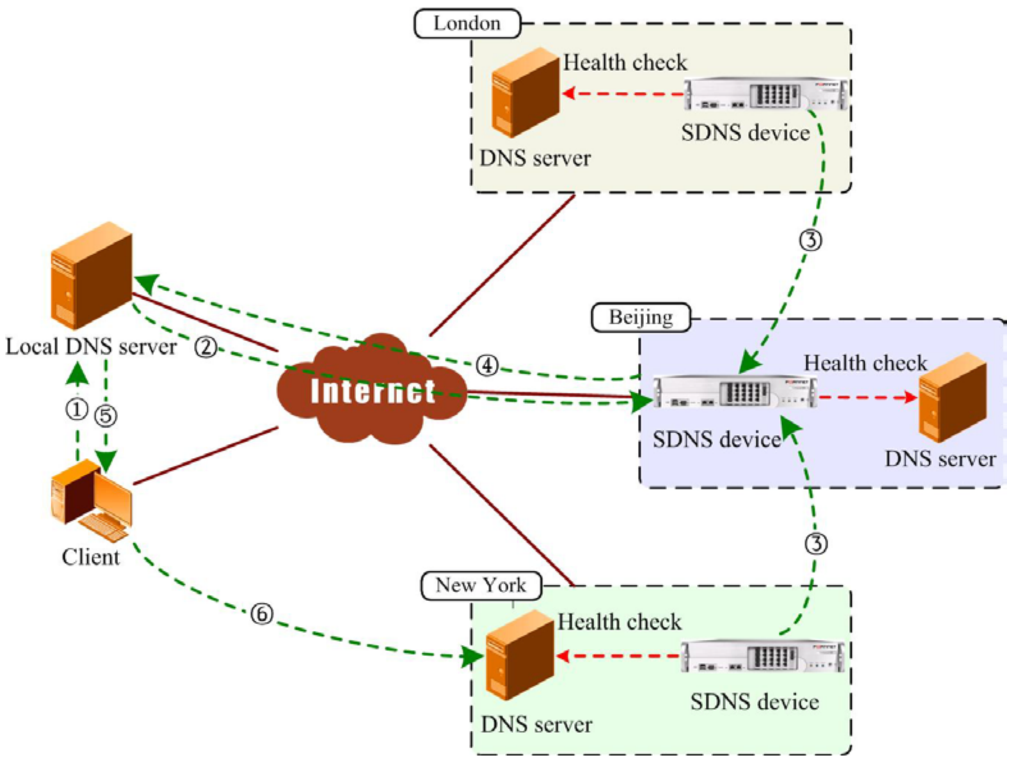


Figure 13-1 FortiBalancer SDNS Working Mechanism

As shown in the above figure, the FortiBalancer 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 DNS servers in its local Domain Name and IP Service Database, and then forwards the request to a proper DNS server based on pre-configured load balancing algorithms. In the above example, the DNS server at New York is selected.
4. The SDNS server at Beijing returns back the IP addresses of the DNS server at New York to the local DNS server of the client.
5. Upon receiving the response, the local DNS 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 a FortiBalancer appliance which participates in the FortiBalancer GSLB management.

13.2.1Smart DNS Member Reporter-Receiver Hierarchy

All SDNS members can be divided into two groups: Smart DNS server and HTTP proxy cache server.

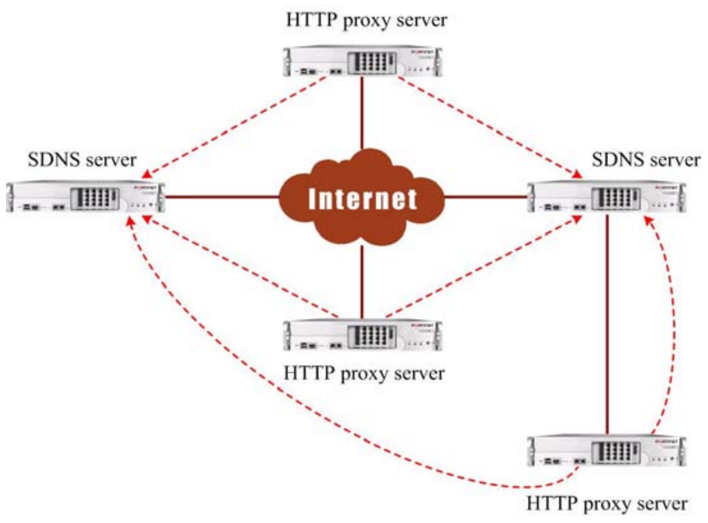


Figure 13-2 SDNS Reporter-Receiver Hierarchy

Smart DNS Servers

Smart DNS servers are responsible for DNS resolving. Every HTTP proxy cache server will report its status information to Smart DNS 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 Smart DNS servers. The HTTP proxy cache servers will collect the local status information and send it to Smart DNS servers at specified frequency.

If a 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 Smart DNS servers (including itself) and collect the status information from all the proxy cache servers.

13.2.2Smart DNS Load Balancing Methods

Server load balancing methods are supported by Smart DNS 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)

Global Round Robin (GRR)

When using this method, Smart DNS 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 Smart DNS servers. HTTP proxy cache servers will regularly report its local virtual service and link load/health information to Smart DNS 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.

VIP-based Weighted Global Round Robin (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.

Global Connection Overflow (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.

Global Least Connection (GLC)

This load balance 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.

IP Overflow (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 "**llb dns host**" command. In the "VWGRR" method, the last argument of the "**llb dns host**" command means the weight of an IP address. For "IPO" method, the last argument of the "**llb dns host**" command means the priority. IPO methods will resolve a host name to the related healthy IP address with the highest priority.

Proximity (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.

Region (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.

13.2.3Smart DNS Member

Actually, a member is a FortiBalancer appliance. A member can be configured to be a Smart DNS server or an HTTP proxy cache server. It also can be configured to be a Smart DNS server and an HTTP proxy cache server at the same time.

13.2.4Smart DNS 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 Smart DNS servers. On HTTP proxy cache servers, these commands are invalid although they can be configured.

13.2.5Smart DNS Proximity

"proximity" is used to determine the site or region which a request source IP address belongs to.

13.2.6Smart DNS 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.

13.2.7Smart DNS 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 Smart DNS servers. They are invalid on HTTP proxy cache servers.

13.2.8Smart DNS 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 Smart DNS servers than the standby sites.

13.2.9Smart DNS 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, IP and host, and collect the traffic statistics on a member, site, region, IP and host, and balance the Web requests to different sites and regions based on the current bandwidth usage.

Five 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)
5) Bandwidth of a host for all the regions or for only one region/site

The system will collect the traffic of a domain name (a host). 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.

13.2.10 Smart DNS Alias

Smart DNS Alias is only useful for SDNS region method with bandwidth management configured.

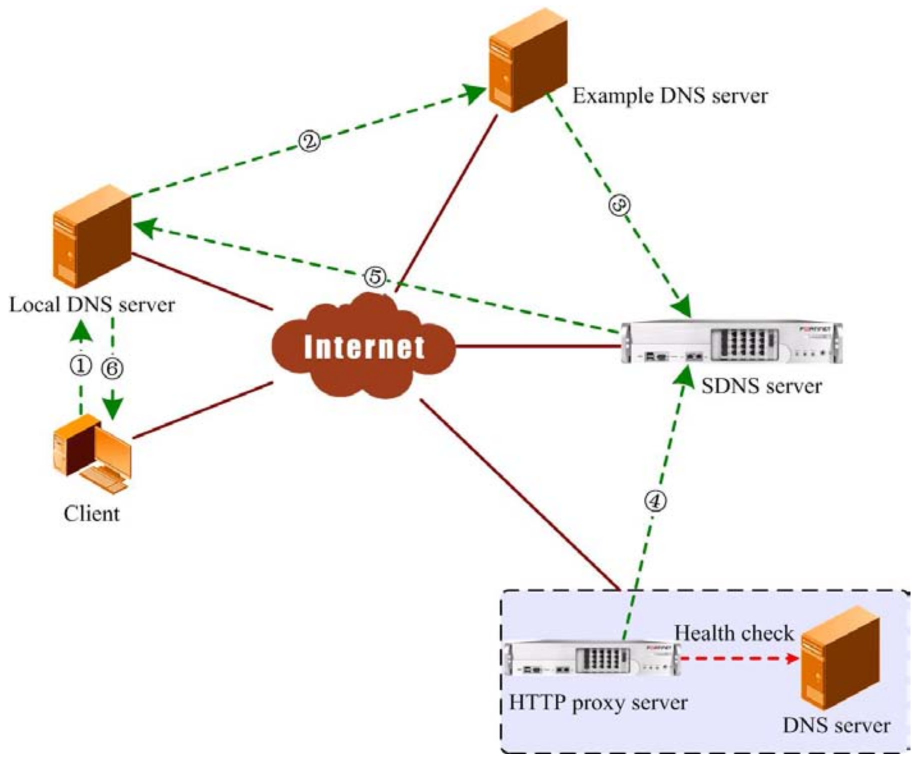


Figure 13-3 SDNS Alias

Now let's see an example:

The client firstly sends a "www.a.com" resolving request to Example Server which will switch "www.a.com" to "www.example.a.com" and send the "www.example.a.com" resolving request to Smart DNS. Then the resolving result 10.10.10.10 is sent back to the client from Smart DNS through Example Server. The client will still use "www.a.com" to access the proxy server. In order to report the correct SDNS status information to Smart DNS for the domain "www.example.a.com", "www.example.a.com" should be configured on the proxy cache server instead of "www.a.com". But the problem is that the proxy server can't collect the traffic statistics on users' access to "www.a.com" (because this domain name is not configured on it). To solve this problem, an SDNS alias should be defined on the proxy cache server to indicate the relationship between the two domains.

13.2.11 Smart DNS 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 Smart DNS, 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)

By using this pool method, IP address with the highest weight in a pool is selected in the first query. Then the local DNS will get the same IP address with the highest weight for all the queries after the first query 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 persistence timeout**".

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.

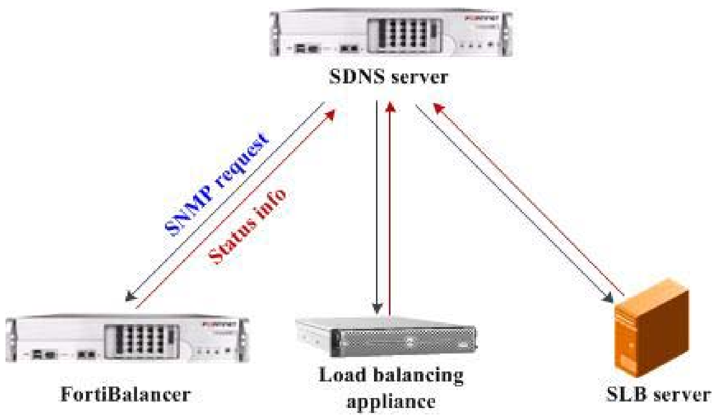


Figure 13-4 SDNS SNMP

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.

13.2.12 Smart DNS 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.

13.2.13 Smart DNS Dynamic Proximity System (DPS)

SDNS Dynamic Proximity aims at providing a dynamically generated proximity rule table, instead of statically configured proximity rules for Fortinet SDNS. All the dynamic proximity entries in the table are collected by proximity detection methods.

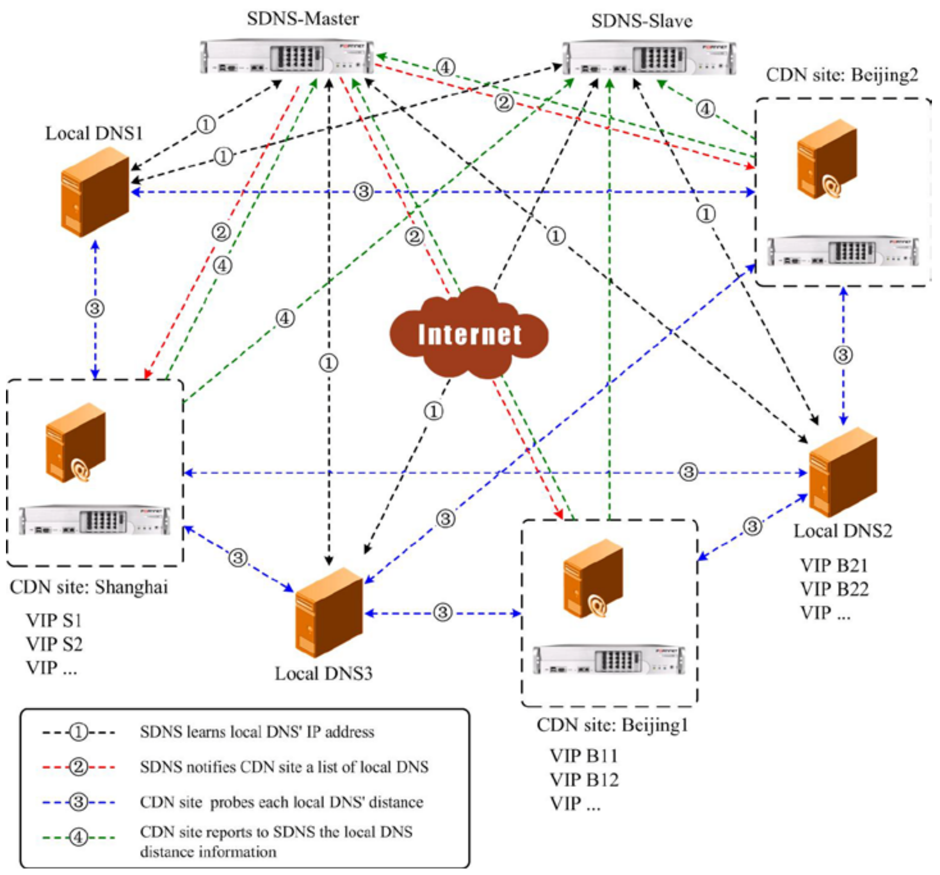

SDNS-Master
SDNS-Slave
CDN site: Beijing2
Local DNS1
Internet
CDN site: Shanghai
VIP S1
VIP S2
VIP ...
Local DNS3
Local DNS2
VIP B21
VIP B22
VIP ...
CDN site: Beijing1
VIP B11
VIP B12
VIP ...
①
②
③
④
---①--- SDNS learns local DNS' IP address
---②--- SDNS notifies CDN site a list of local DNS
---③--- CDN site probes each local DNS' distance
---④--- CDN site reports to SDNS the local DNS distance information

Figure 13-5 SDNS DPS System

For Smart DNS 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

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

13.2.13.1 How Smart DNS 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 won't be turned on automatically, and it can be turned on by using the command "**sdns dps on**".

13.2.14 Smart DNS Full-DNS

Smart DNS 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 Fortinet SDNS. Any query packets of A, AAAA and CNAME record types are processed by Fortinet SDNS, while any other query packets are processed by BIND 9.

Smart DNS supports recursive query. By enabling the recursion, if a client sends a query to the FortiBalancer appliance and it cannot resolve the query by itself, the FortiBalancer appliance can work as a local DNS to perform a recursion by contacting other DNS servers on behalf of the requesting client to fully resolve the query, and then send an answer back to the client.

For every domain name configured in SDNS, there should be at least one record of the domain name in the BIND 9 zone file which the domain name belongs to. For example, a domain name "image.example.com" is already configured in SDNS, and it also belongs to the zone file "example.com". If there is no record of this domain name in its zone file, it is recommended to add a TXT type record for it as follows:

```
image        IN        TXT        "A text string"
```

Note: The "text string" can be any descriptions about this domain name.

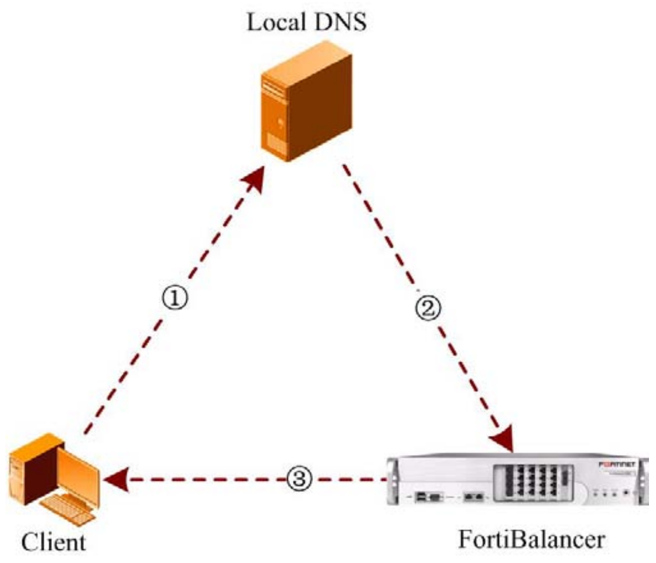


Figure 13-6 Full-DNS Working Flow

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. Fortinet FortiBalancer returns the corresponding DNS record responses. (**Note:** If the DNS query's type is AAAA record, Fortinet FortiBalancer can return AAAA record or CNAME record response; if the DNS query is A record, Fortinet FortiBalancer may return CNAME record response; if the DNS query is CNAME record, Fortinet FortiBalancer returns CNAME record response.)

Note: BIND 9 can only be configured via WebUI, and CNAME can be configured via WebUI or CLI.

13.3 Global Server Load Balance Configuration

13.3.1Configuration Guidelines

13.3.1.1 Topology 1

Three FortiBalancer appliances are configured as "all"-type SDNS member in the following figure.

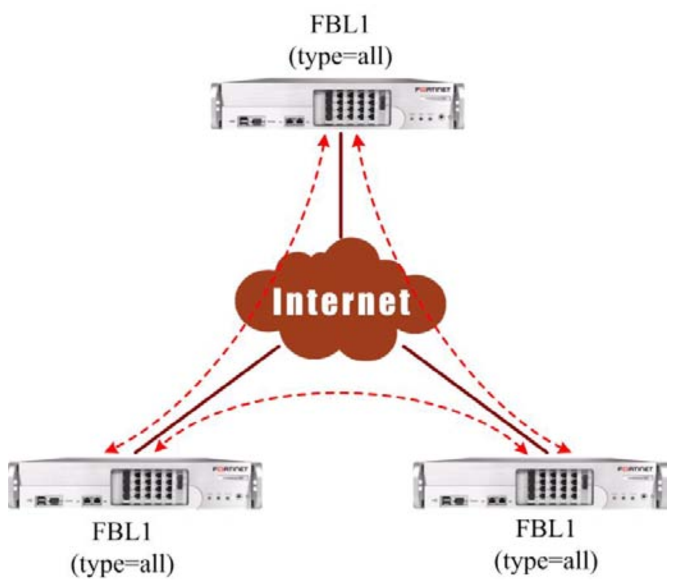


Figure 13-7 Network Topology 1

The inside IP addresses of the three FortiBalancer appliances are as follows:

FBL1: 10.3.200.1
FBL2: 10.3.200.2
FBL3: 10.3.200.3

13.3.1.2 Topology 2

Among the three FortiBalancer appliances in the following figure, FBL1 is configured as "dns"-type SDNS member while FBL2 and FBL3 are configured as "proxy"-type SDNS members.

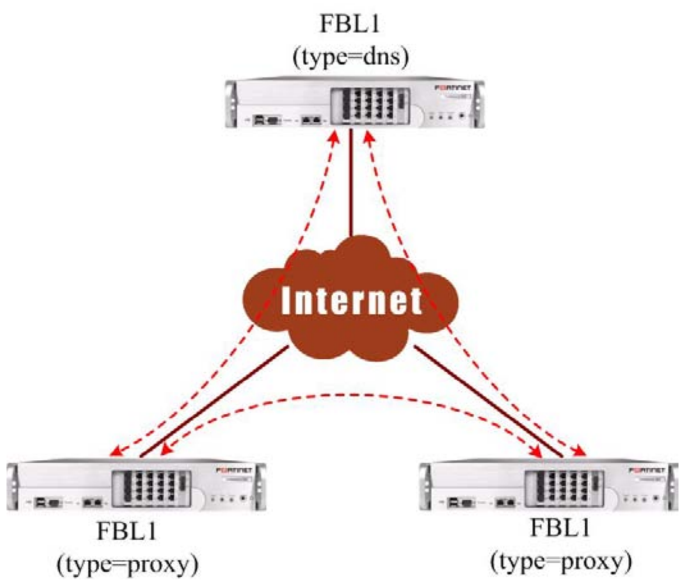


Figure 13-8 Network Topology 2

The inside IP addresses of the three FortiBalancer appliances are as follows:

FBL1: 10.3.200.1
FBL2: 10.3.200.2
FBL3: 10.3.200.3

13.3.1.3 DPS Configuration Topology

Table 13-1 General Settings for GSLB

Operation	Command
Configure SLB	Refer to Server Load Balancing Configuration in in chapter Server Load Balancing.
Configure LLB	Refer to Link Load Balancing Configuration in chapter Link Load Balancing.
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|host} {region|site|member|host name|ip address} <mode> <maxbandwidth> [region|site]*
Configure alias	**sdns alias** *<alias name> <host name>*
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]*

13.3.2Sample Configuration for GSLB based on Topology1 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.

13.3.2.1 Configuring SLB

FBL1

→ **Step 1 Configure a real server**

```
FBL(config)#slb real http “rs1” 10.3.200.110 8080 1000 tcp 1 1
```

→ **Step 2 Configure multiple virtual servers**

```
FBL(config)#slb virtual http “vs1” 10.3.210.1 80
FBL(config)#slb virtual http “vs2” 10.3.210.2 80
FBL(config)#slb virtual http “vs3” 10.3.210.3 80
```

→ **Step 3 Configure SLB policy**

```
FBL(config)# slb policy static “vs1” “rs1”
FBL(config)# slb policy static “vs2” “rs1”
FBL(config)# slb policy static “vs3” “rs1”
```

FBL2

→ **Step 1 Configure a real server**

```
FBL(config)#slb real http “rs1” 10.3.200.110 8080 1000 tcp 1 1
```

→ **Step 2 Configure multiple virtual servers**

```
FBL(config)#slb virtual http “vs1” 10.3.220.1 80
FBL(config)#slb virtual http “vs2” 10.3.220.2 80
FBL(config)#slb virtual http “vs3” 10.3.220.3 80
```

→ **Step 3 Configure SLB policy**

```
FBL(config)# slb policy static “vs1” “rs1”
FBL(config)# slb policy static “vs2” “rs1”
FBL(config)# slb policy static “vs3” “rs1”
```

FBL3

→ **Step 1 Configure a real server**

```
FBL(config)#slb real http “rs1” 10.3.200.110 8080 1000 tcp 1 1
```

→ **Step 2 Configure multiple virtual servers**

```
FBL(config)#slb virtual http “vs1” 10.3.230.1 80
FBL(config)#slb virtual http “vs2” 10.3.230.2 80
FBL(config)#slb virtual http “vs3” 10.3.230.3 80
```

→ Step 3 Configure SLB policy

```
FBL(config)# slb policy static “vs1” “rs1”
FBL(config)# slb policy static “vs2” “rs1”
FBL(config)# slb policy static “vs3” “rs1”
```

13.3.2.2 Configuring LLB

FBL1

→ Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned three IP addresses here.

```
FBL(config)#llb dns host “www.a.com” 10.3.210.1
FBL(config)#llb dns host “www.a.com” 10.3.210.2
FBL(config)#llb dns host “www.a.com” 10.3.210.3

FBL(config)#llb dns host “www.b.com” 10.3.210.1
FBL(config)#llb dns host “www.b.com” 10.3.210.2
FBL(config)#llb dns host “www.b.com” 10.3.210.3

FBL(config)#llb dns host “*.c.com” 10.3.210.1
FBL(config)#llb dns host “*.c.com” 10.3.210.2
FBL(config)#llb dns host “*.c.com” 10.3.210.3
```

→ Step 2 Configure LLB DNS TTL (Time To Live)

```
FBL(config)# llb dns ttl “www.a.com” 60
FBL(config)# llb dns ttl “www.b.com” 60
FBL(config)# llb dns ttl “*.c.com” 60
```

FBL2

→ Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned three IP addresses here.

```
FBL(config)#llb dns host “www.a.com” 10.3.220.1
FBL(config)#llb dns host “www.a.com” 10.3.220.2
FBL(config)#llb dns host “www.a.com” 10.3.220.3

FBL(config)#llb dns host “www.b.com” 10.3.220.1
FBL(config)#llb dns host “www.b.com” 10.3.220.2
FBL(config)#llb dns host “www.b.com” 10.3.220.3

FBL(config)#llb dns host “*.c.com” 10.3.220.1
FBL(config)#llb dns host “*.c.com” 10.3.220.2
FBL(config)#llb dns host “*.c.com” 10.3.220.3
```

→ **Step 2 Configure LLB DNS TTL (Time To Live)**

```
FBL(config)# llb dns ttl “www.a.com” 60
FBL(config)# llb dns ttl “www.b.com” 60
FBL(config)# llb dns ttl “*.c.com” 60
```

FBL3

→ **Step 1 Configure LLB DNS host entry**

Three domain names are configured and each domain name is assigned three IP addresses here.

```
FBL(config)#llb dns host “www.a.com” 10.3.230.1
FBL(config)#llb dns host “www.a.com” 10.3.230.2
FBL(config)#llb dns host “www.a.com” 10.3.230.3

FBL(config)#llb dns host “www.b.com” 10.3.230.1
FBL(config)#llb dns host “www.b.com” 10.3.230.2
FBL(config)#llb dns host “www.b.com” 10.3.230.3

FBL(config)#llb dns host “*.c.com” 10.3.230.1
FBL(config)#llb dns host “*.c.com” 10.3.230.2
FBL(config)#llb dns host “*.c.com” 10.3.230.3
```

→ **Step 2 Configure LLB DNS TTL (Time To Live)**

```
FBL(config)# llb dns ttl “www.a.com” 60
FBL(config)# llb dns ttl “www.b.com” 60
FBL(config)# llb dns ttl “*.c.com” 60
```

13.3.2.3 Configuring Basic SDNS

FBL1

→ **Step 1 Enable SDNS**

```
FBL(config)#sdns on
```

→ **Step 2 Configure SDNS members (their types are “all”)**

```
FBL(config)#sdns member attribute FBL1 10.3.200.1 5888 all
FBL(config)#sdns member attribute FBL2 10.3.200.2 5888 all
FBL(config)#sdns member attribute FBL3 10.3.200.3 5888 all
```

→ **Step 3 Configure FBL1 as a local member**

```
FBL(config)# sdns member local FBL1
```

FBL2

→ **Step 1 Enable SDNS**

```
FBL(config)#sdns on
```

→ Step 2 Configure SDNS members (their types are "all")

```
FBL(config)#sdns member attribute FBL1 10.3.200.1 5888 all
FBL(config)#sdns member attribute FBL2 10.3.200.2 5888 all
FBL(config)#sdns member attribute FBL3 10.3.200.3 5888 all
```

→ Step 3 Configure FBL2 as a local member

```
FBL(config)# sdns member local FBL2
```

FBL3

→ Step 1 Enable SDNS

```
FBL(config)#sdns on
```

→ Step 2 Configure SDNS members (their types are "all")

```
FBL(config)#sdns member attribute FBL1 10.3.200.1 5888 all
FBL(config)#sdns member attribute FBL2 10.3.200.2 5888 all
FBL(config)#sdns member attribute FBL3 10.3.200.3 5888 all
```

→ Step 3 Configure FBL3 as a local member

```
FBL(config)# sdns member local FBL3
```

13.3.2.4 Configuring Host Method

grr

FBL3 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 FBL3

```
FBL(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.) FBL3 is configured as local DNS to resolve "www.a.com".

→ Step 1 Set the weight of "www.a.com" to 1 on FBL1

```
FBL(config)# llb dns host "www.a.com" 10.3.210.1 1
```

→ Step 2 Set the weight of "www.a.com" to 2 on FBL2

```
FBL(config)# llb dns host "www.a.com" 10.3.220.1 2
```

→ Step 3 Set the weight of "www.a.com" to 3 on FBL3

```
FBL(config)# llb dns host "www.a.com" 10.3.230.1 3
```

→ Step 4 Assign "vwgrr" host method to "www.a.com" on FBL3

```
FBL(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 13-2 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

FBL3 is configured as local DNS to resolve a domain name “www.a.com”. Besides the basic configurations, SDNS chain needs to be configured.

FBL3

→ Step 1 Configure an overflow chain called “mychain” on FBL3

```
FBL(config)# sdns overflow chain mychain
```

→ **Step 2 Add FBL1, FBL2, and FBL3 into "mychain"**

```
FBL(config)#sdns overflow member mychain 1 FBL1
FBL(config)#sdns overflow member mychain 2 FBL2
FBL(config)#sdns overflow member mychain 3 FBL3
```

→ **Step 3 Assign "gco" host method to "www.a.com" on FBL3**

```
FBL(config)#sdns host method "www.a.com" gco mychain
```

→ **Step 4 Set the maximum number of TCP connections to 3**

```
FBL(config)# sdns member local FBL3 3
```

FBL1

→ **Step 1 Set the maximum number of TCP connections to 1**

```
FBL(config)# sdns member local FBL1 1
```

FBL2

→ **Step 1 Set the maximum number of TCP connections to 2**

```
FBL(config)# sdns member local FBL2 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 FBL1)
> 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 FBL2 and at the same time maintain the TCP connection to
FBL1)
> 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 FBL1)
Name: www.a.com
Address: 10.3.210.1
```

As is obvious from the above, because the indexes of FBL1, FBL2, and FBL3 are respectively 1, 2, 3, the initial resolving of "www.a.com" will return IP addresses on FBL1. As the number of TCP connection on FBL1 is set to 1, the resolving of "www.a.com" will transfer to FBL2 after maintaining a connection to FBL1. The rest may be deduced by analogy. The resolving of "www.a.com" will transfer to FBL3 after maintaining two connections to FBL2. Once the TCP connection to FBL1 is broken up, the resolving of "www.a.com" will reuse the IP addresses on FBL1.

glc

FBL3 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.
FBL3

→ Step 1 Assign "glc" host method to "www.a.com" on FBL3

```
FBL(config)#sdns host method "www.a.com" glc
```

→ Step 2 Set the maximum number of TCP connections to 3

```
FBL(config)# sdns member local FBL3 3
```

FBL1

→ Step 1 Set the maximum number of TCP connections to 3

```
FBL(config)# sdns member local FBL1 3
```

FBL2

→ Step 1 Set the maximum number of TCP connections to 3

```
FBL(config)# sdns member local FBL2 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 FBL1 is 1, and 2 to FBL2 and FBL3.)
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 FBL2 is 1, and 2 to FBL1 and FBL3.)
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 FBL3 is 1, and 2 to FBL1 and FBL2.)
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

FBL3 is configured as local DNS to resolve a domain name “www.a.com”. Besides the above basic configurations, IP address’s priority should be configured.

FBL3

→ Step 1 Assign “ipo” host method to “www.a.com” on FBL3

```
FBL(config)#sdns host method “www.a.com” ipo
```

→ Step 2 Set “www.a.com” priority to 3

```
FBL(config)# llb dns host “www.a.com” 10.3.230.1 3
```

FBL1

→ Step 1 Set www.a.com priority to 1

```
FBL(config)#llb dns host “www.a.com“ 10.3.210.1 1
```

FBL2

→ Step 1 Set “www.a.com” priority to 2

```
FBL(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.)

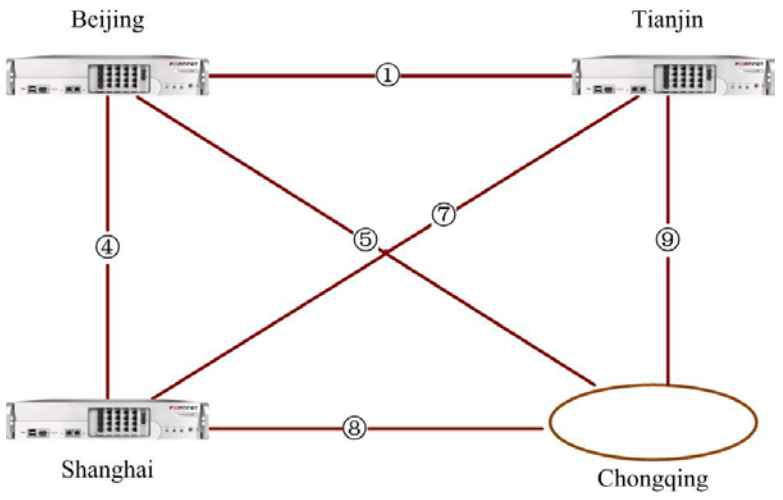


Figure 13-9 Proximity Method

In the above figure, every site has a member, but Chongqing site has no member.

FBL1

→ **Step 1 Configure each site (respectively Beijing, Tianjin, Shanghai and Chongqing)**

```
FBL(config)# sdns site location beijing 42
FBL(config)# sdns site location tianjin 32
FBL(config)# sdns site location shanghai 22
FBL(config)# sdns site location chongqing 12
```

→ **Step 2 Configure the distance value between two sites**

```
FBL(config)# sdns site distance “beijing” “tianjin” 1
FBL(config)# sdns site distance “beijing” “shanghai” 7
FBL(config)# sdns site distance “beijing” “chongqing” 5
FBL(config)# sdns site distance “tianjin” “shanghai” 9
FBL(config)# sdns site distance “tianjin” “chongqing” 5
FBL(config)# sdns site distance “shanghai” “chongqing” 8
```

→ **Step 3 Add the members into sites (Chongqing site has no member)**

```
FBL(config)# sdns site member beijing FBL1
FBL(config)# sdns site member tianjin FBL2
FBL(config)# sdns site member shanghai FBL3
```

→ **Step 4 Configure proximity**

```
FBL(config)# sdns proximity 10.3.50.7 255.255.255.255 beijing 0
FBL(config)# sdns proximity 10.3.200.107 255.255.255.255 tianjin 0
FBL(config)# sdns proximity 10.3.200.108 255.255.255.255 chongqing 0
```

→ **Step 5 Set “www.b.com” method to proximity**

```
FBL(config)# sdns host method “www.b.com” proximity
```

→ **Step 6 Add IP address into “www.b.com”**

```
FBL(config)# slb virtual http “vs4” 10.3.210.4 80
FBL(config)# slb policy static “vs4” “rs1”
FBL(config)# llb dns host “www.b.com” 10.3.210.4
```

FBL2

→ **Step 1 Add IP address into “www.b.com”**

```
FBL(config)# slb virtual http “vs4” 10.3.220.4 80
FBL(config)# slb policy static “vs4” “rs1”
FBL(config)# llb dns host “www.b.com” 10.3.220.4
```

FBL3

→ **Step 1 Add IP address into “www.b.com”**

```
FBL(config)# slb virtual http “vs4” 10.3.230.4 80
FBL(config)# slb policy static “vs4” “rs1”
FBL(config)# llb dns host "www.b.com" 10.3.230.4
```

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 FBL2 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 FBL1 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 FBL1of Beijing site.

13.3.2.5 Configuring Disaster Recovery

Here we assume the host method is configured as “proximity”, then start to configure disaster recovery function.

FBL1

→ Step 1 Create a new host name

```
FBL(config)# llb dns host “www.fortinet.com” 10.3.210.1
```

→ Step 2 Add a disaster recovery group

```
FBL(config)# sdns group dr arr “www.fortinet.com”
```

→ Step 3 Add a primary site

```
FBL(config)# sdns group primary arr Beijing
```

→ Step 4 Add a standby site

```
FBL(config)# sdns group standby arr Tianjin
```

FBL2

→ Step 1 Create a new host name

```
FBL(config)# llb dns host “www.fortinet.com” 10.3.220.1
```

→ Step 2 Add a disaster recovery group

```
FBL(config)# sdns group dr arr “www.fortinet.com”
```

→ Step 3 Add a primary site

```
FBL(config)# sdns group primary arr Beijing
```

→ Step 4 Add a standby site

```
FBL(config)# sdns group standby arr Tianjin
```

“www.fortinet.com” is being resolved through nslookup. The resolving results are as follows:

```
> www.fortinet.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.fortinet.com
Address: 10.3.210.1

> www.fortinet.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.fortinet.com
Address: 10.3.210.1
```

From the above, the IP address of “www.fortinet.com” on the FBL1 will be returned every time, because primary is set to Beijing and the member of Beijing site is FBL1.

Then, set 10.3.210.1 on the FBL1 to DOWN. The result is as follows:

```
> www.fortinet.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.fortinet.com
Address: 10.3.220.1

> www.fortinet.com
Server: [10.3.200.1]
Address: 10.3.200.1

Name: www.fortinet.com
Address: 10.3.220.1
```

At this time, the result is the IP address of "www.fortinet.com" configured on the FBL2, because this is configured on the FBL2 of standby Tianjin site.

13.3.3Sample Configuration for GSLB based on Topology2 via CLI

13.3.3.1 Configuring SLB

SLB configuration is the same for Topology1 and Topology2.

FBL1

→ Step 1 Configure a real server

```
FBL(config)#slb real http "rs1" 10.3.200.110 8080 1000 tcp 1 1
```

→ Step 2 Configure multiple virtual servers

```
FBL(config)#slb virtual http "vs1" 10.3.210.1 80
FBL(config)#slb virtual http "vs2" 10.3.210.2 80
FBL(config)#slb virtual http "vs3" 10.3.210.3 80
```

→ Step 3 Configure SLB policy

```
FBL(config)# slb policy static "vs1" "rs1"
FBL(config)# slb policy static "vs2" "rs1"
FBL(config)# slb policy static "vs3" "rs1"
```

FBL2

→ Step 1 Configure a real server

```
FBL(config)#slb real http "rs1" 10.3.200.110 8080 1000 tcp 1 1
```

→ Step 2 Configure multiple virtual servers

```
FBL(config)#slb virtual http “vs1” 10.3.220.1 80
FBL(config)#slb virtual http “vs2” 10.3.220.2 80
FBL(config)#slb virtual http “vs3” 10.3.220.3 80
```

→ Step 3 Configure SLB policy

```
FBL(config)# slb policy static “vs1” “rs1”
FBL(config)# slb policy static “vs2” “rs1”
FBL(config)# slb policy static “vs3” “rs1”
```

FBL3

→ Step 1 Configure a real server

```
FBL(config)#slb real http “rs1” 10.3.200.110 8080 1000 tcp 1 1
```

→ Step 2 Configure multiple virtual servers

```
FBL(config)#slb virtual http “vs1” 10.3.230.1 80
FBL(config)#slb virtual http “vs2” 10.3.230.2 80
FBL(config)#slb virtual http “vs3” 10.3.230.3 80
```

→ Step 3 Configure SLB policy

```
FBL(config)# slb policy static “vs1” “rs1”
FBL(config)# slb policy static “vs2” “rs1”
FBL(config)# slb policy static “vs3” “rs1”
```

13.3.3.2 Configuring LLB

The “**llb dns host/ttl**” command doesn’t need to be executed for FBL1 on Topology2. LLB configurations for FBL2 and FBL3 are the same on Topology1 and Topology2.

FBL2

→ Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned three IP addresses here.

```
FBL(config)#llb dns host “www.a.com” 10.3.220.1
FBL(config)#llb dns host “www.a.com” 10.3.220.2
FBL(config)#llb dns host “www.a.com” 10.3.220.3

FBL(config)#llb dns host “www.b.com” 10.3.220.1
FBL(config)#llb dns host “www.b.com” 10.3.220.2
FBL(config)#llb dns host “www.b.com” 10.3.220.3

FBL(config)#llb dns host “*.c.com” 10.3.220.1
FBL(config)#llb dns host “*.c.com” 10.3.220.2
FBL(config)#llb dns host “*.c.com” 10.3.220.3
```

→ Step 2 Configure LLB DNS TTL (Time to Live)

```
FBL(config)# llb dns ttl “www.a.com” 60
FBL(config)# llb dns ttl “www.b.com” 60
FBL(config)# llb dns ttl “*.c.com” 60
```

FBL3

→ Step 1 Configure LLB DNS host entry

Three domain names are configured and each domain name is assigned three IP addresses here.

```
FBL(config)#llb dns host “www.a.com” 10.3.230.1
FBL(config)#llb dns host “www.a.com” 10.3.230.2
FBL(config)#llb dns host “www.a.com” 10.3.230.3

FBL(config)#llb dns host “www.b.com” 10.3.230.1
FBL(config)#llb dns host “www.b.com” 10.3.230.2
FBL(config)#llb dns host “www.b.com” 10.3.230.3

FBL(config)#llb dns host “*.c.com” 10.3.230.1
FBL(config)#llb dns host “*.c.com” 10.3.230.2
FBL(config)#llb dns host “*.c.com” 10.3.230.3
```

→ Step 2 Configure LLB DNS TTL (Time to Live)

```
FBL(config)# llb dns ttl “www.a.com” 60
FBL(config)# llb dns ttl “www.b.com” 60
FBL(config)# llb dns ttl “*.c.com” 60
```

13.3.3.3 Configuring Basic SDNS

The basic SDNS configurations on Topology2 are different from these configurations on Topology1. Here, FBL1 needs o be configured as “dns” while FBL2 and FBL3 are configured as “proxy”.

FBL1

→ Step 1 Enable SDNS

```
FBL(config)#sdns on
```

→ Step 2 Configure SDNS members

```
FBL(config)#sdns member attribute FBL1 10.3.200.1 5888 dns
FBL(config)#sdns member attribute FBL2 10.3.200.2 5888 proxy
FBL(config)#sdns member attribute FBL3 10.3.200.3 5888 proxy
```

→ Step 3 Configure FBL1 as a local member

```
FBL(config)# sdns member local FBL1
```

FBL2

→ Step 1 Enable SDNS

```
FBL(config)#sdns on
```

→ Step 2 Configure SDNS members

```
FBL(config)#sdns member attribute FBL1 10.3.200.1 5888 dns
FBL(config)#sdns member attribute FBL2 10.3.200.2 5888 proxy
FBL(config)#sdns member attribute FBL3 10.3.200.3 5888 proxy
```

→ Step 3 Configure FBL2 as a local member

```
FBL(config)# sdns member local FBL2
```

FBL3

→ Step 1 Enable SDNS

```
FBL(config)#sdns on
```

→ Step 2 Configure SDNS members

```
FBL(config)#sdns member attribute FBL1 10.3.200.1 5888 dns
FBL(config)#sdns member attribute FBL2 10.3.200.2 5888 proxy
FBL(config)#sdns member attribute FBL3 10.3.200.3 5888 proxy
```

→ Step 3 Configure FBL3 as a local member

```
FBL(config)# sdns member local FBL3
```

13.3.3.4 Configuring Host Method

region

The logical architecture related to SDNS site/region/pool in this example should be introduced firstly.

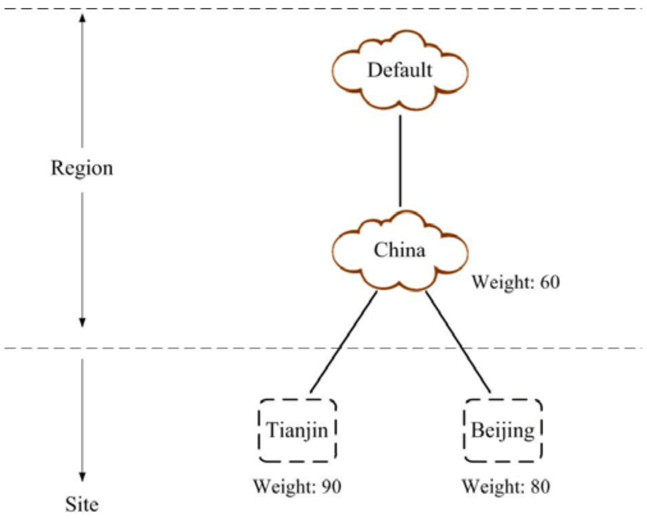


Figure 13-10 SDNS Region

In Configuring Basic SDNS section, FBL1 needs to be configured as “dns”, while FBL2 and FBL3 need to be configured as “proxy”.

FBL1

→ **Step 1 Create two sites: Beijing and Tianjin**

```
FBL(config)# sdns site location beijing 90
FBL(config)# sdns site location tianjin 80
```

→ **Step 2 Add the members into the sites**

```
FBL(config)# sdns site member beijing FBL2
FBL(config)# sdns site member tianjin FBL3
```

→ **Step 3 Create two regions: China and Default**

```
FBL(config)# sdns region location china 60
FBL(config)# sdns region location default 30
```

→ **Step 4 Add the region/site into the region**

```
FBL(config)# sdns region division china beijing
FBL(config)# sdns region division china Tianjin
FBL(config)# sdns region division default china
```

→ **Step 5 Create a pool (www.b.com-beijing) and configure its IP addresses**

```
FBL(config)# sdns pool method “www.b.com” beijing rr 2
FBL(config)# sdns pool ip “www.b.com” beijing 10.3.220.1 5
FBL(config)# sdns pool ip “www.b.com” beijing 10.3.220.2 5
FBL(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**

```
FBL(config)# sdns pool method “www.b.com” tianjin ipo 1
FBL(config)# sdns pool ip “www.b.com” tianjin 10.3.230.1 6
FBL(config)# sdns pool ip “www.b.com” tianjin 10.3.230.2 5
FBL(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**

```
FBL(config)# sdns pool method “www.b.com” china pi 2
FBL(config)# sdns pool ip “www.b.com” china 10.3.220.1 3
FBL(config)# sdns pool ip “www.b.com” china 10.3.230.1 3
FBL(config)# sdns persistence timeout 12
```

→ **Step 8 Create a pool (www.b.com-default) and configure its IP addresses**

```
FBL(config)# sdns pool method “www.b.com” default rr 2
FBL(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**

```
FBL(config)# sdns pool rule “rule1” china rr 3
FBL(config)# sdns pool ip “rule1” china 10.3.220.1 10
```

→ Step 10 Set the rule1 to host www.a.com

```
FBL(config)# sdns host rule “rule1” www.a.com
```

→ Step 11 Set the host method to “region”

```
FBL(config)# sdns host method “www.a.com” region
FBL(config)# sdns host method “www.b.com” region
```

→ Step 12 Set the SDNS proximity

```
FBL(config)# sdns proximity 10.3.200.107 255.255.255.255 beijing
FBL(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’s 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’s number of the pool is assigned to 1, every time the IP address with the highest priority will be returned.

13.3.3.5 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

```
FBL(config)# sdns bandwidth region china 1 10
```

→ Step 2 Set the “beijing site” bandwidth limit to 2M, and the statistics mode is inout

```
FBL(config)# sdns bandwidth site beijing 3 2
```

→ Step 3 Set the “tianjin site” bandwidth limit to 1M, and the statistics mode is in

```
FBL(config)#sdns bandwidth site tianjin 2 1
```

→ Step 4 Set the FBL1 member bandwidth limit to 1M, and the statistics mode is inout

```
FBL(config)# sdns bandwidth member FBL2 1 1
```

→ Step 5 Set the FBL2 member bandwidth limit to 1M, and the statistics mode is inout

```
FBL(config)# sdns bandwidth member FBL3 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:

```
FBL1(config)#show sdns band
Name          Site/Region ID         Limit         Usage          Mode          Status
china                3               10000000      1231638  1

Region: china
www.a.com            3               10000000      1254880  8

Region: default

beijing              1               2000000  615906        3

Site: beijing
www.b.com            1               5000000  0             7

tianjin              2               1000000  666           2

Site: tianjin

FBL3                                 1000000  901           1
FBL2                                 1000000  1230737  1              Full
FBL1                                 -1            0              0

The bandwidth of vips:

The bandwidth of hosts:
www.b.com            0               1000000  1254880  6              Full
```

Set the bandwidth of host name, host name of region and host name of site

→ Step 1 Set the bandwidth limit of the host name "www.b.com" to 1M, and the mode is request and response

```
FBL(config)# sdns bandwidth host "www.b.com" 6 1
```

→ Step 2 Set the bandwidth limit of the host name "www.b.com" in site "beijing" to 5M, and the mode is request

```
FBL(config)# sdns bandwidth host "www.b.com" 7 5 beijing
```

→ Step 3 Set the bandwidth limit of the host name "www.b.com" in site "beijing" to 10M, and the mode is request

```
FBL(config)# sdns bandwidth host www.b.com 8 10 china
```

Access “www.b.com” from 10.3.200.107 (DNS server is set to 10.3.200.1). The traffic is displayed as follows:

```
FBL1(config)#show sdns band
Name          Site/Region ID          Limit         Usage         Mode          Status
china                3                10000000      1231638  1

Region: china
www.a.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

FBL3                                  1000000  901           1
FBL2                                  1000000  1230737  1               Full
FBL1                                  -1            0             0

The bandwidth of vips:

The bandwidth of hosts:
www.b.com            0                1000000  1254880  6               Full
```

13.3.3.6 Configuring SDNS Alias

Here we assume that the domain name users will access is “www.e.com”. But after passing some DNS-related servers, the domain name is transferred to “www.fortinet.e.com”. FortiBalancer appliance must configure “www.fortinet.e.com”, but the traffic of users accessing “www.e.com” need to be recorded as the traffic of “www.fortinet.e.com”.

FBL1

→ Step 1 Configure the bandwidth of “www.fortinet.e.com”

```
FBL(config)# sdns bandwidth host “www.fortinet.e.com” 6
```

FBL2

→ Step 1 Add "www.fortinet.e.com" host name

```
FBL(config)# llb dns host "www.fortinet.e.com" 10.3.220.1
```

→ Step 2 Set "www.fortinet.e.com" as the alias of "www.e.com"

```
FBL(config)# sdns alias "www.e.com" "www.fortinet.e.com"
```

Access "www.e.com" by a certain method and download a large packet. The result is displayed on FBL1 as follows:

The bandwidth of hosts:

```
ww.b.com              0       1000000     0           6           Full
www.fortinet.e.com    0       1000000     1610372     6           Full
```

As is obvious from the above, the statistics of the traffic of "www.fortinet.e.com" has been collected.

13.3.4Sample Configuration for GSLB DPS via CLI

We should configure a DPS master and a DPS slave.

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.

FBL1 (DPS master)

→ Step 1 Basic SDNS configuration

```
FBL(config)# sdns on Check
FBL(config)# sdns interval heartbeat 2
FBL(config)# sdns site location beijing 0
FBL(config)# sdns site location shanghai 0
FBL(config)# sdns interval report 30
```

→ Step 2 SDNS DPS master configuration

```
FBL(config)# sdns dps interval send 15
FBL(config)# sdns dps interval query 15
FBL(config)# sdns dps history 9000
FBL(config)# sdns dps method hops
FBL(config)# sdns dps detector beijing 10.3.17.19 44544 15
FBL(config)# sdns dps detector shanghai 172.16.63.204 44544 15
FBL(config)# sdns dps member 10.3.17.100
FBL(config)# sdns dps member 10.3.17.20
FBL(config)# sdns dps on
FBL(config)# sdns dps master on 55456
FBL(config)# sdns statistics on localdns
```

FBL2 (DPS slave)

→ Step 1 Basic SDNS configuration

```
FBL(config)# sdns on Check
FBL(config)# sdns interval heartbeat 2
FBL(config)# sdns site location beijing 0
FBL(config)# sdns site location shanghai 0
FBL(config)# sdns interval report 30
```

→ Step 2 SDNS DPS slave configuration

```
FBL(config)# sdns dps interval send 15
FBL(config)# sdns dps interval query 15
FBL(config)# sdns dps history 9000
FBL(config)# sdns dps method rtt
FBL(config)# sdns dps detector beijing 10.3.17.19 44544 15
FBL(config)# sdns dps detector shanghai 172.16.63.204 44544 15
FBL(config)# sdns dps on
FBL(config)# sdns dps master off
```

Chapter 14 Logging

14.1 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 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.

14.2 Understanding Logging

14.2.1Syslog

Syslog is a protocol that is used for the transmission of event notification message across networks.

Fortinet 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.

14.2.2HTTP Access Logging

HTTP Access Logging is the logging of information about every HTTP request and its response in a specific predefined format.

Fortinet 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.

14.2.3Log 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 FBLOS 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.

14.3 Logging Configuration

14.3.1Configuration Guidelines

Table 14-1 General Settings of Logging

Operation	Command
Enable the logging	**log {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>*

14.3.2Sample Configuration for Logging via CLI

→ Step 1 Enable FortiBalancer logging function

The logging system is on by default.

```
FBL(config)#log on
```

→ Step 2 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.

```
FBL(config)#log host 10.2.37.1 514 udp 1
```

→Step 3 Set log filters for the configured host

No more than 3 log filters can be set on one syslog host. Log filter can’t 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.

```
FBL(config)#log filter 1 1 “index”
```

→ Step 4 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*.

```
FBL(config)#log level err
```

→ Step 5 Change the syslog facility

The default facility is *LOCAL0*.

```
FBL(config)#log facility LOCAL0
```

→ Step 6 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.

```
FBL(config)#log http squid
```

Chapter 15 Administrative Tools

15.1 Overview

This final chapter of the User Guide will focus on various configuration maintenance elements, such as downloading new FBLOS software, rebooting your FortiBalancer appliance, reverting your configuration to a previously saved status or returning the FortiBalancer 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 WebUI) 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.

15.2 Administrative Tools Configuration

15.2.1Configuration Guidelines

Table 15-1 General Settings for Administrative Tools

Operation	Command
Configuring External Authentication	**admin aaa {on|off}** **admin aaa method** *[radius|tac_x]* **admin aaa server** *{es01|es02} <hostname_ip> <port> <secret>*
System shutdown and reboot	**system shutdown** **system reboot**
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>*

Operation	Command
	config file *<file_name>*
Software upgrade	**system update** *<url>*
Configuration Synchronization	**synconfig peer** *<name> <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]* **show ntp** **clear ntp**
XML RPC	**xmlrpc {on|off}** *[https|http]* **xmlrpc port** *<port>* **show xmlrpc** **clear xmlrpc**

15.2.2Sample Configuration for Administrative Tools via CLI

15.2.2.1 Configuring External Authentication

If you have an external authentication server (RADIUS/Tacacs), you may use these servers to authenticate the SSH/WebUI 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.

```
FBL(config)#admin aaa on
FBL(config)#admin aaa method RADIUS
FBL(config)#admin aaa server es01 “10.1.1.1” 1812 radiussecret
FBL(config)#admin aaa server es02 radius_host 1812 radiussecret
```

15.2.2.2 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.

```
FBL(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, 15-second period, users may turn off the appliance.

In some cases when dealing with configuration changes you might need to reboot the box.

```
FBL(config)#system reboot
```

15.2.2.3 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 FBLOS 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:

```
FBL(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:

```
FBL(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:

```
FBL(config)#config memory
FBL(config)#config file new_lb
FBL(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.

15.2.2.4 Software Upgrade Procedure

To see the current version of FBLOS™ software that is running, we use the "**show version**" command.

```
FBL(config)#show version

FBLOS Beta.FBL.8.1.1.4 build on Wed Feb 23 15:05:32 2011

Host name          : FBL
System CPU         : Intel(R) Xeon(R) CPU      X5670   @ 2.93GHz
System Module      : X8DTH-i
System RAM         : 8063480 kbytes.
System boot time   : Fri Feb 25 06:04:17 GMT (+0000) 2011
Current time       : Fri Feb 25 08:34:52 GMT (+0000) 2011
System up time     : 2:31
Platform Bld Date  : Wed Feb 23 15:05:22 UTC 2011
SSL HW             : HW ( 2X8D ) Initialized
Compression HW     : No HW Available
Power supply       : 2U, AC, 2-cords, Redundancy
Network Interface  : 10 x Gigabit Ethernet copper
Model              : FortiBalancer 3000
Serial Number      : 0437A3345200010002262002502162
Licensed Features  : Clustering   L4SLB   L7SLB   Caching   SSL
                     tProxy   AppGateway   SwCompression   LLB   GSLB   QoS
                     DynRoute   FFO
License Key        : 469b9404-135a3d73-b1288158-aa6f25ec-e96bb2ce-00000000-00c5d9aa-99999999

FORTINET Customer Support
Telephone          : please go to URL support.fortinet.com
Email              : please go to URL support.fortinet.com
Update             : please contact support for instructions
Website            : http://www.fortinet.com

Other Root Version
Beta.FBL.8.1.1.3 build on Tues Feb 22 15:46:00 2011
```

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 by: https://support.fortinet.com/

Once you have received a password and verified with a customer support engineer that FBLOS 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 FBLOS. 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:

```
FBL(config)#system update http://192.168.10.10/Rel_FBL_8_1_1_4.click
This will upgrade your system from http://192.168.10.10/Rel_FBL_8_1_1_4.click
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 FBLOS 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 (please go to: https://support.fortinet.com/) to obtain a new license key.

15.2.2.5 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 Machine1

```
AN1(config)#synconfig peer machine1 192.168.1.1
AN1(config)#synconfig to machine2
```

→ Step 2 Configure configuration synchronization on Machine2

```
AN2(config)#synconfig peer machine1 192.168.1.1
AN2(config)#synconfig peer machine2 192.168.1.2
AN2(config)#synconfig from machine1
```

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 Fortinet machines (FortiBalancer appliance and the sync peer).

15.2.2.6 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 FBL1 are synchronized to remote FBL2.

→ Step 1 Configure SDNS configuration synchronization on FBL1

```
AN1(config)#synconfig sdns peer peerlocal 172.16.83.180
AN1(config)#synconfig sdns peer peerremote 172.16.83.120
```

→ Step 2 Start SDNS configuration synchronization from FBL1 to FBL2

```
AN1(config)#synconfig sdns to peerremote
```

15.2.2.7 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

```
FBL(config)# graph name aa
FBL(config)# graph rename aa bb
FBL(config)# graph settings displaymode stack bb
FBL(config)# graph item bb "System" "CPU Utilization" "1" "red" "2"
```

15.2.2.8 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 FBLOS updates, but they are much smaller in size.

15.2.2.9 NTP Time Synchronizer

NTP (Network Time Protocol) time synchronizer keeps the system time in synchronism with the specified NTP server.

After NTP time synchronizer is enabled, the system time will be automatically synchronized with the NTP server for every approximate 15 minutes.

Attention! For the first time the offset between the system time and the NTP server exceeds 1000s (approximately 17 mins), the system clock will be synchronized with the NTP server. However, for the second time the limit is exceeded, NTP time synchronizer will exit and the system clock will not be synchronized with the NTP server any more. That is to say, if NTP time synchronizer is enabled, users should not set the system clock manually; otherwise, NTP time synchronizer doesn't work.

If multiple NTP servers are configured, FortiBalancer appliance will calculate the round-trip delay according to the timing information in the response from each NTP server, and synchronize its system time with the one with the minimum delay.

→ Step 1 Configure an NTP server

```
AN1(config)#ntp server 207.46.197.32 4
```

→ Step 2 Turn on NTP time synchronizer

```
AN1(config)#ntp on
```

Users also can use the command "**show ntp**" to view the current NTP configuration.

```
AN1(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:

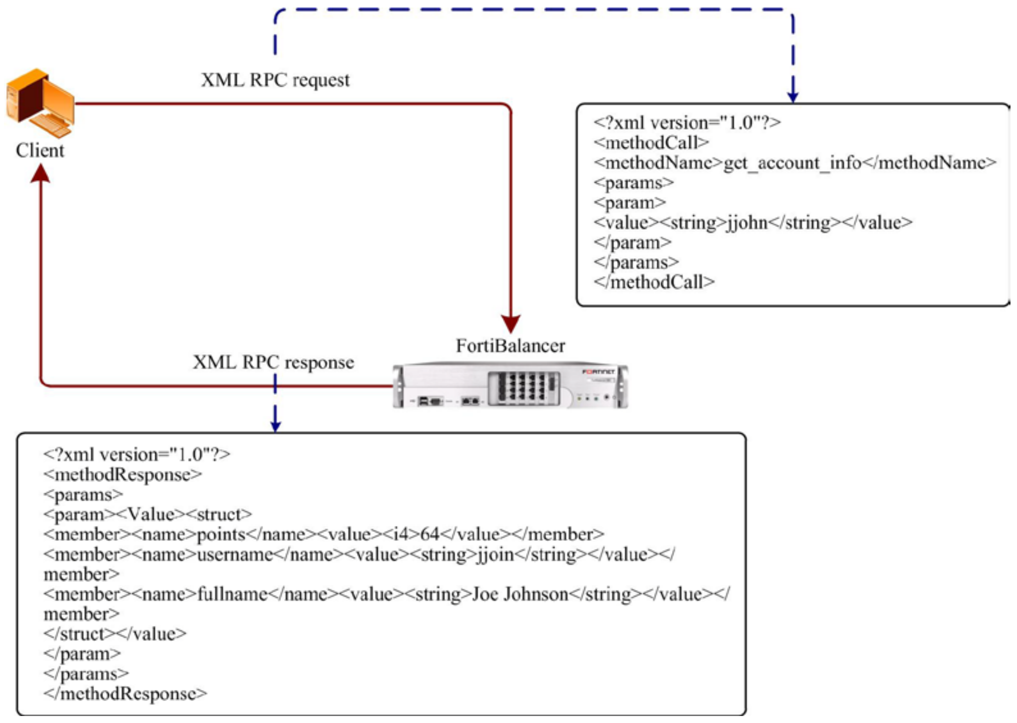
Time since restart:	The time in hours since the system was last rebooted.
Time since reset:	The time since the statistics were reset and the system statistics monitoring file was updated. This is designed for busy servers, such as those operated by NIST, USNO, and intended as early warning detector of 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.

15.2.2.10 XML RPC

XML RPC allows clients to run some CLI commands remotely in FBLOS. This enables system programmers to automate remote configuration which is difficult with WebUI.

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 following figure, Client sends an HTTP POST Request to Fortinet FortiBalancer. XML RPC message is the body of the HTTP Request, in which the commands to run and the commands' parameters are specified. Then, Fortinet 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 15-1 XML RPC Working Mechanism

To realize the communication between Client and FortiBalancer, a Perl script MUST be first executed on Client. The command executed the script is:

```
array_xmlrpc.pl –d <address> -p <port> -f <data_file>
```

In this command, <address> specifies the Fortinet 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>test</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][hctype][hcup][hcdown]". XML PRC method is embedded in a <methodName> tag (Please refer to Appendix III, in which all XML RPC methods supported by Fortinet 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, Fortinet 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 Fortinet FortiBalancer, you need to configure two commands:

→ Step 1 Turn on XML RPC

```
AN1(config)#xml on https
```

→ Step 2 Set the port for XML RPC to listen

```
AN1(config)#xml port 9999
```

15.2.2.11 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 FortiBalancer 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 FortiBalancer 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 WebUI by clicking the "Flight Deck" node at the bottom of the WebUI 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 WebUI from the “Graph→SNMP Monitoring” page under the “Admin Tools” node. Some of the statistics also require additional configuration, which will be described below.

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.

Appendix I SNMP OID List

SNMP OID List		
1	.1.3.6.1.4.1.7564	This file defines the private CA SNMP MIB extensions.
2	.1.3.6.1.4.1.7564.4.1.0	Current system total available memory.
3	.1.3.6.1.4.1.7564.16.1.1.0	Current status of the reverse proxy cache - on or off.
4	.1.3.6.1.4.1.7564.16.1.2.0	Total number of requests received by the reverse proxy cache.
5	.1.3.6.1.4.1.7564.16.1.3.0	Total GET requests received by the reverse proxy cache.
6	.1.3.6.1.4.1.7564.16.1.4.0	Total HEAD requests received by the reverse proxy cache.
7	.1.3.6.1.4.1.7564.16.1.5.0	Total PURGE requests received by the reverse proxy cache.
8	.1.3.6.1.4.1.7564.16.1.6.0	Total POST requests received by the reverse proxy cache.
9	.1.3.6.1.4.1.7564.16.1.7.0	Number of current client connections (e.g. from the browsers).
10	.1.3.6.1.4.1.7564.16.1.8.0	Number of current backend server connections.
11	.1.3.6.1.4.1.7564.16.1.9.0	Requests redirected to HTTPS.
12	.1.3.6.1.4.1.7564.16.1.10.0	Requests redirected based on regex match.
13	.1.3.6.1.4.1.7564.16.1.11.0	Requests forwarded with rewritten url.
14	.1.3.6.1.4.1.7564.16.1.12.0	Locations rewritten to HTTPS.
15	.1.3.6.1.4.1.7564.16.1.13.0	Locations rewritten based on regex match.
16	.1.3.6.1.4.1.7564.16.1.14.0	Cache skip, cache off.
17	.1.3.6.1.4.1.7564.16.1.15.0	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.
18	.1.3.6.1.4.1.7564.16.1.16.0	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.
19	.1.3.6.1.4.1.7564.16.1.17.0	Cache hit, reply with Precondition Failed.
20	.1.3.6.1.4.1.7564.16.1.18.0	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).
21	.1.3.6.1.4.1.7564.16.1.19.0	The request does not result in a cache table search. Something in the request made us deem it

SNMP OID List		
		non-cacheable (e.g. very long URL, a 'Cache-Control: no-store' header etc).
22	.1.3.6.1.4.1.7564.16.1.20.0	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).
23	.1.3.6.1.4.1.7564.16.1.21.0	Cache miss, create new entry, resp noncacheable.
24	.1.3.6.1.4.1.7564.16.1.22.0	Cache hit reply using cache + cache reply with 'not modified'.
25	.1.3.6.1.4.1.7564.18.1.1.0	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.
26	.1.3.6.1.4.1.7564.18.1.2.0	Current number of entries in the vrrpTable.
27	.1.3.6.1.4.1.7564.18.1.3	A table containing clustering configuration.
28	.1.3.6.1.4.1.7564.18.1.3.1	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.
29	.1.3.6.1.4.1.7564.18.1.3.1.1	The cluster virtual table index.
30	.1.3.6.1.4.1.7564.18.1.3.1.2	The cluster identifier.
31	.1.3.6.1.4.1.7564.18.1.3.1.3	The current state of the cluster.
32	.1.3.6.1.4.1.7564.18.1.3.1.4	The interface name on which the cluster is defined.
33	.1.3.6.1.4.1.7564.18.1.3.1.5	A virtual ip address (VIP) in the cluster.
34	.1.3.6.1.4.1.7564.18.1.3.1.6	Type of authentication being used. none(0) - no authentication simple-text-password(1) - use password specified in cluster virtual for authentication.
35	.1.3.6.1.4.1.7564.18.1.3.1.7	The password for authentication.
36	.1.3.6.1.4.1.7564.18.1.3.1.8	This is for controlling whether a higher priority Backup VRRP virtual preempts a low priority Master.
37	.1.3.6.1.4.1.7564.18.1.3.1.9	VRRP advertisement interval.
38	.1.3.6.1.4.1.7564.18.1.3.1.10	Priority of the local node in the cluster.
39	.1.3.6.1.4.1.7564.19.1.1.1.0	Number of real services currently configured.
40	.1.3.6.1.4.1.7564.19.1.1.2	A table containing the configuration of real

SNMP OID List		
		services.
41	.1.3.6.1.4.1.7564.19.1.1.2.1	An rsTable entry containing the information of one real service.
42	.1.3.6.1.4.1.7564.19.1.1.2.1.1	The reference index for each real service.
43	.1.3.6.1.4.1.7564.19.1.1.2.1.2	The name of the real service.
44	.1.3.6.1.4.1.7564.19.1.1.2.1.3	The protocol of the real service.
45	.1.3.6.1.4.1.7564.19.1.1.2.1.4	The real service IP address.
46	.1.3.6.1.4.1.7564.19.1.1.2.1.5	The port number of the real service.
47	.1.3.6.1.4.1.7564.19.1.1.2.1.6	Maximum number of connections per real service.
48	.1.3.6.1.4.1.7564.19.1.1.2.1.8	The current status of real service - up or down.
49	.1.3.6.1.4.1.7564.19.1.1.2.1.9	Server Average Response Time (in microseconds).
50	.1.3.6.1.4.1.7564.19.1.2.1.0	Number of virtual services currently configured.
51	.1.3.6.1.4.1.7564.19.1.2.2	A table containing the configuration of virtual services.
52	.1.3.6.1.4.1.7564.19.1.2.2.1	A vsTable entry containing the configuration of one virtual service.
53	.1.3.6.1.4.1.7564.19.1.2.2.1.1	Reference index for each virtual service.
54	.1.3.6.1.4.1.7564.19.1.2.2.1.2	Name of the virtual service.
55	.1.3.6.1.4.1.7564.19.1.2.2.1.3	The protocol of the virtual service.
56	.1.3.6.1.4.1.7564.19.1.2.2.1.4	The virtual service IP address.
57	.1.3.6.1.4.1.7564.19.1.2.2.1.5	The port of the virtual service.
58	.1.3.6.1.4.1.7564.19.1.2.2.1.6	The max connection of virtual service.
59	.1.3.6.1.4.1.7564.19.1.3.1	Number of groups currently configured.
60	.1.3.6.1.4.1.7564.19.1.3.2	A table containing group members' configurations.
61	.1.3.6.1.4.1.7564.19.1.3.2.1	A gpTable entry containing one group member's configuration.
62	.1.3.6.1.4.1.7564.19.1.3.2.1.1	Reference index for each group member.
63	.1.3.6.1.4.1.7564.19.1.3.2.1.2	Name of the group.
64	.1.3.6.1.4.1.7564.19.1.3.2.1.3	Name of the real service.
65	.1.3.6.1.4.1.7564.19.1.3.2.1.4	Metric used to balance real services within the group.
66	.1.3.6.1.4.1.7564.19.2.1.1	Real service statistics table.
67	.1.3.6.1.4.1.7564.19.2.1.1.1	An rsStatsTable entry containing the statistics of one real service.
68	.1.3.6.1.4.1.7564.19.2.1.1.1.1	Reference index for each real service.
69	.1.3.6.1.4.1.7564.19.2.1.1.1.2	Name of the real service.
70	.1.3.6.1.4.1.7564.19.2.1.1.1.3	Real service IP address.
71	.1.3.6.1.4.1.7564.19.2.1.1.1.4	The port number of the real service.
72	.1.3.6.1.4.1.7564.19.2.1.1.1.5	Number of outstanding requests to the real service.

SNMP OID List		
73	.1.3.6.1.4.1.7564.19.2.1.1.1.6	Number of open connections to the real service.
74	.1.3.6.1.4.1.7564.19.2.1.1.1.7	The total number of requests sent to the real service.
75	.1.3.6.1.4.1.7564.19.2.1.1.1.8	The health status (up or down) of the real service.
76	.1.3.6.1.4.1.7564.19.2.2.1	A statistics table for virtual service.
77	.1.3.6.1.4.1.7564.19.2.2.1.1	A vsStatsTable entry containing the statistics of one virtual service.
78	.1.3.6.1.4.1.7564.19.2.2.1.1.1	Reference index for each virtual service.
79	.1.3.6.1.4.1.7564.19.2.2.1.1.2	Name of the virtual service.
80	.1.3.6.1.4.1.7564.19.2.2.1.1.3	IP address of the virtual service.
81	.1.3.6.1.4.1.7564.19.2.2.1.1.4	Port number of the virtual service.
82	.1.3.6.1.4.1.7564.19.2.2.1.1.5	Number of QoS URL policy hits for the virtual service.
83	.1.3.6.1.4.1.7564.19.2.2.1.1.6	Number of QoS Hostname policy hits for the virtual service.
84	.1.3.6.1.4.1.7564.19.2.2.1.1.7	Number of Persistent Cookie policy hits for the virtual service.
85	.1.3.6.1.4.1.7564.19.2.2.1.1.8	Number of QoS Cookie hits for the virtual service.
86	.1.3.6.1.4.1.7564.19.2.2.1.1.9	Number of Default policy hits for the virtual service.
87	.1.3.6.1.4.1.7564.19.2.2.1.1.10	Number of Persistent URL policy hits for the virtual service.
88	.1.3.6.1.4.1.7564.19.2.2.1.1.11	Number of Static policy hits for the virtual service.
89	.1.3.6.1.4.1.7564.19.2.2.1.1.12	Number of QoS Network policy hits for the virtual service.
90	.1.3.6.1.4.1.7564.19.2.2.1.1.13	Number of QoS URL policy hits for the virtual service.
91	.1.3.6.1.4.1.7564.19.2.2.1.1.14	Number of Backup policy hits for the virtual service.
92	.1.3.6.1.4.1.7564.19.2.2.1.1.15	Number of Cache hits for the virtual service.
93	.1.3.6.1.4.1.7564.19.2.2.1.1.16	Number of Regex policy hits for the virtual service.
94	.1.3.6.1.4.1.7564.19.2.2.1.1.17	Number of Rewrite Cookie policy hits for the virtual service.
95	.1.3.6.1.4.1.7564.19.2.2.1.1.18	Number of Insert Cookie policy hits for the virtual service.
96	.1.3.6.1.4.1.7564.19.2.3.1	A statistics table of the group.
97	.1.3.6.1.4.1.7564.19.2.3.1.1	A gpStatsTable entry containing the statistics of one group.
98	.1.3.6.1.4.1.7564.19.2.3.1.1.1	Reference index for each group.
99	.1.3.6.1.4.1.7564.19.2.3.1.1.2	Name of the group.

SNMP OID List		
100	.1.3.6.1.4.1.7564.19.2.3.1.1.3	Total hits for the group.
101	.1.3.6.1.4.1.7564.20.1.2.0	Number of vhosts currently configured.
102	.1.3.6.1.4.1.7564.20.2.1.0	Total number of open SSL connections (all vhosts).
103	.1.3.6.1.4.1.7564.20.2.2.0	Total number of accepted SSL connections (all vhosts).
104	.1.3.6.1.4.1.7564.20.2.3.0	Total number of requested SSL connections (all vhosts).
105	.1.3.6.1.4.1.7564.20.2.4	SSL vhost statistics table.
106	.1.3.6.1.4.1.7564.20.2.4.1	sslTable entry for one vhost.
107	.1.3.6.1.4.1.7564.20.2.4.1.1	The SSL table index.
108	.1.3.6.1.4.1.7564.20.2.4.1.2	Name of the SSL vhost.
109	.1.3.6.1.4.1.7564.20.2.4.1.3	Open SSL connections for vhostName.
110	.1.3.6.1.4.1.7564.20.2.4.1.4	Number of accepted SSL connections for vhostName.
111	.1.3.6.1.4.1.7564.20.2.4.1.5	Number of requested SSL connections for vhostName.
112	.1.3.6.1.4.1.7564.20.2.4.1.6	Number of resumed SSL sessions for vhostName"
113	.1.3.6.1.4.1.7564.20.2.4.1.7	Number of resumable SSL sessions for vhostName.
114	.1.3.6.1.4.1.7564.20.2.4.1.8	Number of session misses for vhostName.
115	.1.3.6.1.4.1.7564.22.1.0	Status of VIP statistics gathering - on or off.
116	.1.3.6.1.4.1.7564.22.2.0	The hostname that the VIP is representing (hostname of the appliance).
117	.1.3.6.1.4.1.7564.22.3.0	The current time in the format of MM/DD/YY HH:MM.
118	.1.3.6.1.4.1.7564.22.4.0	Total number of ip packets received on all VIPs.
119	.1.3.6.1.4.1.7564.22.5.0	Total number of ip packets sent out on all VIPs.
120	.1.3.6.1.4.1.7564.22.6.0	Total number of IP bytes received on all VIPs.
121	.1.3.6.1.4.1.7564.22.7.0	Total number of IP bytes sent out on all VIPs.
122	.1.3.6.1.4.1.7564.22.8	A table of VIP statistics.
123	.1.3.6.1.4.1.7564.22.8.1	An entry in the ipStatsTable which is created for each VIP.
124	.1.3.6.1.4.1.7564.22.8.1.1	The VIP statistics table index.
125	.1.3.6.1.4.1.7564.22.8.1.2	The VIP address.
126	.1.3.6.1.4.1.7564.22.8.1.3	Total number of IP packets received on the VIP.
127	.1.3.6.1.4.1.7564.22.8.1.4	Total number of bytes received on the VIP.
128	.1.3.6.1.4.1.7564.22.8.1.5	Total number of packets sent out on the VIP.
129	.1.3.6.1.4.1.7564.22.8.1.6	Total number of bytes sent out on the VIP.
130	.1.3.6.1.4.1.7564.22.8.1.7	The time statistics gathering was enabled for the VIP.
131	.1.3.6.1.4.1.7564.23.1.0	The number of network interfaces presented on

SNMP OID List		
		this system.
132	.1.3.6.1.4.1.7564.23.2.0	The total accumulated number of octets received on all the active interfaces (loopback is not included).
133	.1.3.6.1.4.1.7564.23.3.0	The total accumulated number of octets transmitted out on all the active interfaces (loopback is not included).
134	.1.3.6.1.4.1.7564.23.4	A table of interface statistics. The number of entries is given by the value of infNumber.
135	.1.3.6.1.4.1.7564.23.4.1	An infTable entry for one interface.
136	.1.3.6.1.4.1.7564.23.4.1.1	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.
137	.1.3.6.1.4.1.7564.23.4.1.2	Name of the interface.
138	.1.3.6.1.4.1.7564.23.4.1.3	The current operational state of the interface (up or down).
139	.1.3.6.1.4.1.7564.23.4.1.4	The interface's IP address.
140	.1.3.6.1.4.1.7564.23.4.1.5	The total number of octets received on the interface, including framing characters.
141	.1.3.6.1.4.1.7564.23.4.1.6	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.
142	.1.3.6.1.4.1.7564.23.4.1.7	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.
143	.1.3.6.1.4.1.7564.23.4.1.8	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.
144	.1.3.6.1.4.1.7564.23.4.1.9	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.
145	.1.3.6.1.4.1.7564.23.4.1.10	For packet-oriented interfaces, the number of packets received via the interface which were

SNMP OID List		
		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.
146	.1.3.6.1.4.1.7564.23.4.1.11	The total number of octets transmitted out of the interface, including framing characters.
147	.1.3.6.1.4.1.7564.23.4.1.12	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.
148	.1.3.6.1.4.1.7564.23.4.1.13	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.
149	.1.3.6.1.4.1.7564.23.4.1.14	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.
150	.1.3.6.1.4.1.7564.24.1.1.0	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.
151	.1.3.6.1.4.1.7564.24.1.2.0	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.
152	.1.3.6.1.4.1.7564.24.1.3.0	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

SNMP OID List		
		their severity decreases, e.g. error(4) is more severe than debug(8).
153	.1.3.6.1.4.1.7564.24.2.1.0	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.
154	.1.3.6.1.4.1.7564.24.2.2	A table of syslog messages generated by this device. All 'interesting' syslog messages (i.e. severity <= logMaxSeverity) are entered into this table.
155	.1.3.6.1.4.1.7564.24.2.2.1	A syslog message that was previously generated by this device. Each entry is indexed by a message index.
156	.1.3.6.1.4.1.7564.24.2.2.1.1	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.
157	.1.3.6.1.4.1.7564.24.2.2.1.2	The severity of the message.
158	.1.3.6.1.4.1.7564.24.2.2.1.3	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.
159	.1.3.6.1.4.1.7564.24.3.1	If the fastlog level is larger than or equal to error （include emerg, alert, crit, err）， this trap will be sent. Some examples below: 1. Excute "ifconfig em0 up/down" 2. One of dual power supply failed/recovered 3. Excute "log test"" 4. CPU fan failed/recovered 5. CPU overheat
160	.1.3.6.1.4.1.7564.25.1.0	The number of times ClickTCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.
161	.1.3.6.1.4.1.7564.25.2.0	The number of times ClickTCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.
162	.1.3.6.1.4.1.7564.25.3.0	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 connections have made a direct transition to the LISTEN state

SNMP OID List		
		from the SYN-RCVD state.
163	.1.3.6.1.4.1.7564.25.4.0	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.
164	.1.3.6.1.4.1.7564.25.5.0	The number of ClickTCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.
165	.1.3.6.1.4.1.7564.25.6.0	The total number of ClickTCP segments received, including those received in error. This count includes segments received on currently established connections.
166	.1.3.6.1.4.1.7564.25.7.0	The total number of ClickTCP segments sent, including those on current connections but excluding those containing only retransmitted octets.
167	.1.3.6.1.4.1.7564.25.8.0	The total number of segments retransmitted - that is, the number of ClickTCP segments transmitted containing one or more previously transmitted octets.
168	.1.3.6.1.4.1.7564.25.9.0	The total number of segments received in error (e.g., bad ClickTCP checksums).
169	.1.3.6.1.4.1.7564.25.10.0	The number of ClickTCP segments sent containing the RST flag.
170	.1.3.6.1.4.1.7564.25.11	A table containing ClickTCP connection-specific information.
171	.1.3.6.1.4.1.7564.25.11.1	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.
172	.1.3.6.1.4.1.7564.25.11.1.1	A unique value for each clicktcp connection.
173	.1.3.6.1.4.1.7564.25.11.1.2	The state of this TCP connection.
174	.1.3.6.1.4.1.7564.25.11.1.3	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.
175	.1.3.6.1.4.1.7564.25.11.1.4	The local port number for this TCP connection.
176	.1.3.6.1.4.1.7564.25.11.1.5	The remote IP address for this TCP connection.
177	.1.3.6.1.4.1.7564.25.11.1.6	The remote port number for this TCP connection.
178	.1.3.6.1.4.1.7564.27.1.1.0	The number of real services being checked.

SNMP OID List		
179	.1.3.6.1.4.1.7564.27.1.2	Health Check statistics table.
180	.1.3.6.1.4.1.7564.27.1.2.1	An hcStatsTable entry containing health check statistics for one real service.
181	.1.3.6.1.4.1.7564.27.1.2.1.1	Reference index for each real service being checked.
182	.1.3.6.1.4.1.7564.27.1.2.1.2	Real service name.
183	.1.3.6.1.4.1.7564.27.1.2.1.3	Health Check IP address.
184	.1.3.6.1.4.1.7564.27.1.2.1.4	Health Check port.
185	.1.3.6.1.4.1.7564.27.1.2.1.5	The status (UP/DOWN) of the health check.
186	.1.3.6.1.4.1.7564.27.1.2.1.6	The reason why the health check is being marked UP/DOWN.
187	.1.3.6.1.4.1.7564.27.1.2.1.7	The number of times the health check is down.
188	.1.3.6.1.4.1.7564.27.1.2.1.8	The number of times the health check is up.
189	.1.3.6.1.4.1.7564.27.1.2.1.9	The number of connections attempted.
190	.1.3.6.1.4.1.7564.27.1.2.1.10	The number of successful connections.
191	.1.3.6.1.4.1.7564.27.1.2.1.11	The number of connection failures.
192	.1.3.6.1.4.1.7564.28.1.0	Total number of bytes received.
193	.1.3.6.1.4.1.7564.28.2.0	Total number of bytes sent.
194	.1.3.6.1.4.1.7564.28.3.0	Number of bytes received per second.
195	.1.3.6.1.4.1.7564.28.4.0	Number of bytes sent per second.
196	.1.3.6.1.4.1.7564.28.5.0	Peak received bytes per second.
197	.1.3.6.1.4.1.7564.28.6.0	Peak sent bytes per second.
198	.1.3.6.1.4.1.7564.28.7.0	Number of currently active transaction.
199	.1.3.6.1.4.1.7564.30.1.0	Current percentage of CPU utilization.
200	.1.3.6.1.4.1.7564.30.2.0	Number of connections per second.
201	.1.3.6.1.4.1.7564.30.3.0	Number of requests per second.
202	.1.3.6.1.4.1.7564.31.1.0	Total DNS requests.
203	.1.3.6.1.4.1.7564.31.2.0	Total successful DNS resolvings.
204	.1.3.6.1.4.1.7564.31.3.0	Total failed DNS resolvings.
205	.1.3.6.1.4.1.7564.31.4.0	Total DNS requests in the last second.
206	.1.3.6.1.4.1.7564.31.5.0	Total successful DNS resolvings in the last second.
207	.1.3.6.1.4.1.7564.31.6.0	Total failed DNS resolvings in the last second.
208	.1.3.6.1.4.1.7564.31.7.0	Peak DNS requests in a second.
209	.1.3.6.1.4.1.7564.31.8.0	Peak successful DNS resolvings in a second.
210	.1.3.6.1.4.1.7564.31.9.0	Total DNS requests in the last minute.
211	.1.3.6.1.4.1.7564.31.10.0	Total successful DNS resolvings in the last minute.
212	.1.3.6.1.4.1.7564.31.11.0	Total failed DNS resolvings in the last minute.
213	.1.3.6.1.4.1.7564.31.12.0	Peak DNS requests in a minute.
214	.1.3.6.1.4.1.7564.31.13.0	Peak successful DNS resolvings in a minute.
215	.1.3.6.1.4.1.7564.31.14.0	Total DNS requests in the last hour.

SNMP OID List		
216	.1.3.6.1.4.1.7564.31.15.0	Total successful DNS resolvings in the last hour.
217	.1.3.6.1.4.1.7564.31.16.0	Total failed DNS resolvings in the last hour.
218	.1.3.6.1.4.1.7564.31.17.0	Peak DNS requests in an hour.
219	.1.3.6.1.4.1.7564.31.18.0	Peak successful DNS resolvings in an hour.
220	.1.3.6.1.4.1.7564.31.19.0	Total DNS requests in the last day.
221	.1.3.6.1.4.1.7564.31.20.0	Total successful DNS resolvings in the last day.
222	.1.3.6.1.4.1.7564.31.21.0	Total failed DNS resolvings in the last day.
223	.1.3.6.1.4.1.7564.31.22.0	Peak DNS requests in a day.
224	.1.3.6.1.4.1.7564.31.23.0	Peak successful DNS resolvings in a day.
225	.1.3.6.1.4.1.7564.31.24.0	Total DNS requests in the last 5 seconds.
226	.1.3.6.1.4.1.7564.31.25.0	Total successful DNS resolvings in the last 5 seconds.
227	.1.3.6.1.4.1.7564.31.26.0	Total failed DNS resolvings in the last 5 seconds.
228	.1.3.6.1.4.1.7564.31.27.0	Peak DNS requests in 5 seconds.
229	.1.3.6.1.4.1.7564.31.28.0	Peak successful DNS resolvings in 5 seconds.
230	.1.3.6.1.4.1.7564.251.1	Excute "snmp on" and save configuration, reboot system. When system bootup, this trap will be sent.
231	.1.3.6.1.4.1.7564.251.2	Excute "snmp on", reboot system. When system shutdown, this trap will be sent.
232	Float	A single precision floating-point number. The semantics and encoding are identical for type 'single' defined in IEEE Standard 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.
233	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 Glossary

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
GMT	Greenwich Mean Time
GSLB	Global Server Load Balance (also known as Smart DNS)
HC	Health Check
HTML	HyperText Markup Language

Acronym	Full Spelling
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Socket 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
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

Acronym	Full Spelling
PKI	Public Key Infrastructure
POP3	Post Office Protocol - Version 3
PST	Pacific Standard Time
QoS	Quality of Service
RADIUS	Remote Authentication Dial In User Service
RAM	Random Access Memory
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 (also known as GSLB)
SIP	Session Initiation Protocol
SLB	Server Load Balancing
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSH	Security 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

Acronym	Full Spelling
VCID	Virtual Cluster ID
VIP	Virtual IP
VLAN	Virtual Local Area Network
VOD	Video On Demand
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WebUI	Web User’s Interface
WELF	WebTrends Enhanced Log Format

Appendix III XML RPC Methods

The following table lists all XML RPC methods supported by Fortinet 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**
FBLOS_cli _enable	All commands in Enable mode	{num,int}, {cli_string0,string }, {cli_string1,string }, {cli_string2,string }, {cli_string3, string} ……	num	1. If no "num" value is given, it defaults to 1 and "cli_string0" should be configured. 2. The names of CLI strings should start from "cli_string0" and end at "cli_string{n-1}". 3. If some intermediate CLI strings are missing, the XML RPC system will just ignore and not complain.
FBLOS_cli _config	All commands in Config mode			

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_time out	"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},		"noarp" is of type integer. "name" is the name of virtual

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
		{ip,string}, {port,string}, {noarp, int}		service.
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	1. "threshold" and "threshold_rr" are optional parameters for "lc" and "sr" methods. 2. "cookie" and "path_attr" are optional parameters for "ic" method. 3. "cookie" is required and "offset" is optional for "rc" method. 4. "header" is a required parameter for "hh" method. 5. "sess_timeout" is an optional parameter for "sslsid" method. 6. Parameters in the XML RPC message which don't 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	1. "name" is the name of the group and "real_name" is the name of real service. 2. "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},		1. "policy" is the type of policy. The values of

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
		{virtual_name, string}, {group_name, string}, {real_name,string} {regex,string}, {policy_string, string}, {precedence,int}, {ip,string}, {netmask, string}		"policy" can be one of "static", "default", "regex", "icookie", "rcookie", "persistent url", "persistent cookie", "qos cookie", "qos hostname", "qos url", and "qos network". 2. "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. 3. "precedence" is required for "regex", "icookie", "rcookie", "persistent url", "persistent cookie", "qos cookie", and "qos hostname", "qos url" and "qos network". 4. "regex" is required for "regex" policy. 5. "policy_string" is required for "persistent url", "persistent cookie", "qos url", "qos cookie", and "qos hostname". 6. "ip" and "netmask" are required for "qos network" policy.
no_slb_policy	no slb policy	{policy, string}, {name, string}, {virtual_name,		policy" is the type of policy, "name" is the name of policy and "virtual_name" is the

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
		string}		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_virtual_on	cluster virtual on	{vcid,int}, {interface, string}	Vcid interface	
cluster_virtual_off	cluster virtual off	{vcid,int}, {interface, string}	Vcid interface	
cluster_virtual_ifname	cluster virtual ifname	{vcid,int}, {interface, string}		
cluster_virtual_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_virtual_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_virtual_preempt	cluster virtual preempt	{vcid,int}, {interface,string}, {preempt, int}		"preempt" value should be 0 or 1.
no_cluster_virtual_preempt	no cluster virtual preempt	{vcid,int}, {interface, string}		
cluster_virtual_interval	cluster virtual interval	{vcid,int}, {interface,string}, {interval, int}	interval	"interval" value should be between 3 and 10 and the default value is 5.
no_cluster_virtual_inte	no cluster virtual interval	{vcid,int}, {interface, string}		

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
rval				
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}		