ArrayOS APV 10.4.3

User Guide

POWERED BY SpeedCore™

Copyright Statement

Warning: Modifications made to the Array Networks unit, unless expressly approved by Array Networks, could void the user’s authority to operate the equipment.

Revision History

Date	Description
February 21, 2022	First release.
July 15, 2022	Updated for APV 10.4.3 release.

Table of Contents

Chapter 1 Initial System Setup & Configuration

1.1 Overview

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

1.1.1 Connecting to APV

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

- Console (recommended)
- SSH
- WebUI

1.1.1.1 Console Connection

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

Table 1–1 Console Setup

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

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

1.1.1.2 SSH Connection

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

Note: If you require SSH software for Windows, 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:

Run the SSH program on your workstation.

```
>> # ssh array@10.3.55.251
```

10.3.55.251 is the APV appliance's IP address.

After you establish a connection, the APV appliance will ask you for a privilege password.

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

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

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

1.1.1.3 WebUI Connection

This section introduces the connection method via APV WebUI (Web User Interface). It provides graphical user interface for configuring and managing the APV appliance in a visual and friendly way, which greatly simplifies the configuration operation but achieves the same configuration effects as the CLI.

The APV WebUI can:

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

Note: WebUI is disabled by default. To configure and manage the APV appliance using the WebUI, please see section 1.2.2.4 "Setting up the WebUI".

The APV appliance provides WebUI for system management and configuration, which can be accessed by entering the IP address and port number of the appliance in the address bar of your browser.

Example for accessing the WebUI (8888 is the default port number for accessing the WebUI):

```
https://192.168.1.200:8888
```

And then press "Enter". The welcome screen should appear, prompting for username and password. The default username and password is **array** and **admin**.

The WebUI supports Chrome, Safari and Edge browsers. Browser resolution should be set to 1024×786 or higher.

1.1.1.4 WebUI SSL Configuration

1.1.1.4.1 SSL Client Authentication Settings

In SSL communication, if there is no need to authenticate the client identity, the SSL negotiation process usually only authenticates the identity of the server, that is, performs one-way SSL authentication. In some scenarios that have strict requirements on the client identity, the SSL negotiation process needs to authenticate both the client identity and the server identity, that is, two-way SSL authentication needs to be performed.

The APV appliance supports the WebUI SSL client authentication function to meet the two-way SSL authentication requirements in specific scenarios. At the same time, APV also supports mandatory mode for the WebUI SSL client authentication function. When the mandatory mode is enabled, the WebUI client must pass the client authentication before establishing an SSL connection with the APV WebUI. Otherwise, the WebUI access will fail. If SSL client authentication is enabled but the mandatory mode is not enabled, the administrator can still access the WebUI, but the SSL client will not provide the client certificate.

- **Configuration Example via CLI**

To enable the WebUI SSL client authentication function, perform the following steps:

1. Import certificate chain file for WebUI.

```
AN(config)#webui ssl import certificate ftp://10.8.6.20/cert/chain.pem
```

Import client CA certificate.

```
AN(config)#webui ssl import clientca ftp://10.8.6.20/cert/webui.pem
```

Enable the SSL client authentication function for the WebUI.

```
AN(config)#webui ssl settings clientauth enable
```

Enable the mandatory mode of the SSL client authentication function for the WebUI.

```
AN(config)# webui ssl settings authmandatory enable
```

1.1.1.4.2 SSL Protocol Versions and Cipher Suites Settings

The system also supports modifying the SSL protocol versions and cipher suites supported by the WebUI (SSL server).

Use the following command to modify the SSL protocol versions supported by the WebUI:

```
AN(config)#webui ssl settings protocol TLSv11:TLSv12
```

Use the following command to modify the cipher suites supported by the WebUI:

```
AN(config)#webui ssl settings ciphersuites
"ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
```

ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-GCM-SHA384"

1.1.2 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

1.1.2.1 Command Usage Breakdown

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

The APV 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 1–2 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 APV CLI commands will generally adhere to the following style conventions:

Table 1–3 APV CLI Style Conventions

Style	Convention
Bold	The body of a CLI command is in Boldface.
Italic	CLI parameters are in Italic.
< >	Parameters in angle brackets < > are required.
[ ]	Parameters in square brackets [ ] are optional. Subcommand such as "**no**", "**show**" and "**clear**" commands.
{x|y|…}	Alternative items are grouped in braces and separated by vertical bars. At least one should be selected.

Style	Convention
[x|y|…]	Optional alternative items are grouped in square brackets and separated by vertical bars. One or none is selected.

For example:

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

Note: It is recommended to enclose the string-type paramter value by doube quotes to make sure that the appliance can execute the command correctly.

1.1.2.2 Levels of Access Control

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

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

```
AN>
```

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 APV appliance prompts the user for the appropriate password. If correct password is entered, the CLI prompt will change from "AN>" to "AN#", which means the user is granted access to the Enable level. The default password for the Enable level is null, i.e. users simply need to press "Enter".

```
AN>enable
Enable password:
AN#
```

The final access level is the Config level. It is with this level of authority that the user can make changes to the configuration of the box. By default, no two users can access the Config level at the same time. User of the Config level authority can implement commands in all three levels. To gain access to the full configurable functions of the APV appliance, the user must use the following command:

```
AN#config terminal
```

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

```
AN(config)#
```

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

```
AN#config terminal force
```

In addition, the system supports the multiconfig function. That is, the system can allow multiple administrator accounts to be simultaneously in the Config mode or one administrator account to simultaneously establish multiple SSH connections of the Config level authority. Administrator can enable this feature by executing the command “config multiconfig enable”.

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

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

1.2 General Settings Configuration

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

1.2.1 Configuration Guidelines

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

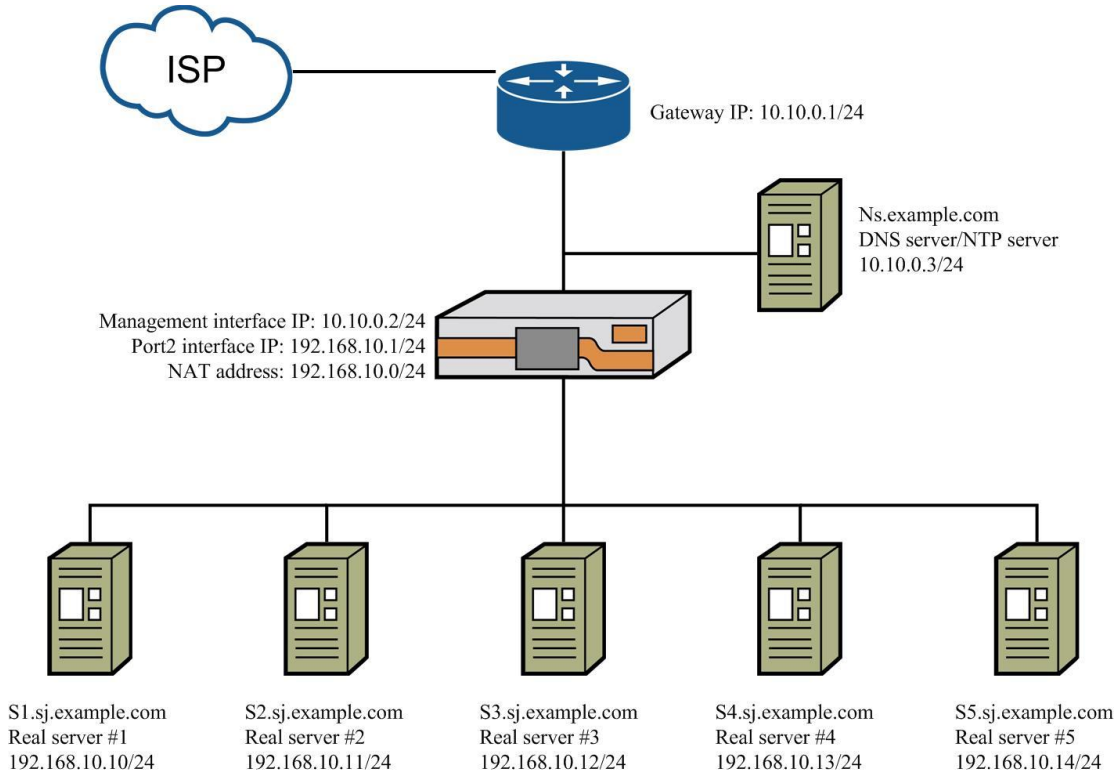


Figure 1–1 Basic Network Architecture

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

Table 1–4 Basic Network Configurations

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

Table 1–5 General Settings of Basic Network Configuration

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}* **ip dhcp {on|off}** *<interface_name>*
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 Configuration Example via CLI

1.2.2.1 Assigning the IP Address for Interfaces

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

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

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

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

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

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

1.2.2.2 Assigning the IP Address for Gateway

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

To define the gateway IP address:

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

1.2.2.3 Viewing the IP Configuration

To verify that APV 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:

```
AN(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:

```
AN(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:

```
AN(config)#show ip address
ip address "port1" 10.10.0.2 255.255.255.0
ip address "port2" 192.168.10.1 255.255.255.0

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

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

1.2.2.4 Setting up the WebUI

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

In our example, we use the port1 interface IP address as the default WebUI IP address, and the default port number 8888 to access the WebUI.

Execute the following command to enable the WebUI:

```
AN(config)#webui on
```

Note:

To change the IP address for accessing the WebUI, use the "**webui ip**" command.

To change the port number for accessing the WebUI, use the "**webui port**" command.

Now you can access the WebUI of the APV appliance for configuration and management via the Web browser. For detailed information about how to use the WebUI, please see related introductions in section 1.1.1.3 "WebUI Connection".

1.2.2.5 Assigning the Host Name

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

```
AN(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 APV appliance calls upon at reboot.

1.2.2.7 Override Configuration

To achieve configuration override when executing commands repeatedly for the same virtual service or real service, execute the following command:

```
AN(config)#system command override on
```

This feature is enabled by default. To disable the command override function, execute the following command:

```
AN(config)#system command override off
```

Chapter 2 Advanced Network Configuration

2.1 Overview

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

Note: Basic network configurations such as IP address, the default route and advanced system configurations are usually associated with or used by other configurations. After these configurations are set up, please do not modify or change them, or it may lead to unpredicted errors. To modify network configurations, please execute the commands "**write memory/file**", "**clear config all**" and "**config memory/file**" in turn after modifying network configurations.

2.1.1 VLAN

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

- Port-based VLAN

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

- Tag-based VLAN

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

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

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

For example:

IPv4-based VLAN:

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

IPv6-based VLAN:

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

2.1.2 MNET

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

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

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

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

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

2.1.3 Port Forwarding

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

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

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

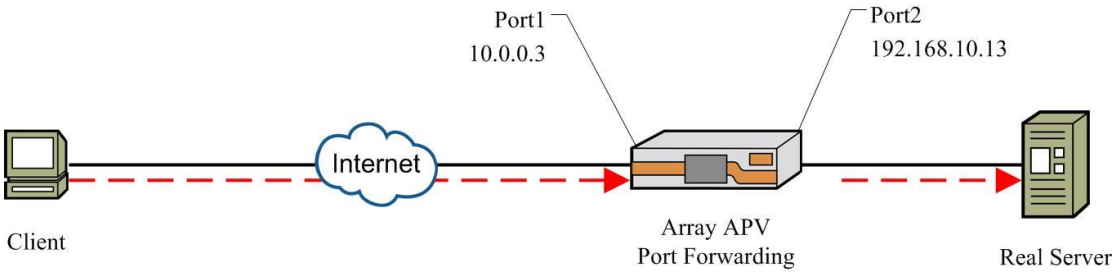


Figure 2–1 Port Forwarding

2.1.4 NAT

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

2.1.4.1 How NAT Works

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

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

NAT Traversal of PPTP

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

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

2.1.4.2 Supported NAT Types

The APV appliance supports four types of NAT:

- **Static NAT**

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

In static NAT, the computer with the IP address (192.168.10.13) will be always translated into 212.0.0.3:

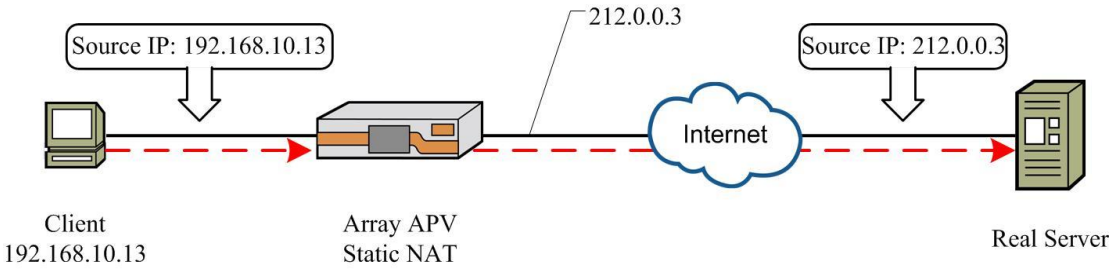


Figure 2–2 Static NAT

- **Port-level NAT**

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

In port-level NAT, the computers with the IP address in the range from 192.168.10.11 to 192.168.10.13 will be translated into 212.0.0.3 with a different port on the outside network:

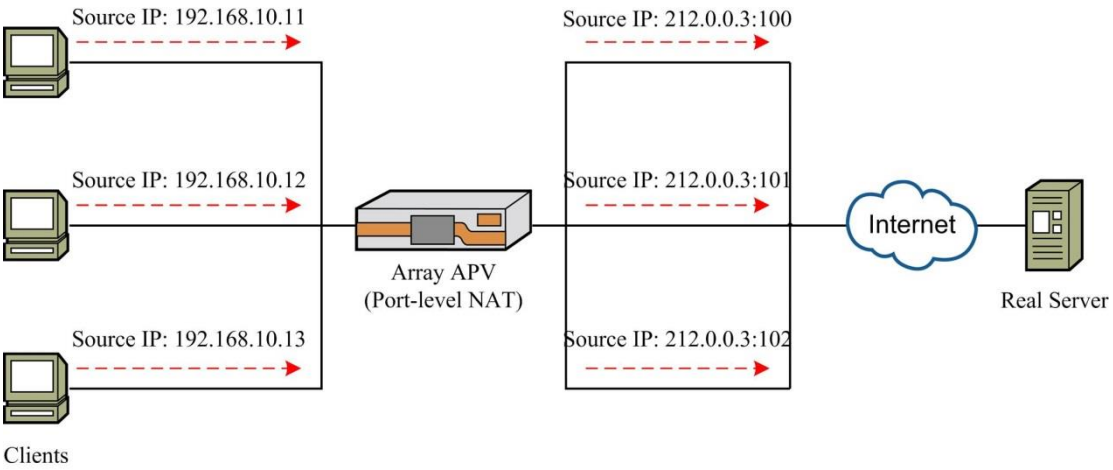


Figure 2–3 Port-level NAT

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

- **IP pool-based dynamic NAT**

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

As shown below, in IP pool-based dynamic NAT, the client IP addresses in the range from 192.168.10.11 to 192.168.10.13 will be translated into 212.0.0.1 to 212.0.0.3 and get connected to the Internet.

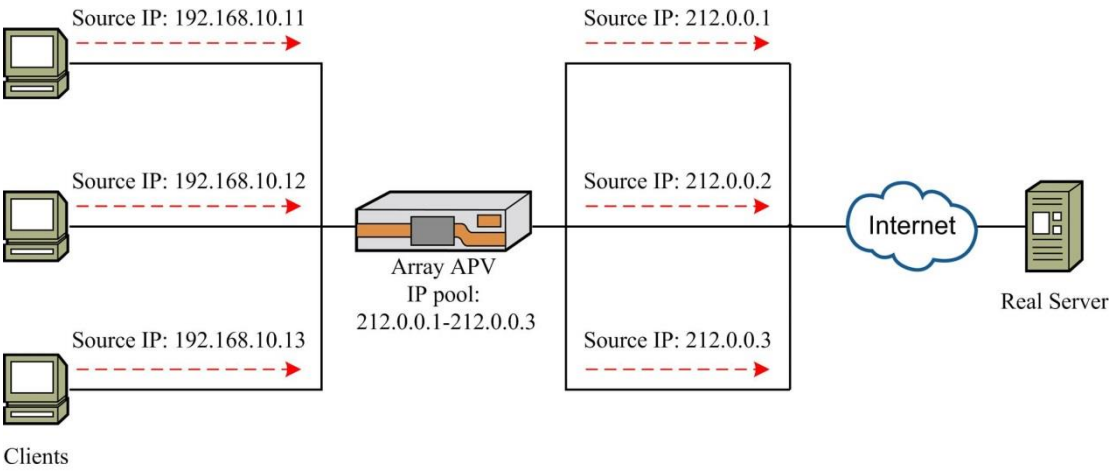


Figure 2–4 IP Pool-based Dynamic NAT

➢ **Destination IP based NAT**

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

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

Configuration Example via CLI

1. Configure IP pool.

```
AN(config)#ip pool "pool1" 212.16.72.45 212.16.72.55
AN(config)#ip pool "pool3" 3ffd::45 3ffd::46
```

Configure destination NAT.

```
AN(config)#nat portdst "pool1" 212.16.72.0 255.255.255.0 500 212.16.72.101
AN(config)#nat portdst "pool3" 3ff1:: 65 50
```

Check NAT configurations and statistics

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

- Check NAT table.

```
AN(config)#show nat table
From 212.16.73.108(5208) through 212.16.72.53(50776) to 212.16.72.77(80)
From 212.16.73.108(5200) through 212.16.72.53(50768) to 212.16.72.77(80)
From 212.16.73.108(5880) through 212.16.72.53(51448) to 212.16.72.77(80)
From 212.16.73.108(6008) through 212.16.72.53(51576) to 212.16.72.77(80)
From 3ffd::108(42720) through 3ffd (45672) to 3ff1::77(80)
```

```
From 3ffd::108(42728) through 3ffd (45680) to 3ff1::77(80)
From 3ffd::108(43336) through 3ffd (46288) to 3ff1::77(80)
From 3ffd::108(43376) through 3ffd (46328) to 3ff1::77(80)
From 3ffd::108(43464) through 3ffd (46416) to 3ff1::77(80)
From 3ffd::108(43408) through 3ffd (46360) to 3ff1::77(80)
From 3ffd::108(43840) through 3ffd (46792) to 3ff1::77(80)
From 3ffd::108(43896) through 3ffd (46848) to 3ff1::77(80)
```

- Check NAT statistics.

```
AN(config)#show statistics nat
nat portdst "pool1" 212.16.72.0 255.255.255.0 500 212.16.72.101
        protocol(total/current):icmp(1/1) udp(0/0) tcp(23724/251)
nat portdst "pool3" 3ff1:: 65 50
        protocol(total/current):icmp(0/0) udp(0/0) tcp(23716/246)
```

2.1.5 DNS NAT

The DNS NAT function is applicable to the scenario where the resolved IPs in the DNS responses returned by internal DNS servers are the IP addresses of internal servers that cannot be publicly accessed. With the DNS NAT function enabled, when receiving DNS responses from an internal DNS server, the appliance will translate the internal resolved IP addresses in the DNS responses to the public IP addresses based on the NAT static entries (configured using the "**nat static**" command) and return the translated DNS responses to the client. When external clients use the publicly accessible IP addresses for access, the system translates the publicly accessible IP addresses back to the internal resolved IP addresses of internal severs based on the NAT static entries. In this way, external clients can access internal servers. By default, this function is disabled.

2.1.6 Website Classification

Website classification is a dynamic website category recognition function that the APV appliance provides by subscribing Webroot BrightCloud's Website Classification service. This function allows the APV appliance to look up the category of a website that clients access via the local cache, local database and online connection to the Webroot server. To use the online lookup function, administrators need to make sure the APV appliance is able to access the external network.

To enable the website classification function, administrators need to import a license into the system. This function supports two types of licenses:

- Trial license: 30-day free trial
- Formal license: 365-day validity period

After the license expires, the website classification function will be unavailable. The Webroot server will deny the APV appliance's access and the APV appliance will stop sending website

category lookup queries to the server. To obtain a license, please contact Customer Support and provide the device model and serial number.

2.1.7 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), OSPFv2 (Open Shortest Path First version 2) and OSPFv3 (Open Shortest Path First version 3), and BGP (Border Gateway Protocol, including multi-protocol extensions).

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

2.1.8 IP Pool

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

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

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

Table 2–1 Maximum IP Pool Entry

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

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

When configuring the IP pool, please note:

- Each IP pool should be assigned with a unique name in the system.
- The IP addresses in one IP pool must belong to the same subnet.

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

2.1.9 VXLAN

The appliance supports the Virtual eXtensible Local Area Network (VXLAN) function. VXLAN adopts the packet encapsulation mode of MAC-over-UDP to encapsulate Layer 2 packets with Layer 3 protocol. In this way, the Layer 2 network can be extended within the Layer 3 range, which meets the demand of the large Layer 2 virtual migration of data center. VXLAN uses the 24-bit VXLAN Network Identifier (VNI) to distinguish different VXLAN networks. It can support isolation for up to 16 million tenants and solve the problem of mass tenant isolation in cloud computing.

2.1.9.1 General Concepts

VXLAN includes the following general concepts:

- VNI: VXLAN network uses VNI to distinguish different VXLAN networks. Virtual machines which belong to different VNIs cannot communicate directly at Layer 2.
- Tunnel: VXLAN tunnel is a virtual tunnel established over the Underlay network, and is used to transmit the encapsulated VXLAN packets.
- VTEP: VXLAN Tunnel End Point (VTEP) is the starting and ending point of the VXLAN tunnel and is responsible for VXLAN encapsulation and de-encapsulation of the original Ethernet packets. VTEP at one end of VXLAN tunnel encapsulates the Layer 2 datagram, and then sends it through VXLAN tunnel. After receiving the encapsulated datagram, VTEP at the other end decapsulates the datagram. VTEP can be any device that supports VXLAN. Acting as a VTEP, APV can be either a Layer 3 gateway or a Layer 2 gateway.

2.1.9.2 VXLAN Working Mode

VXLAN tunnel can transmit unicast, broadcast and multicast datagram packets, supporting two working modes:

- P2MP: in this working mode, for a VXLAN, the administrator needs to create tunnels for the appliance with all other VTEPs.

- Multicast: in this working mode, for a VXLAN, the administrator needs to create a multicast tunnel to this VXLAN for the appliance.

Besides, VXLAN tunnel supports both IPv4 and IPv6. IPv6 packets can be transmitted through the IPv4 tunnel and IPv4 packets can be transmitted through IPv6 tunnel.

2.1.9.3 VXLAN Learning

The appliance supports the VXLAN learning function. When this function is enabled, the appliance automatically learns the mapping relationship between the VTEP IP and MAC addresses and creates the forwarding table. The lifetime of automatically learned entries is 20 minutes.

2.1.10 Interface Shutdown

The appliance supports manually shutting down the specified physical interface. After the physical interface is shut down, properties of the interface, such as configuration options, will change. For details, refer to the "interface shutdown" command in the CLI handbook.

After the physical interface is manually shut down, the system displays "status: down (Administratively down)" when the administrator executes the "show interface" command.

➢ **Configuration Example**

Execute the following command to shut down the specified physical interface.

```
AN(config)#interface shutdown port2
```

Execute the following command to cancel the shutdown of the specified physical interface.

```
AN(config)#no interface shutdown port2
```

Note: After the physical interface is manually shut down, the interface status will be displayed as "status: down (Administratively down)". In this case, the system will not check the interface status anymore. If the administrator continues to unplug the cable, the interface status will still be "down (Administratively down)".

2.2 Advanced Network Configuration

2.2.1 Configuration Guidelines

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

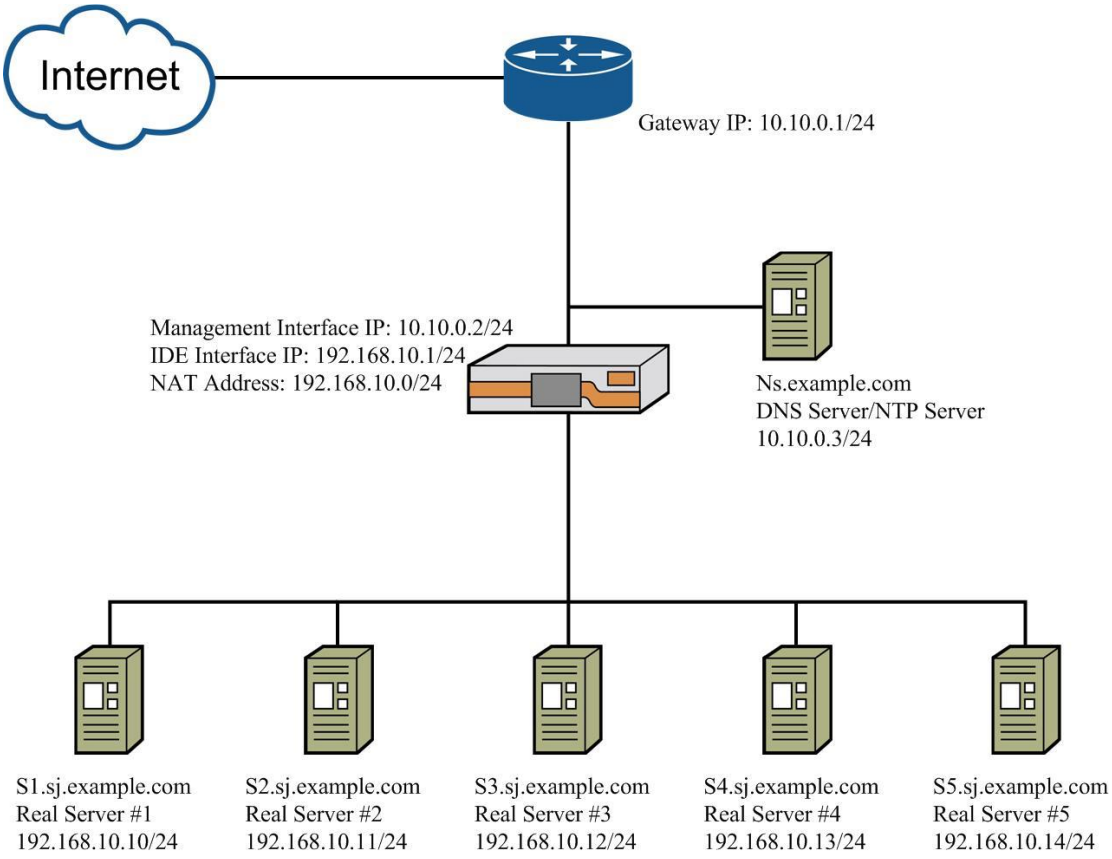


Figure 2–5 Advanced Network Architecture

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

Table 2–2 Advanced Network Configurations

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

Table 2–3 General Settings of Advanced Network Configuration

Operation	Command
Configure VLAN	**vlan** *{system_ifname|bond_ifname}* *<user_interface_name>* *<vlan_tag>*
Configure MNET	**mnet** *{system_ifname|bond_ifname|vlan_ifname}* *<user_interface_name>*
Configure Port Forwarding	**fwd tcp** *<local_ip>* *<local_port>* *<remote_ip>* *<remote_port>* *[timeout]* **fwd udp** *<local_ip>* *<local_port>* *<remote_ip>* *<remote_port>*

Operation	Command
	[timeout]
Configure NAT	**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]* **nat static** *<vip> <network_ip> [timeout] [gateway] [description]*
Configure Dynamic Routing	**rip {on|off}** **rip network** *<ip_address> <netmask>* **ospf {on|off}** **ospf network** *<ip_address> <netmask> <area_id>*
Configure IP pool	**ip pool** *<pool_name> <start_ip> [end_ip]* **slb proxyip global** *<pool_name>* **slb proxyip group** *<group_name> <pool_name>*

2.2.2 Configuration Example via CLI

2.2.2.1 VLAN Configuration

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

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

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

Assign an IP address to each VLAN interface by using the "**ip address**" command.

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

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

2.2.2.2 MNET Configuration

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

1. Define our mnet interfaces by using the "**mnet**" command.

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

Assign an IP address to each MNET by using the "**ip address**" command.

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

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

2.2.2.3 Port Forwarding Configuration

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

```
AN(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 APV appliance since other services might be listening on those ports, i.e. 443 (for SSL) and 80 (for HTTP). We can choose a port below 1024 on the real server since that is the service that we want to connect to. To view or alter these forwarding instructions, employ the show, no or clear versions of the above commands.

2.2.2.4 NAT Configuration

For our configuration example strategy, use the command as:

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

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

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

```
AN(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:

```
AN(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:

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

2.2.2.5 DNS NAT

- **Configuration Objectives**

In the following scenario, a user sends a DNS query “www.abc.com” for DNS resolution. The APV appliance is used for Link Load balancing (LLB).

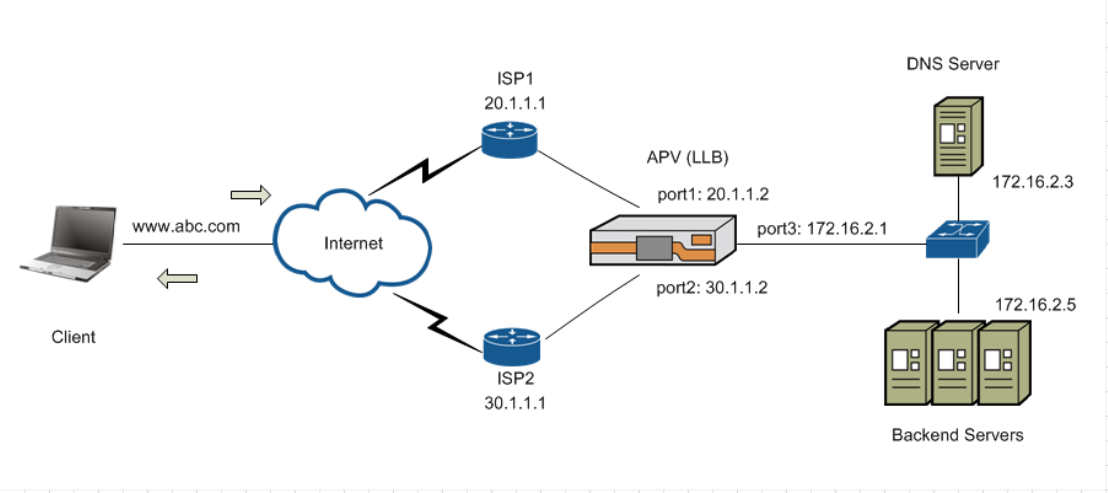


Figure 2–6 DNS NAT Scenario

The configuration objectives are as follows:

- The DNS responses with the public IP address “20.1.1.5” or “30.1.1.5” as the resolved IP addresses will be returned to users.
- Users can successfully access the backend server “172.16.2.5” using the resolved public IP address.

➢ **Configuration Steps**

1. Configure NAT entries.

```
AN(config)#nat static 20.1.1.3 172.16.2.3
AN(config)#nat static 30.1.1.3 172.16.2.3
AN(config)#nat static 20.1.1.5 172.16.2.5
AN(config)#nat static 30.1.1.5 172.16.2.5
```

Configure LLB link routes.

```
AN(config)#llb link route ISP1 20.1.1.1
AN(config)#llb link route ISP2 30.1.1.1
```

Configure LLB health checks.

```
AN(config)#llb link health checker icmp ISP1 “20.1.1.1” 10 5 3 3
AN(config)#llb link health checker icmp ISP2 “30.1.1.1” 10 5 3 3
```

Enable the DNS NAT function.

```
AN(config)#nat protocol dns
```

Note: In non-LLB environment, only steps 1 and 4 should be configured.

2.2.2.6 Dynamic Routing Configuration

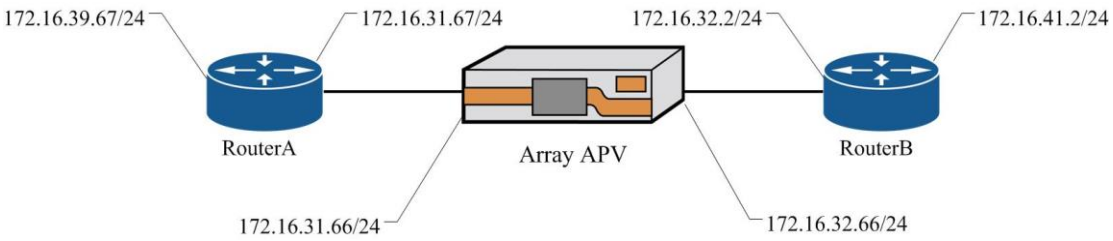


Figure 2–7 Dynamic Routing Configuration

1. RIP Configurations.

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

OSPF Configurations.

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

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

```
AN(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 APV appliance to operate seamlessly within the network architecture.

2.2.2.7 IP Pool Configuration

Configuration Example for NAT IP Pool via CLI

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

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

```
AN(config)#ip pool "pool1" 124.0.0.22 124.0.0.22
```

```
AN(config)#ip pool "pool2" 124.0.1.22 124.0.1.22
```

Define the IP pool for NAT via the "**nat port**" command.

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

Configuration Example for SLB IP Pool via CLI

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

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

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

Define the IP pool as the global proxy IP pool by using the "**slb proxyip global**" command.

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

Assign the IP pools for SLB group.

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

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

2.2.2.8 VXLAN Configuration Example

2.2.2.8.1 Deploying APV as VXLAN Layer 3 Gateway

In the following figure, clients in the subnet 192.10.10.100/24 need to communicate with virtual machines VM1 and VM2. APV acts as a VXLAN layer 3 gateway.

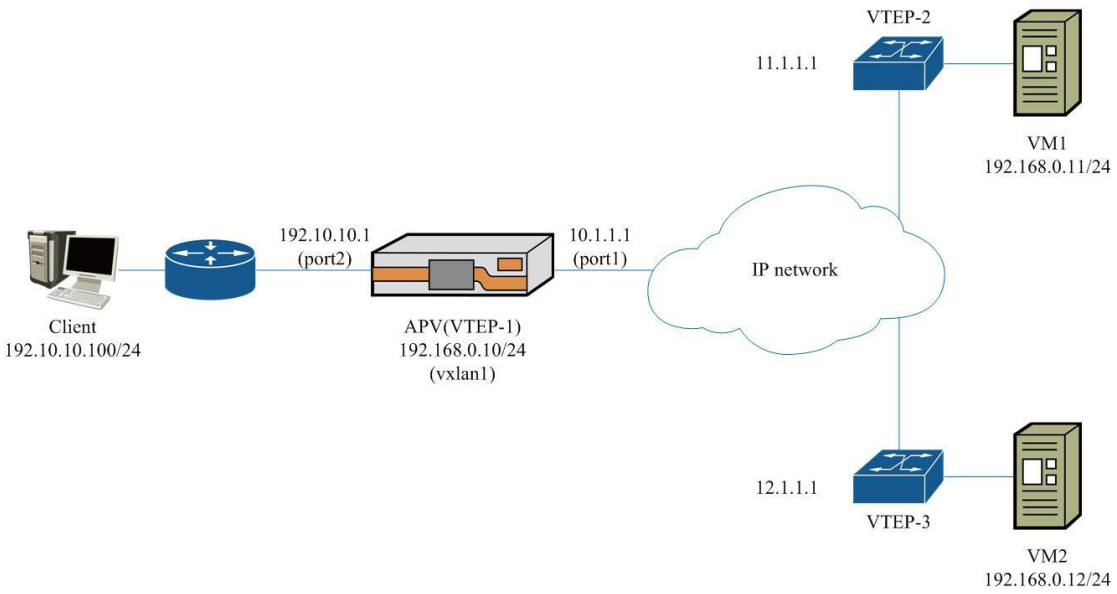


Figure 2–8 VXLAN Layer 3 Gateway

The communication flow between the client and the virtual machine is as follows:

1. The client sends VXLAN data packets.

When receiving data packets, the appliance queries the routing table and finds that data packets need to be forwarded to an appliance in the VXLAN network, assuming it is VM1. The appliance queries the forwarding table and retrieves the IP address of VTEP-2, encapsulates the data packets and sends the data packets to VTEP-2 through the VXLAN tunnel.

VTEP-2 sends the data packets to VM1.

When receiving data packets returned by VM1, VTEP-2 sends the data packets to the appliance through the tunnel.

The appliance encapsulates the data packets and forwards them to the client. Meantime, if the VXLAN learning function is enabled, the appliance will learn the mapping relationship between the VTEP IP and MAC address, record it in the forwarding table or refresh the existing records.

The client receives the returned data packets.

- **CLI**

1. Set the VXLAN working mode.

```
AN(config)#vxlan mode p2mp
```

Set the destination port number for the VXLAN tunnel.

```
AN(config)#vxlan port 4789
```

Enable the VXLAN learning function.

```
AN(config)#vxlan learn on
```

Create a VXLAN interface and associate it with the specified VXLAN.

```
AN(config)#vxlan interface vxlan1 9000
```

Configure an IP address for the VXLAN interface.

```
AN(config)#ip address vxlan1 192.168.0.10 24
```

Create two VXLAN tunnels to remote VTEPs.

```
AN(config)#vxlan tunnel tunnel1 10.1.1.1 11.1.1.1
AN(config)#vxlan tunnel tunnel2 10.1.1.1 12.1.1.1
```

Bind the VXLAN tunnels to the specified VXLAN interface.

```
AN(config)#vxlan bind vxlan1 tunnel1
AN(config)#vxlan bind vxlan2 tunnel2
```

Enable the VXLAN function.

AN(config)#**vxlan enable**

2.2.2.8.2 Deploying APV as VXLAN Layer 2 Gateway

In the following figure, clients in the subnet 192.168.0.20/24 need to communicate with virtual machines VM1 and VM2. APV acts as a VXLAN layer 2 gateway.

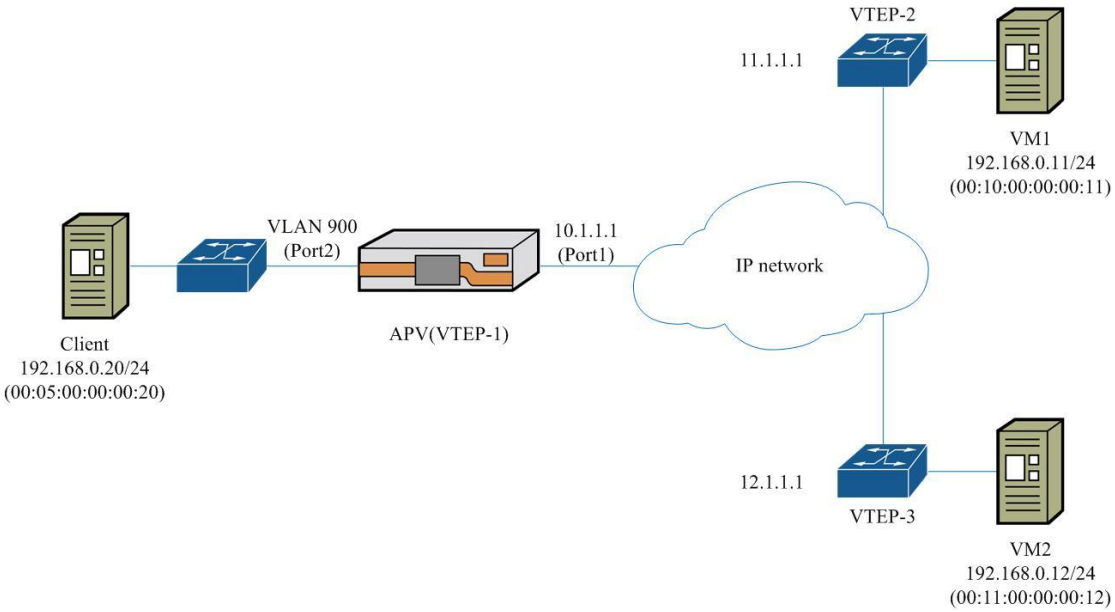


Figure 2–9 VXLAN Layer 2 Gateway

The communication flow between the client and the virtual machine is as follows:

1. The client sends VLAN data packets.

When receiving the data packets, the appliance finds that the data packets need to be forwarded to an appliance in the VXLAN network. Assume it is VM1 and the destination MAC address is 00:10:00:00:00:11. The appliance queries the forwarding table according to the MAC address, retrieves the IP address of VTEP-2, encapsulates the data packets and sends the data packets to VTEP-2 through the VXLAN tunnel.

VTEP-2 sends the data packets to VM1.

When receiving data packets returned by VM1, VTEP-2 sends the VXLAN data packets to the appliance through the VXLAN tunnel.

The appliance received the VXLAN data packets from VTEP-2. The appliance encapsulates the data packets and forwards them to the associated VLAN network. Meantime, if the VXLAN learning function is enabled, the appliance will learn the mapping relationship between the VTEP IP and MAC address, record it in the forwarding table or refresh the existing records.

The client receives the returned data packets.

Note: In step 2:

- When the destination MAC address of the data packet is a multicast or broadcast address, the appliance encapsulates the data packet as a VXLAN packet. In P2MP mode, the data packets are sent to each VTEP one by one through each VXLAN tunnel; in Multicast mode, the data packets are sent to all VTEPs belonging to the same multicast group through the multicast tunnel.
- If there is no matched forwarding entry, the data packet is encapsulated as a VXLAN packet. In P2MP mode, the data packets are sent to each VTEP one by one through each VXLAN tunnel; in Multicast mode, the data packets are sent to all VTEPs belonging to the same multicast group through the multicast tunnel.

➢ **CLI**

1. Set the VXLAN working mode.

```
AN(config)#vxlan mode p2mp
```

Set the destination port number of the VXLAN tunnel.

```
AN(config)#vxlan port 4789
```

Create a VXLAN interface and associate it with the specified VXLAN.

```
AN(config)#vxlan interface vxlan1 9000
```

Create two VXLAN tunnels to remote VTEPs.

```
AN(config)#vxlan tunnel tunnel1 10.1.1.1 11.1.1.1
AN(config)#vxlan tunnel tunnel2 10.1.1.1 12.1.1.1
```

Create a VLAN interface.

```
AN(config)#vlan port2 vlan1 900
```

Bind the VXLAN tunnels to the specified VXLAN interface.

```
AN(config)#vxlan bind vxlan1 tunnel1
AN(config)#vxlan bind vxlan2 tunnel2
```

Associate the VLAN interface with the VXLAN interface.

```
AN(config)#vxlan associate vxlan1 vlan1
```

Add two static VXLAN forwarding entries to the remote VTEP.

```
AN(config)#vxlan forwarding remote 9000 00:10:00:00:00:11 11.1.1.1
AN(config)#vxlan forwarding remote 9000 00:11:00:00:00:12 12.1.1.1
```

Add a static VXLAN forwarding entry to the L2 interface.

```
AN(config)#vxlan forwarding local 9000 00:05:00:00:00:20
```

Note: If the VXLAN learning function is enabled, step 8 and 9 are not required.

Enable the VXLAN function.

AN(config)#**vxlan enable**

Chapter 3 Link Aggregation

3.1 Overview

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

3.2 Understanding Link Aggregation

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

The APV appliance supports at most 8 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 APV appliance also supports configuring MNET or VLAN on bond interface. The bond interface configuration must be performed before configuring MNET or VLAN on it.

3.3 Link Aggregation Health Check

The Link Aggregation Health Check is used to determine the health status (“up” or “down”) of the bond interface. The Link Aggregation Health Check allows the APV appliance to check every subinterface of the bond interface, and mark the subinterface as “up” or “down”. With the Link Aggregation Health Check, the users can use the subinterface which is marked as “up” to transmit traffic.

Note: When the health status of all the bond subinterfaces is marked “down” with the transmission of ARP and IPv6 NS packets still behaving normally, the bond interface will still be in a usable state, and all the service traffic will be sent to every subinterface.

Meanwhile, please check if the destination IP of the bond health check is configured properly.

3.4 Link Aggregation Configuration

3.4.1 Configuration Guidelines

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

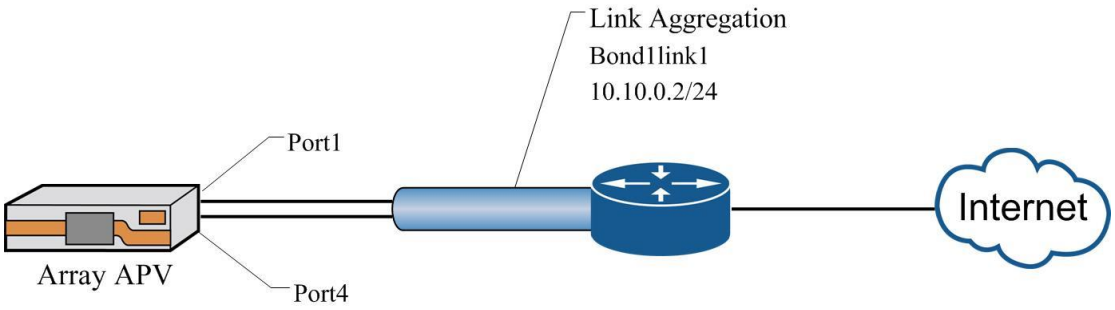


Figure 3–1 Link Aggregation Configuration

Table 3–1 General Settings of Link Aggregation

Operation	Command
Bond system interfaces	**bond interface** *<bond_name> <interface_name> [1|0]*
Assign a name for the bond interface	**bond name** *<bond_id> <bond_name>*
Assign IP address to the bond interface	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix}*
Assign default route	**ip route default** *<gateway_ip>*
Configure link aggregation health check	**bond health** *<bond_name> <destination_ip> [interval] [timeout] [up_check_times] [down_check_times] [gateway_ip]*

3.4.2 Configuration Example via CLI

1. Bind system interfaces with bond interface.

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

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

Assign a name for the bond interface.

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

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

Assign an IP address and netmask to the bond interface.

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

Set the gateway IP address.

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

To verify that APV 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.

```
AN(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
```

Configure and enable the Link Aggregation health check. (Optional).

```
AN(config)#bond health link1 172.16.77.81 5 3 3 3 172.16.77.1
```

Chapter 4 Clustering

4.1 Overview

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

4.2 Understanding Clustering

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

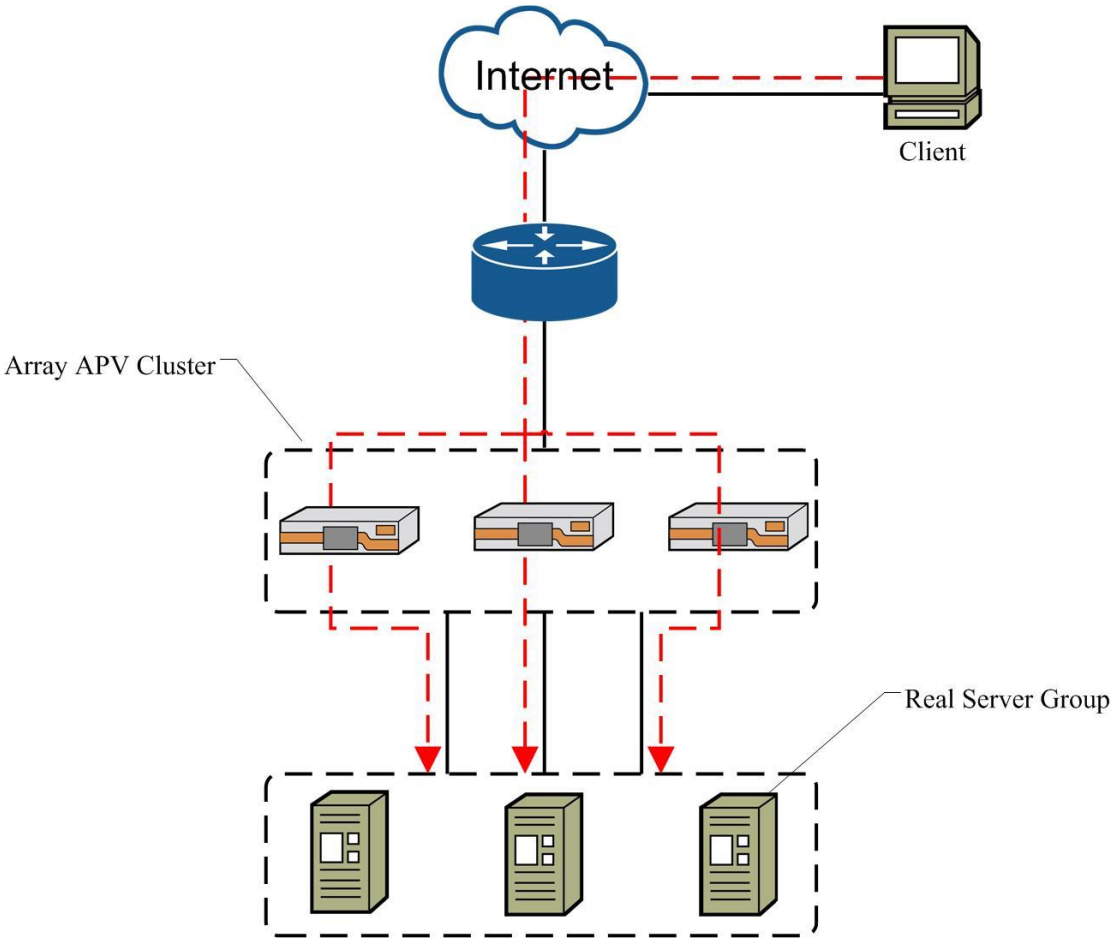


Figure 4–1 APV Clustering

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

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

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

4.2.1 Fast Failover

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

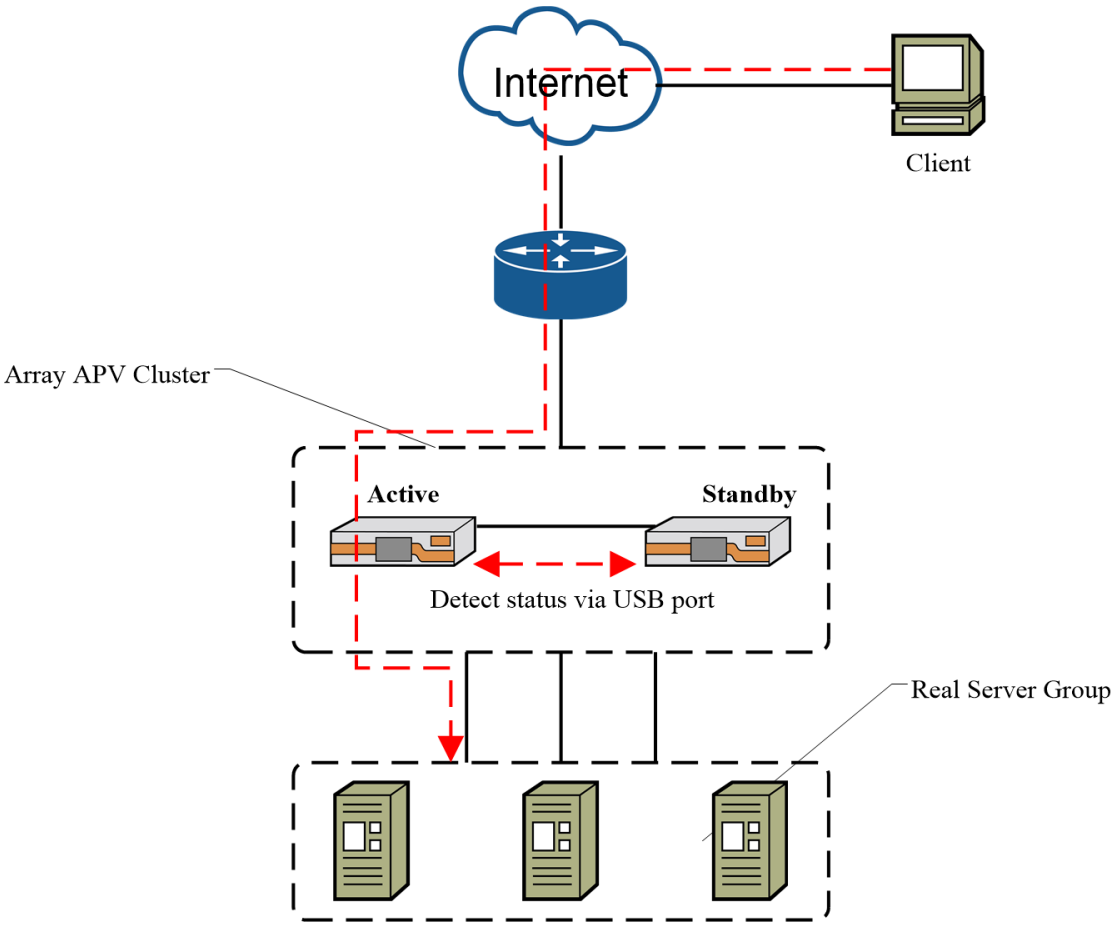


Figure 4–2 Clustering FFO Mode

4.2.2 Discreet Backup Mode

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

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

The following shows how the Discreet Backup mode works.

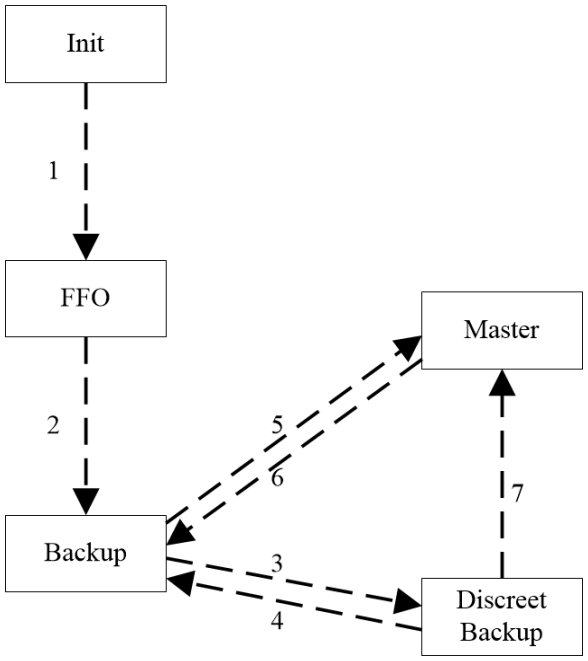


Figure 4–3 Discreet Backup Mode Working Mechanism

1. After turning on clustering, the device enters into Init state. Then, in order to check the health of the heartbeat cable, the Init device switches to FFO state.

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.

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

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.

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 does not receive the VRRP packet from the master.

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

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.

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

4.2.3 IPv6 Support for Clustering

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

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

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

4.3 Clustering Configuration

4.3.1 Clustering SLB VIPs

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

For information about SLB, please refer to Chapter 7 Server Load Balancing (SLB).

4.3.1.1 Active-Standby: Two Nodes

Configuration Guidelines

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

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

- APV1 is the current master, and handles SLB traffic for VIP.
- APV2 is the backup, and listens for advertisements from the master. It will resume master status if APV1 stops sending advertisements (i.e. APV1 fails).

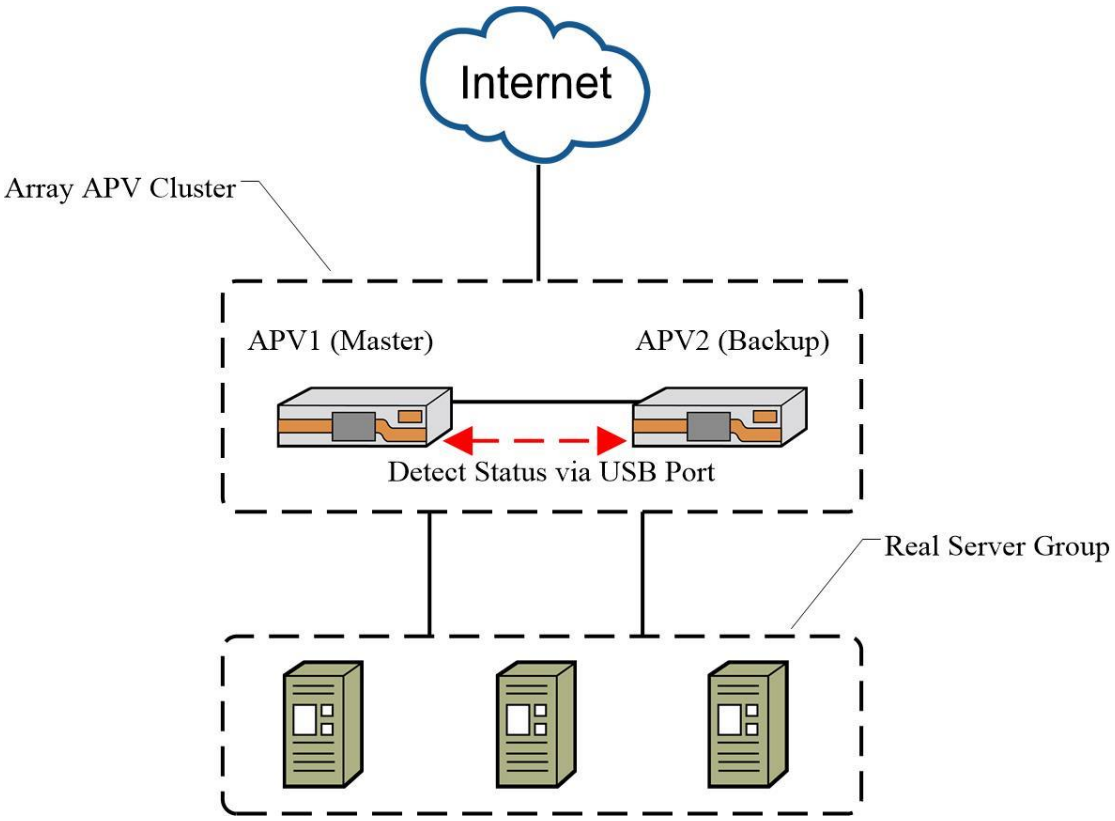


Figure 4–4 Active-Standby Two-Node Architecture

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

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

Operation	Command
	[synconfig_peer_name]
Enable the virtual cluster	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*
Enable fast failover feature	**cluster virtual ffo {on|off}** **cluster virtual ffo interface carrier loss timeout** <*interface_timeout*>

Configuration Example for Active-Standby SLB Clustering via CLI

Now let's start to configure APV1 and APV2:

1. Configure SLB for both APV1 and APV2.

```
APV1(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV1(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV1(config)#slb group method "group1" rr
APV1(config)#slb group member "group1" "server1" 1
APV1(config)#slb group member "group1" "server2" 1
APV1(config)#slb virtual http "vip1" 192.168.2.100 80
APV1(config)#slb policy default "vip1" "group1"
APV2(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV2(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV2(config)#slb group method "group1" rr
APV2(config)#slb group member "group1" "server1" 1
APV2(config)#slb group member "group1" "server2" 1
APV2(config)#slb virtual http "vip1" 192.168.2.100 80
APV2(config)#slb policy default "vip1" "group1"
```

Configure a virtual interface name.

```
APV1(config)#cluster virtual ifname "port1" 100
APV2(config)#cluster virtual ifname "port1" 100
```

Configure virtual cluster authentication.

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

```
APV1(config)#cluster virtual auth port1 100 0
APV2(config)#cluster virtual auth port1 100 0
```

Configure virtual cluster preemption.

Now we configure APV1 to preempt the VIP when the initial master returns online. For APV2, it will not preempt the VIP from the master node, but will take over if the master ceases operations.

```
APV1(config)#cluster virtual preempt port1 100 1
APV2(config)#cluster virtual preempt port1 100 0
```

Define the VIP by the "**cluster virtual vip**" command.

```
APV1(config)#cluster virtual vip "port1" 100 192.168.2.100
APV2(config)#cluster virtual vip "port1" 100 192.168.2.100
```

Define the priority.

Cluster priority determines which node becomes the master. The node with highest priority becomes the master. Since we want APV1 to always be master of the VIP, we will set its priority to 255. For APV2, 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:

```
APV1(config)#cluster virtual priority port1 100 255
APV2(config)#cluster virtual priority port1 100 100
```

Note: The state is the backup on APV2. This is expected since it is of lower priority than the master.

Turn on the clustering.

```
APV1(config)#cluster virtual on
APV2(config)#cluster virtual on
```

Turn on fast failover.

```
APV1(config)#cluster virtual ffo on
APV1(config)#cluster virtual ffo interface carrier loss timeout 1000
APV2(config)#cluster virtual ffo on
APV2(config)#cluster virtual ffo interface carrier loss timeout 1000
```

4.3.1.2 Active-Active: Two Nodes

Configuration Guidelines

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

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

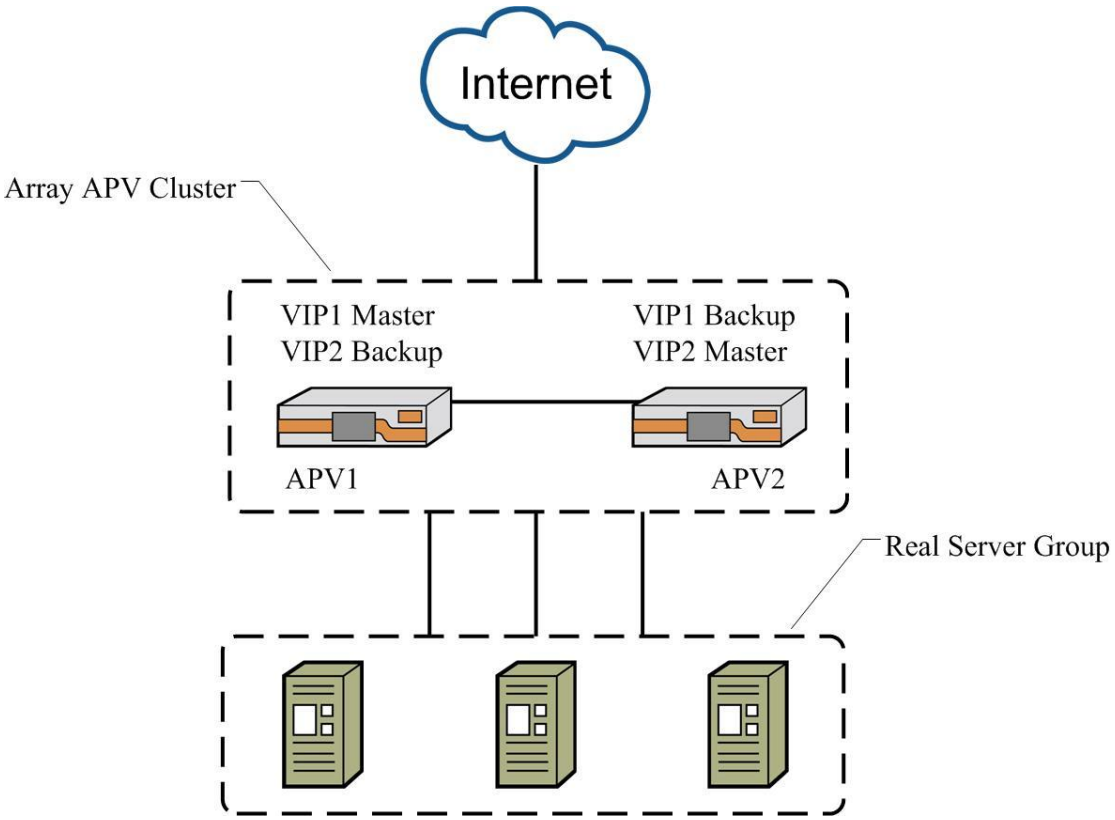


Figure 4–5 Active-Active Two-Node Architecture

In the above figure, APV1 is the master for VIP1 and the backup for VIP2 and APV2 is the master for VIP2 and the backup for VIP1.

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

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

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

Configuration Example for Active-Active SLB Clustering via CLI

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

1. Configure SLB for both APV1 and APV2.

```
APV1(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV1(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV1(config)#slb group method "group1" rr
APV1(config)#slb group member "group1" "server1" 1
APV1(config)#slb group member "group1" "server2" 1
APV1(config)#slb virtual http "vip1" 192.168.2.100 80
APV1(config)#slb virtual http "vip2" 192.168.2.101 80
APV1(config)#slb policy default "vip1" "group1"
APV1(config)#slb policy default "vip2" "group1"

APV2(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV2(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV2(config)#slb group method "group1" rr
APV2(config)#slb group member "group1" "server1" 1
APV2(config)#slb group member "group1" "server2" 1
APV2(config)#slb virtual http "vip1" 192.168.2.100 80
APV2(config)#slb virtual http "vip2" 192.168.2.101 80
APV2(config)#slb policy default "vip1" "group1"
APV2(config)#slb policy default "vip2" "group1"
```

Configure a virtual interface name.

```
APV1(config)#cluster virtual ifname "port1" 100
APV1(config)#cluster virtual ifname "port1" 101
APV2(config)#cluster virtual ifname "port1" 100
APV2(config)#cluster virtual ifname "port1" 101
```

Configure virtual cluster authentication.

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

```
APV1(config)#cluster virtual auth port1 100 0
APV1(config)#cluster virtual auth port1 101 0
APV2(config)#cluster virtual auth port1 100 0
APV2(config)#cluster virtual auth port1 101 0
```

Configure virtual cluster preemption.

```
APV1(config)#cluster virtual preempt port1 100 1
APV1(config)#cluster virtual preempt port1 101 0
APV2(config)#cluster virtual preempt port1 100 0
```

```
APV2(config)#cluster virtual preempt port1 101 1
```

Define the VIP by the "**cluster virtual vip**" command.

```
APV1(config)#cluster virtual vip "port1" 100 192.168.2.100
APV1(config)#cluster virtual vip "port1" 101 192.168.2.101
APV2(config)#cluster virtual vip "port1" 100 192.168.2.100
APV2(config)#cluster virtual vip "port1" 101 192.168.2.101
```

Define the priority.

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

```
APV1(config)#cluster virtual priority port1 100 255
APV1(config)#cluster virtual priority port1 101 100
APV2(config)#cluster virtual priority port1 100 100
APV2(config)#cluster virtual priority port1 101 255
```

Turn on the clustering.

```
APV1(config)#cluster virtual on
APV2(config)#cluster virtual on
```

4.3.2 Clustering Inside Interfaces

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

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

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

4.3.2.1 Active-Standby (One VIP)

Configuration Guidelines

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

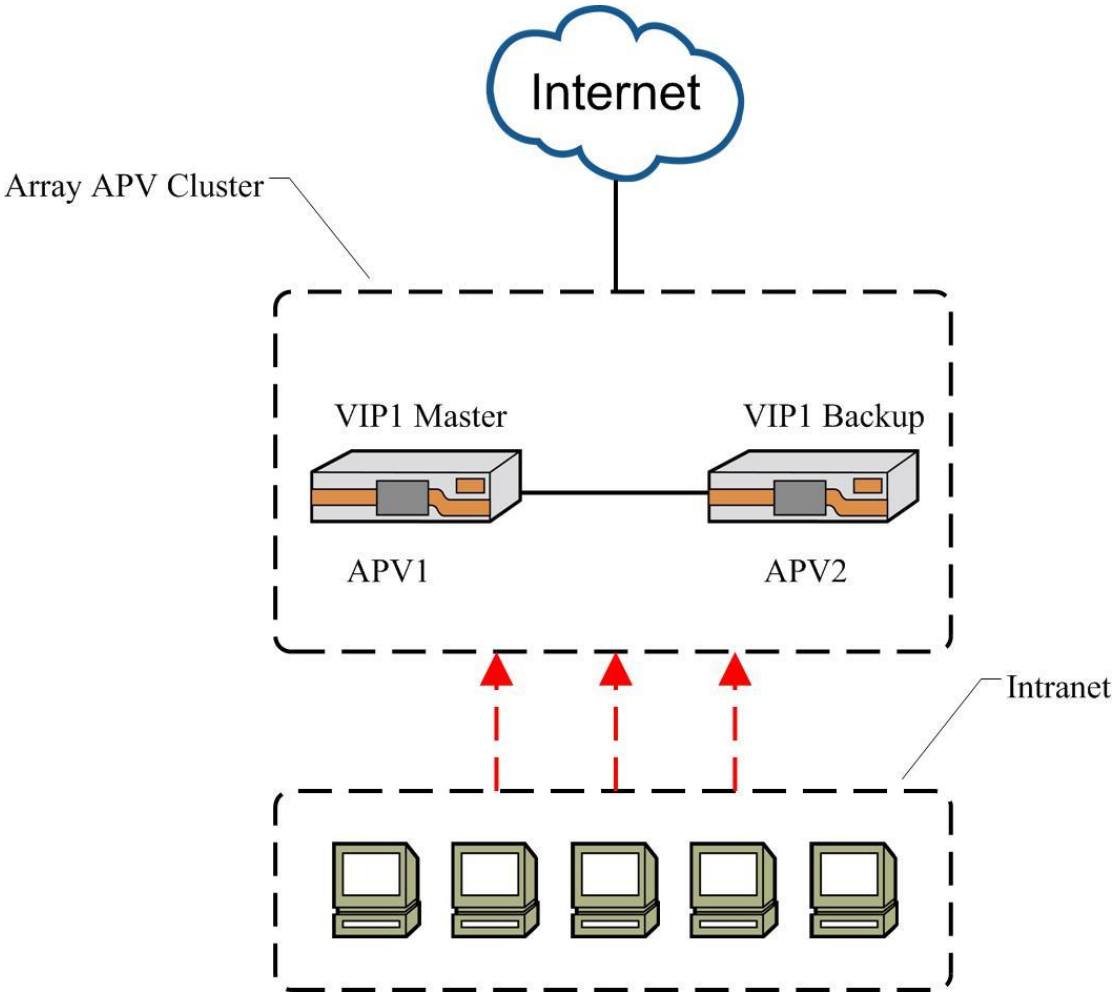


Figure 4–6 Inside Interface Active-Standby Mode

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

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

Configuration Example for Active-Standby Clustering Inside Interface via CLI

1. Configure a virtual interface and its cluster ID.

```
APV1(config)#cluster virtual ifname "port2" 100
APV2(config)#cluster virtual ifname "port2" 100
```

Define the VIP by the "**cluster virtual vip**" command.

```
APV1(config)#cluster virtual vip "port2" 100 192.168.1.3
```

```
APV2(config)#cluster virtual vip "port2" 100 192.168.1.3
```

Define the priority.

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

```
APV1(config)#cluster virtual priority port2 100 255
APV2(config)#cluster virtual priority port2 100 100
```

Turn on the clustering.

```
APV1(config)#cluster virtual on
APV2(config)#cluster virtual on
```

4.3.2.2 Active-Active (Two VIPs)

Configuration Guidelines

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

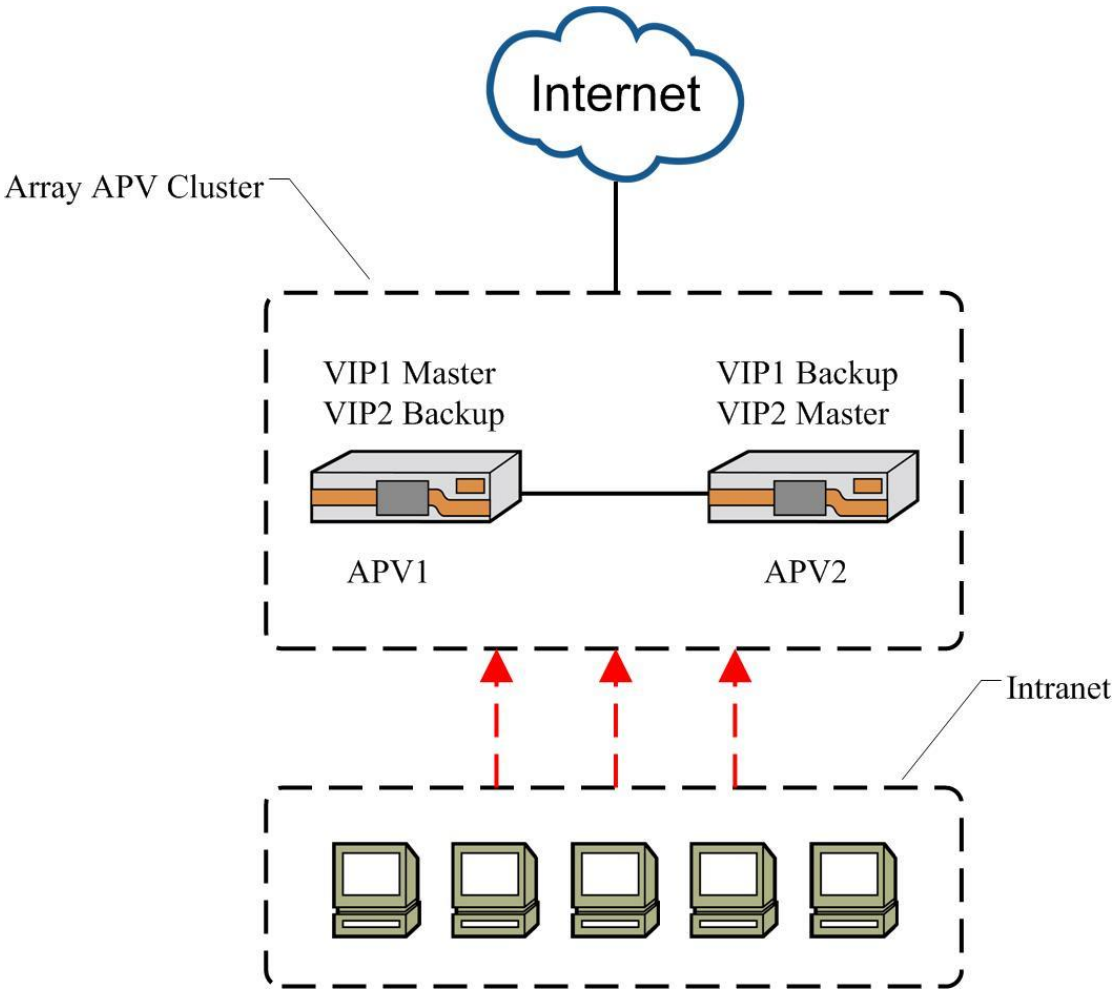


Figure 4–7 Inside Interface Active- Active Mode

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

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

Configuration Example for Active-Active Clustering Inside Interface via CLI

We proceed along these lines by executing the following:

1. Configure a virtual interface and its cluster ID.

```
APV1(config)#cluster virtual ifname "port2" 100
APV1(config)#cluster virtual ifname "port2" 101
APV2(config)#cluster virtual ifname "port2" 100
APV2(config)#cluster virtual ifname "port2" 101
```

Define the VIP by the "**cluster virtual vip**" command.

```
APV1(config)#cluster virtual vip "port2" 100 192.168.1.3
APV1(config)#cluster virtual vip "port2" 101 192.168.1.4
APV2(config)#cluster virtual vip "port2" 100 192.168.1.3
APV2(config)#cluster virtual vip "port2" 101 192.168.1.4
```

Define the priority.

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

```
APV1(config)#cluster virtual priority port2 100 255
APV1(config)#cluster virtual priority port2 101 100
APV2(config)#cluster virtual priority port2 100 100
APV2(config)#cluster virtual priority port2 101 255
```

Turn on the clustering.

```
APV1(config)#cluster virtual on
APV2(config)#cluster virtual on
```

4.3.2.3 Active-Standby: Two Nodes

Configuration Guidelines

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

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

- APV1 is the current master, and handles SLB traffic for VIP.
- APV2 is the backup, and listens for advertisements from the master. It will resume master status if APV1 stops sending advertisements (i.e. APV1 fails).

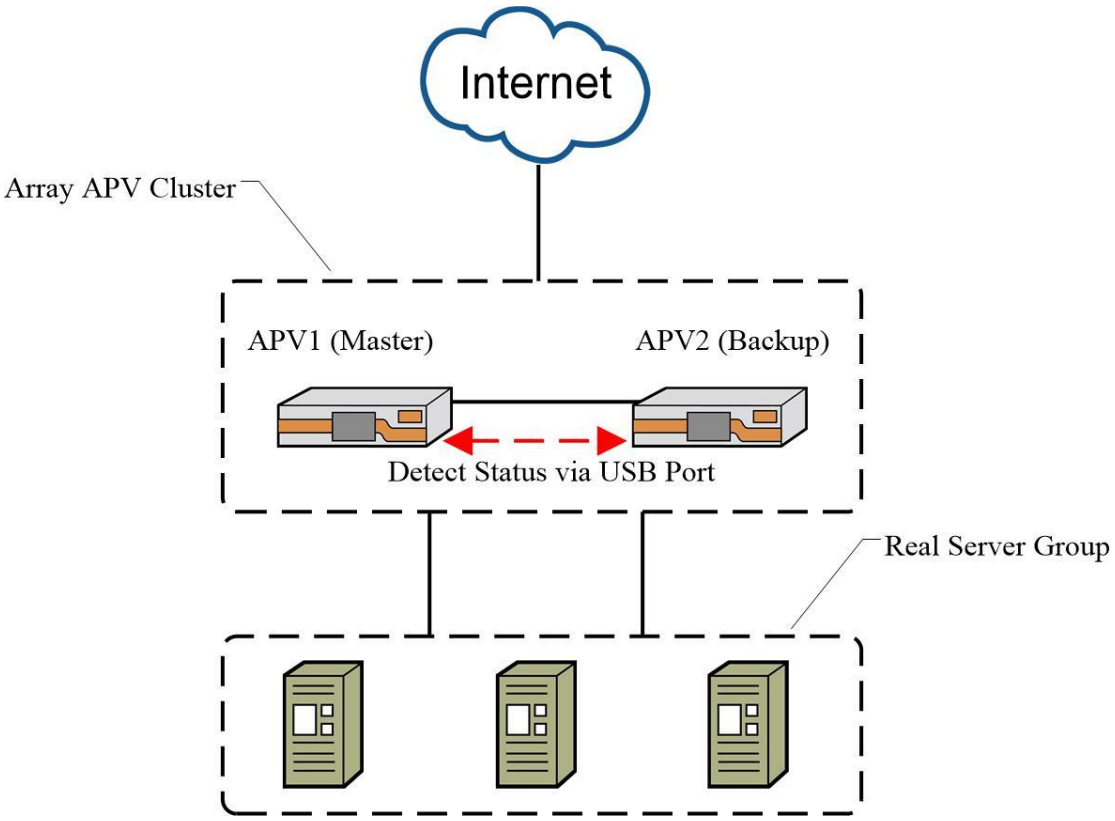


Figure 4–8 Active-Standby Two-Node Architecture

Table 4–5 General Settings of Active-Standby Two-Node Clustering

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

Operation	Command
	[synconfig_peer_name]
Enable the virtual cluster	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]*
Enable fast failover feature	**cluster virtual ffo {on|off}** **cluster virtual ffo interface carrier loss timeout** <*interface_timeout*>

Configuration Example for Active-Standby SLB Clustering via CLI

Now let's start to configure APV1 and APV2:

1. Configure SLB for both APV1 and APV2.

```
APV1(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV1(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV1(config)#slb group method "group1" rr
APV1(config)#slb group member "group1" "server1" 1
APV1(config)#slb group member "group1" "server2" 1
APV1(config)#slb virtual http "vip1" 192.168.2.100 80
APV1(config)#slb policy default "vip1" "group1"
APV2(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV2(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV2(config)#slb group method "group1" rr
APV2(config)#slb group member "group1" "server1" 1
APV2(config)#slb group member "group1" "server2" 1
APV2(config)#slb virtual http "vip1" 192.168.2.100 80
APV2(config)#slb policy default "vip1" "group1"
```

Configure a virtual interface name.

```
APV1(config)#cluster virtual ifname "port1" 100
APV2(config)#cluster virtual ifname "port1" 100
```

Configure virtual cluster authentication.

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

```
APV1(config)#cluster virtual auth port1 100 0
APV2(config)#cluster virtual auth port1 100 0
```

Configure virtual cluster preemption.

Now we configure APV1 to preempt the VIP when the initial master returns online. For APV2, it will not preempt the VIP from the master node, but will take over if the master ceases operations.

```
APV1(config)#cluster virtual preempt port1 100 1
APV2(config)#cluster virtual preempt port1 100 0
```

Define the VIP by the "**cluster virtual vip**" command.

```
APV1(config)#cluster virtual vip "port1" 100 192.168.2.100
APV2(config)#cluster virtual vip "port1" 100 192.168.2.100
```

Define the priority.

Cluster priority determines which node becomes the master. The node with highest priority becomes the master. Since we want APV1 to always be master of the VIP, we will set its priority to 255. For APV2, 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:

```
APV1(config)#cluster virtual priority port1 100 255
APV2(config)#cluster virtual priority port1 100 100
```

Note: The state is the backup on APV2. This is expected since it is of lower priority than the master.

Turn on the clustering.

```
APV1(config)#cluster virtual on
APV2(config)#cluster virtual on
```

Turn on fast failover.

```
APV1(config)#cluster virtual ffo on
APV1(config)#cluster virtual ffo interface carrier loss timeout 1000
APV2(config)#cluster virtual ffo on
APV2(config)#cluster virtual ffo interface carrier loss timeout 1000
```

4.3.2.4 Active-Active: Two Nodes

Configuration Guidelines

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

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

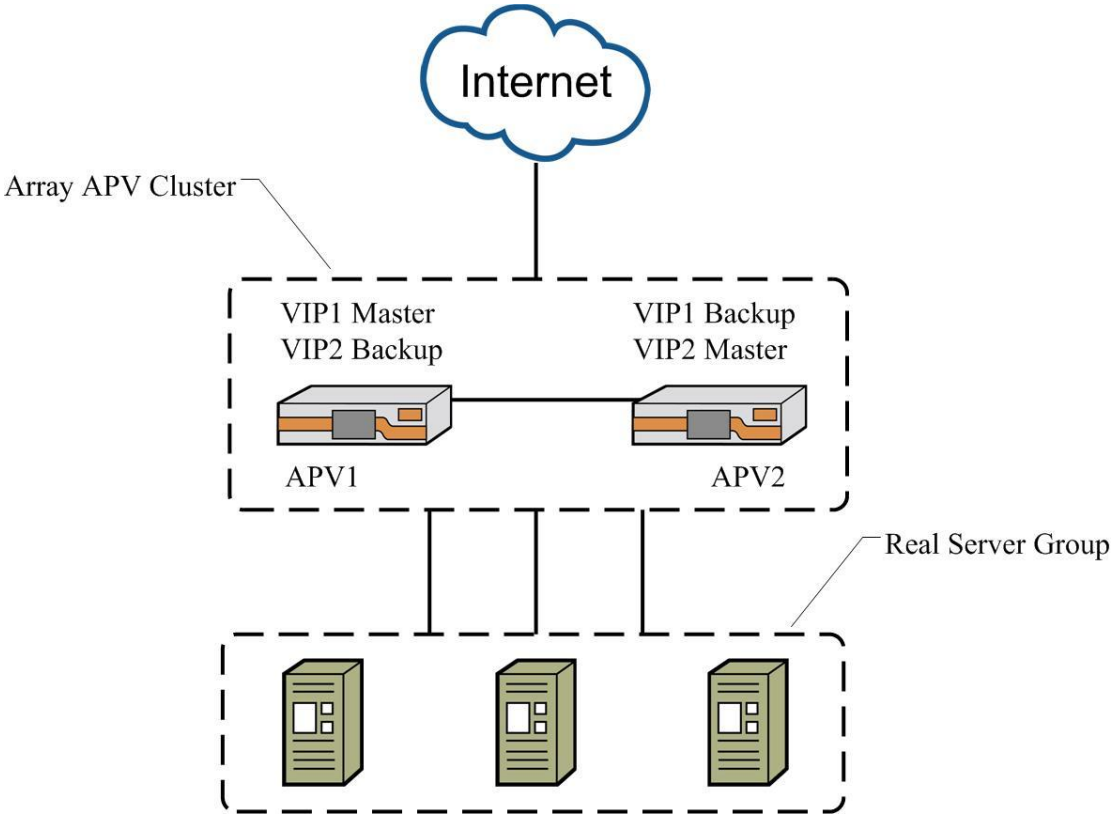


Figure 4–9 Active-Active Two-Node Architecture

In the above figure, APV1 is the master for VIP1 and the backup for VIP2 and APV2 is the master for VIP2 and the backup for VIP1.

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

Table 4–6 General Settings of Active-Active Two-Node Clustering

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

Configuration Example for Active-Active SLB Clustering via CLI

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

1. Configure SLB for both APV1 and APV2.

```
APV1(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV1(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV1(config)#slb group method "group1" rr
APV1(config)#slb group member "group1" "server1" 1
APV1(config)#slb group member "group1" "server2" 1
APV1(config)#slb virtual http "vip1" 192.168.2.100 80
APV1(config)#slb virtual http "vip2" 192.168.2.101 80
APV1(config)#slb policy default "vip1" "group1"
APV1(config)#slb policy default "vip2" "group1"

APV2(config)#slb real http "server1" 192.168.1.50 80 1000 tcp 1 1
APV2(config)#slb real http "server2" 192.168.1.51 80 1000 tcp 1 1
APV2(config)#slb group method "group1" rr
APV2(config)#slb group member "group1" "server1" 1
APV2(config)#slb group member "group1" "server2" 1
APV2(config)#slb virtual http "vip1" 192.168.2.100 80
APV2(config)#slb virtual http "vip2" 192.168.2.101 80
APV2(config)#slb policy default "vip1" "group1"
APV2(config)#slb policy default "vip2" "group1"
```

Configure a virtual interface name.

```
APV1(config)#cluster virtual ifname "port1" 100
APV1(config)#cluster virtual ifname "port1" 101
APV2(config)#cluster virtual ifname "port1" 100
APV2(config)#cluster virtual ifname "port1" 101
```

Configure virtual cluster authentication.

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

```
APV1(config)#cluster virtual auth port1 100 0
APV1(config)#cluster virtual auth port1 101 0
APV2(config)#cluster virtual auth port1 100 0
APV2(config)#cluster virtual auth port1 101 0
```

Configure virtual cluster preemption.

```
APV1(config)#cluster virtual preempt port1 100 1
APV1(config)#cluster virtual preempt port1 101 0
APV2(config)#cluster virtual preempt port1 100 0
```

```
APV2(config)#cluster virtual preempt port1 101 1
```

Define the VIP by the "**cluster virtual vip**" command.

```
APV1(config)#cluster virtual vip "port1" 100 192.168.2.100
APV1(config)#cluster virtual vip "port1" 101 192.168.2.101
APV2(config)#cluster virtual vip "port1" 100 192.168.2.100
APV2(config)#cluster virtual vip "port1" 101 192.168.2.101
```

Define the priority.

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

```
APV1(config)#cluster virtual priority port1 100 255
APV1(config)#cluster virtual priority port1 101 100
APV2(config)#cluster virtual priority port1 100 100
APV2(config)#cluster virtual priority port1 101 255
```

Turn on the clustering.

```
APV1(config)#cluster virtual on
APV2(config)#cluster virtual on
```

Chapter 5 High Availability (HA)

5.1 Overview

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

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

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

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

5.2 HA Basics

5.2.1 HA Domain and Unit

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

5.2.2 Floating IP Group and Floating MAC

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

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

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

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

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

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

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

The HA function supports the floating MAC function. When a group switchover is triggered, the floating MAC address will be switched to the interface of the new unit on which the group status is "Active". In this way, after group status switches, the clients will not be aware that the appliance that provides the application services has been changed, because the MAC addresses of the appliances that provide application services before and after group status switch remain unchanged. Different floating IP addresses (including floating IP range) in the same floating IP group can be bound with the same floating MAC address, but the floating MAC addresses between floating IP groups cannot overlap. By default, the floating MAC function is disabled. Before this function is enabled, the HA function must be first disabled by executing the command "**ha off**".

When defining a floating IP address, the administrator can selectively bind a floating MAC address to it. After floating MAC address binding, the MAC address will be added to the MAC address list of the physical interface on which the floating IP group resides. The floating MAC address will be used as the source MAC address in the packets sending from the floating IP address on the interface. Other packets sent from the interface will use the interface MAC address. The MAC addresses of the interface, including the floating MAC address, can be viewed via the "**show interface**" command.

5.2.3 Group Failover Mode

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

When a floating IP group is enabled on multiple units:

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

5.2.4 HA Deployment Scenarios

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

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

5.3 Reliable Communication Links

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

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

The FFO link must work together with the primary or secondary network link. The FFO link is established by directly connecting two HA units’ FFO ports through a dedicated FFO cable. Therefore, it can only be used for the Active/Active and Active/Standby deployment scenarios

with two units. By default, the FFO link is disabled. The main functions of the FFO link are as follows:

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

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

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

- Heartbeat packet transmission: The local unit can send the heartbeat packets to its peer units through the primary link to detect the peer units' status.
- Bootup Synconfig: With both Bootup Synconfig and the HA function enabled, the local unit can synchronize the configurations saved by the "**write memory**" command from the peer units.
- Runtime Synconfig: With Runtime Synconfig enabled, when the whitelist configurations of the local unit are modified, the local unit can synchronize the modifications to the peer units via the primary link. When the blacklist configurations of the local unit are modified, the local unit will not synchronize the modifications to the peer units.

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

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

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

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

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

Table 5–1 Differences and Similarities Among the HA Links

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

5.4 Failover Rules

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

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

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

HA supports the following types of health checks:

1. Predefined health check: HA has predefined network connectivity check mechanism (PORT_1~PORT_32) to detect network interface faults and network disconnections between nodes. By default, when a network interface is faulty, the system will perform Group_Failover. Administrators can modify a failover action associated with the predefined health check condition.

Note:

- Only when the network connections of all interfaces in a bond interface become down, will the “Group_Failover” action be taken for the floating IP group to which the IP addresses of the bond interface belong.
- For the units who provide failover support for the same floating IP group in the HA domain, the failover rules of the floating IP group configured on the units should be exactly the same, including the name of the health check condition in failover rules.
- For the units who provide failover support for “Unit_Failover” in the HA domain, the failover rules of the “Unit_Failover” configured on the units should be exactly the same, including the name of the health check condition in failover rules.

System health check: HA supports the reuse of the system health check as failover conditions. Administrator can define the following health check conditions for HA failover.

- Hardware:
  - CPU overheat health check condition
  - SSL card health check condition
- Software:
  - CPU utilization health check condition
- Network condition:
  - Gateway health check condition

In some complex application environments, more complicated failover rules are required. For example, theoretically, in the environment with a bond interface deployed, only when the network connections of all interfaces in the bond interface become down will failover action be taken. However, in practical application, it is required to take failover action as long as the network connection of one interface becomes down. To meet this kind of complicated applications, HA

further introduces the concept of health check condition group (vcondition). A vcondition comprises multiple health check sub-conditions. A sub-condition can be a real health check condition or another vcondition, which further comprises sub-conditions. The logical relationship among multiple sub-conditions can be either “AND” or “OR”. To apply vcondition to the above application, administrators can first define health check conditions for each of the network interfaces in a bond interface, and combine these conditions into a vcondition by setting the logical relationship to “OR”. Then, associate the vcondition with some failover actions.

Self-defined health check script: The system supports the import of self-defined health check scripts from an SCP or TFTP server. Self-defined health check scripts take effect only after signature verification. To use this function, please contact Customer Support.

5.5 Configuration Synchronization

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

5.5.1 Bootup Synconfig

Bootup Synconfig is to synchronize the configurations of the peer unit to the local unit via the primary link after the local unit logs into the HA domain. If the FFO link is enabled, the local unit will synchronize HA unit and link configurations from the peer unit via the FFO link. When the configuration synchronization secure mode is enabled using the command “**synconfig secure on**”, before using the Bootup Synconfig function, the administrator needs to:

- Make sure the same Challenge Code is configured on both local unit and peer unit using the “**synconfig challenge**” command.
- Configure synchronization peers on both peer unit and local unit using the “**synconfig peer** *<peer_name> <peer_ip>*” command. In addition, the value of the “peer_name” parameter should be identical with the value of the “unit_name” parameter specified in the “**ha unit** *<unit_name> <ip_address> [port]*” command.

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

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

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

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

Execute the "**ha link network on**" command to enable the network links.

Execute the "**ha on**" command to enable the HA function.

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

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

Note:

1. In Bootup Synconfig, only the configurations saved by executing the command "**write memory**" can be synchronized.
2. In the HA domain login process via the network link, the local unit will synchronize the system time of the peer unit.

5.5.2 Runtime Synconfig

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

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

5.5.3 Runtime Synconfig

Incremental Synconfig is to synchronize the incremental configurations on the local unit to the specified peer unit or all peer units.

5.6 Stateful Session Failover (SSF)

The Stateful Session Failover (SSF) function supports the connection synchronization between at most three HA units. With SSF enabled, the information about the TCP and UDP connections established on the "Active" floating IP group will be updated to all "Standby" floating IP groups in real time. Once any failover action is taken, all the existing TCP and UDP connections will not be interrupted because the connection information has been updated to the new "Active" unit by the SSF function. However, if the SSF function is disabled, the existing TCP and UDP connections will be interrupted.

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

Note:

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

5.7 HA Logging

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

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

HA internal information is recorded in HA log, and the changes of floating IP groups and units is recorded in the system log.

5.8 Configuration Examples

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

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

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

5.8.1 Scenario 1: Active/Standby

5.8.1.1 Configuration Objectives

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

- The HA domain contains two HA units, each of which is enabled with the same floating IP group.

- The Floating IP group contains the VIP addresses of two application services.
- Unit 1 provides application services, while unit 2 provides backup for such services.
- Fast failover is carried out through the FFO link.

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

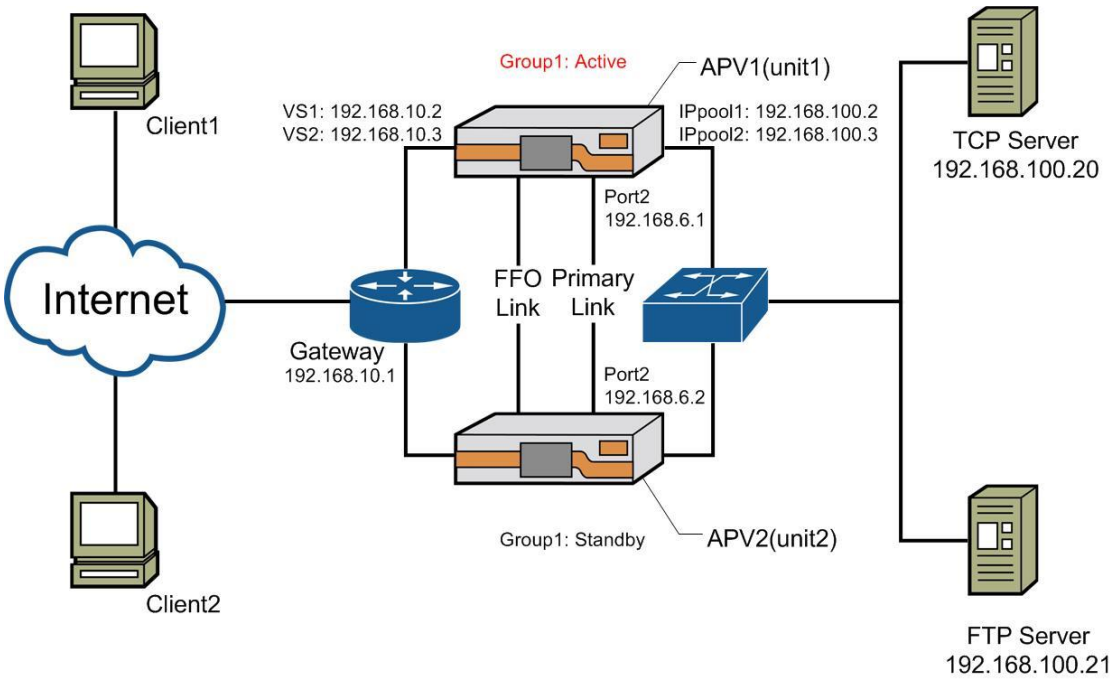


Figure 5–1 Active/Standby Deployment Scenario

5.8.1.2 Configuration Examples

➢ **APV1:**

1. Execute the following commands to complete SLB configurations:

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

Execute the following commands to configure HA units, synchronization peers and links:

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

Execute the following commands to configure floating IP group:

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

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

```
AN(config)#monitor network gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor network gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor system cpu utilization 90 5000 3 3
AN(config)#monitor vcondition name vcondition1 VCONDITION_1 AND
AN(config)#monitor vcondition member vcondition1 GATEWAY_1
AN(config)#monitor vcondition member vcondition1 CPU_UTIL
```

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

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

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

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

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

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

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

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

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

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

- **APV2：**

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

1. Execute the following commands to configure synchronization peers:

```
AN(config)#synconfig peer "unit1" 192.168.6.1
AN(config)#synconfig peer "unit2" 192.168.6.2
```

Execute the following commands to enable the network link, the FFO function and the Bootup Synconfig mode:

```
AN(config)#ha link network on
AN(config)#ha link ffo on
AN(config)#ha synconfig bootup on
```

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

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

Execute the following command to enable the HA function:

```
AN(config)#ha on
```

Once the HA function is enabled, unit2 (APV2) will join the HA domain. After unit2 joins the HA domain, it first synchronizes configuration information about HA units, FFO link and network links through the FFO link and then synchronizes other configurations through the primary link from unit1 (APV1).

5.8.2 Scenario 2: Active/Active

5.8.2.1 Configuration Objectives

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

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

- Fast failover is carried out through the FFO link.

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

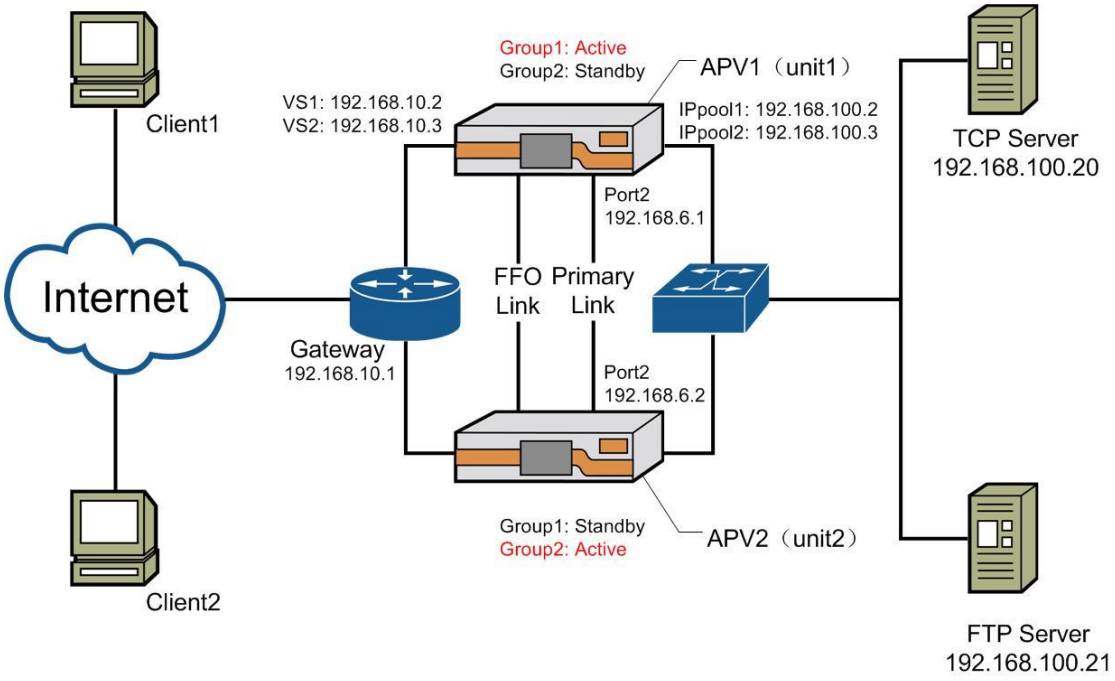


Figure 5–2 Active/Active Deployment Scenario

5.8.2.2 Configuration Examples

➢ **APV1:**

1. Execute the following commands to complete SLB configurations:

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

Execute the following commands to configure HA units, synchronization peers and links:

```
AN(config)#ha unit "unit1" 192.168.6.1 65521
```

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

Execute the following commands to configure floating IP groups:

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

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

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

```
AN(config)#monitor network gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor network gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor system cpu utilization 90 5000 3 3
AN(config)#monitor vcondition name vcondition1 VCONDITION_1 AND
AN(config)#monitor vcondition member vcondition1 GATEWAY_1
AN(config)#monitor vcondition member vcondition1 CPU_UTIL
```

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

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

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

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

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

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

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

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

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

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

- **APV2:**

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

1. Execute the following commands to configure synchronization peers:

```
AN(config)#synconfig peer "unit1" 192.168.6.1
AN(config)#synconfig peer "unit2" 192.168.6.2
```

Execute the following commands to enable the network link, the FFO function and the Bootup Synconfig mode:

```
AN(config)# ha link network on
AN(config)#ha link ffo on
AN(config)#ha synconfig bootup on
```

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

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

Execute the following command to enable the HA function:

```
AN(config)#ha on
```

Once the HA function is enabled, unit2 (APV2) will join the HA domain. After unit2 joins the HA domain, it first synchronizes configuration information about HA units, FFO link and network links through the FFO link and then synchronizes other configurations through the primary link from unit1 (APV1).

5.8.3 Scenario 3: N+1

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

5.8.3.1 Configuration Objectives

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

- The HA domain contains four HA units and provides three floating IP groups.

- Each floating IP group contains the VIP address of a virtual service.
- Unit1 to unit3 provide the virtual services of group1 to group3 respectively, while unit4 provides backup for unit1 to unit3.

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

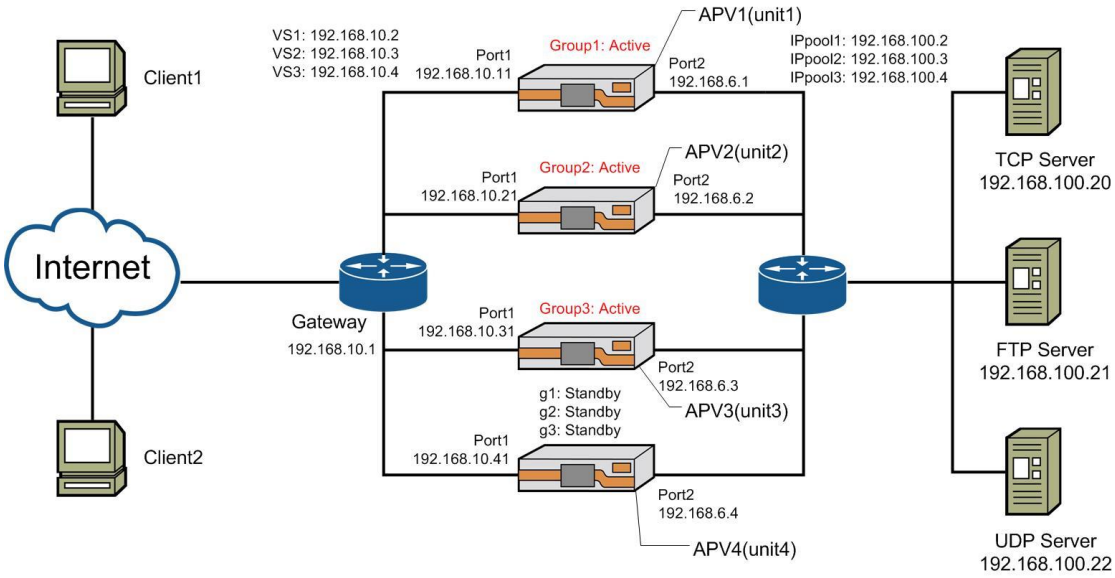


Figure 5–3 N+1 Deployment Scenario

5.8.3.2 Configuration Examples

➢ **APV1:**

1. Execute the following commands to complete SLB configurations:

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

```
AN(config)#ip pool p3 192.168.100.4 192.168.100.4
AN(config)#slb proxyip group slb_g1 p1
AN(config)#slb proxyip group slb_g2 p2
AN(config)#slb proxyip group slb_g3 p3
```

Execute the following commands to configure HA units, synchronization peers and links:

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

Execute the following commands to configure floating IP groups:

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

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

AN(config)#ha group id 3
AN(config)#ha group fip 3 192.168.10.4 port1
AN(config)#ha group fip 3 192.168.100.4 port3
```

```
AN(config)#ha group priority unit1 3 100
AN(config)#ha group priority unit2 3 50
AN(config)#ha group priority unit3 3 200
AN(config)#ha group priority unit4 3 150
AN(config)#ha group preempt on 3
AN(config)#ha group enable 3
```

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

```
AN(config)#monitor network gateway unit1 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor network gateway unit2 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor network gateway unit3 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor network gateway unit4 192.168.10.1 GATEWAY_1 1000 3 3
AN(config)#monitor system cpu utilization 90 5000 3 3
AN(config)#monitor vcondition name vcondition1 VCONDITION_1 AND
AN(config)#monitor vcondition member vcondition1 GATEWAY_1
AN(config)#monitor vcondition member vcondition1 CPU_UTIL
```

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

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

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

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

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

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

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

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

- **APV2:**

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

```
AN(config)#ha unit "unit1" 192.168.6.1 65521
AN(config)#ha unit "unit2" 192.168.6.2 65521
AN(config)#ha unit "unit3" 192.168.6.3 65521
AN(config)#ha unit "unit4" 192.168.6.4 65521
AN(config)#synconfig peer "unit1" 192.168.6.1
AN(config)#synconfig peer "unit2" 192.168.6.2
AN(config)#synconfig peer "unit3" 192.168.6.3
AN(config)#synconfig peer "unit4" 192.168.6.4
```

```
AN(config)#ha link network secondary unit1 1 192.168.10.11
AN(config)#ha link network secondary unit2 1 192.168.10.21
AN(config)#ha link network secondary unit3 1 192.168.10.31
AN(config)#ha link network secondary unit4 1 192.168.10.41
AN(config)#ha link network on
```

Execute the following command to enable the Bootup Synconfig mode:

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

Execute the following command to enable the HA function:

```
AN(config)#ha on
```

Once the HA function is enabled, unit2 (APV2) will join the HA domain and start to synchronize configuration information from unit1 (APV1).

- **APV3:**

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

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

Execute the following command to enable the Bootup Synconfig mode:

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

Execute the following command to enable the HA function:

```
AN(config)#ha on
```

Once the HA function is enabled, unit3 (APV3) will join the HA domain and start to synchronize configuration information from unit1 (APV1).

- **APV4:**

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

```
AN(config)#ha unit "unit1" 192.168.6.1 65521
AN(config)#ha unit "unit2" 192.168.6.2 65521
```

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

Execute the following command to enable the Bootup Synconfig mode:

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

Execute the following command to enable the HA function:

```
AN(config)#ha on
```

Once the HA function is enabled, unit4 (APV4) will join the HA domain and start to synchronize configuration information from unit1 (APV1).

Chapter 6 Single System Image (SSI)

6.1 Overview

A single system image (SSI) domain is a group of APV appliances that appears to be one single system. Logically, the multiple appliances in SSI deployment are regarded as one appliance.

The SSI feature allows the administrators to provide one virtual service by deploying multiple parallel APV appliances, to meet higher performance demand.

6.2 SSI Basics

SSI has the same architecture as HA, and therefore has the same basic concepts including domain, unit, floating IP group and communication link.

Note: The SSI feature and HA feature cannot be deployed at the same time.

6.3 Understanding SSI

The deployment of SSI requires switches with high bandwidth and LACP support. SSI supports two-arm server load balance.

Note: The SSI feature only supports SLB reverse proxy mode

As shown in SSI Work Flow, the data flow in SSI deployment is:

1. The data packets sent by the client arrive at switch1.
2. The switch1 forwards the data packets to an APV appliance (because the APV appliances in the SSI domain have the same virtual service IP address and floating MAC configuration, the data packets will be sent to certain APV appliance according to certain algorithm of the switch, hash based on the source IP and port, for example).
3. The APV appliance forwards the data packets to the real servers through switch2.
4. The real server sends the response data packets to switch2.
5. The switch2 sends the data packets to the APV appliance (the data packets can be returned to the APV appliance originally performed the processing, because the APV appliances have different IP pools for proxy IP).
6. The APV appliance sends the data packets to the client.

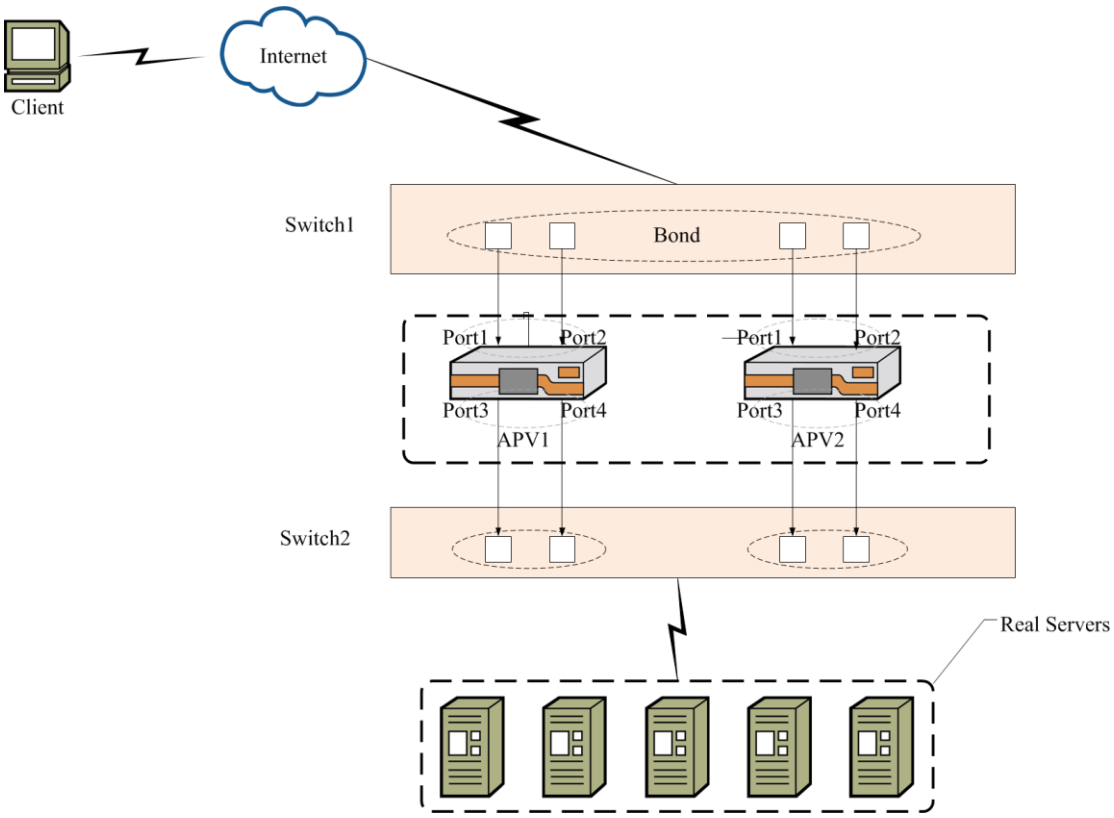


Figure 6–1 SSI Work Flow

As shown in the above picture, the following configurations need particular attention:

- Add the switch1 ports connecting with the APV inbound ports in a bond.
- Add the inbound ports on every APV appliance into a bond.
- Add the switch2 (intranet switch) ports connecting with the APV1 outbound ports in a bond.
- Add the switch2 ports connecting with the APV2 outbound ports in a bond.
- Add the outbound ports of each APV appliance in a bond.
- Assign different IP pools to each APV as the proxy IP.

Note: The SSI feature only supports SLB reverse proxy mode. The inbound ports of the SSI units can only serve as service port instead of management port, that is they cannot work for SSH link establishment, WebUI link establishment or system upgrade. If separate management for each unit is required, please assign unique management port to each unit.

6.4 Switch Failover

The SSI switch failover function provides standby switch for the switch between the client and the SSI domain, and the switch failover will happen when the current switch fail to provide service. When the connection between the APV appliance and the switch is abnormal, or certain condition defined by the administrator is met, switch failover will take place according to the defined rule.

The failover support for the switch between the client and the SSI domain provided by the switch failover function allows the virtual service can be switched by the SSI units to the standby switch, when the current switch undergoes electricity or connection interruption, thus guaranteeing the service continuance.

The configuration required by the SSI switch failover function is as follows:

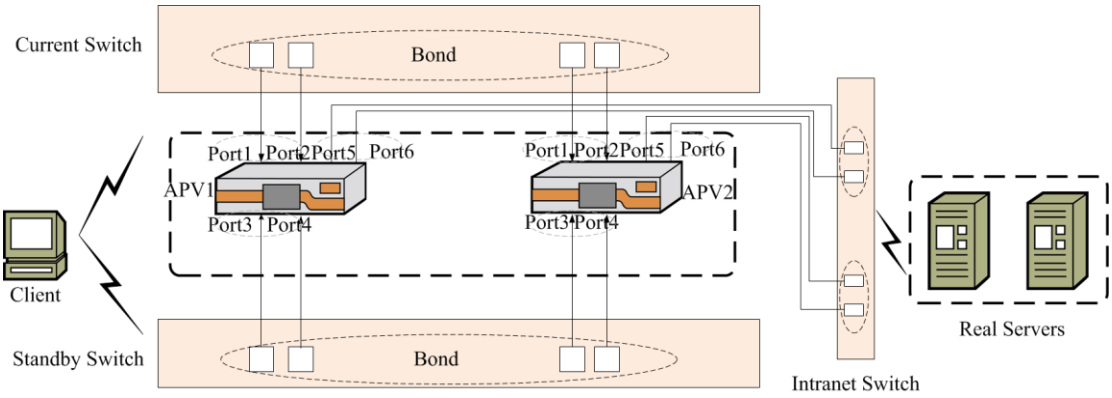


Figure 6–2 Switch Failover Work Flow

The following configurations of switch failover need particular attention:

- The current switch and the standby unit have the same port configuration, including bond configuration, LACP protocol enabling and flow distribution setting.
- Add the APV ports connecting with the switches into two floating IP groups, that is, add port1 and port2 of APV1 and APV2 into group1, and add port3 and port4 of APV1 and APV2 into group2.

6.5 SSI Configuration

6.5.1 SSI Configuration for Two-Arm SLB

To achieve the configuration effect in SSI Work Flow, the following configuration steps are required:

- Switch configuration
- Bond and interface IP configuration, and heartbeat interface configuration to include the APV appliances into one broadcast domain
- SLB configuration
- SSI configuration

Note: The SSI function requires that configuration on each APV appliance except proxy IP be identical.

The detailed configuration is as follows:

- Switch configuration
  - Connect port1 and port2 of APV1 to switch1, and port3 and port4 to switch2.
  - Connect port1 and port2 of APV2 to switch1, and port3 and port4 to switch2.
  - Add the switch1 ports connecting with port1 and port2 of APV1 and APV2 into a bond.
  - Enable the LACP protocol on the bond of switch1.
  - Configure flow distribution (for example, hash based on the source IP and port) on the bond of switch1.
  - Add the switch2 ports connecting with por3 and port4 of APV1 into a bond.
  - Add the switch2 ports connecting with por3 and port4 of APV2 into a bond.
- IP configuration of bond and interface on the APV appliances

```
#APV1
AN(config)#bond interface bond1 port1
AN(config)#bond interface bond1 port2
AN(config)#bond interface bond2 port3
AN(config)#bond interface bond2 port4
AN(config)#ip address bond1 100.8.1.1 255.255.255.0
AN(config)#ip address bond2 192.168.1.1 255.255.255.0
#heartbeat configuration for SSI domain
AN(config)#ip address port5 172.16.1.1 255.255.255.0

#APV2
AN(config)#bond interface bond1 port1
AN(config)#bond interface bond1 port2
AN(config)#bond interface bond2 port3
AN(config)#bond interface bond2 port4
AN(config)#ip address bond1 100.8.1.1 255.255.255.0
AN(config)#ip address bond2 192.168.1.2 255.255.255.0
#heartbeat configuration for SSI domain
AN(config)#ip address port5 172.16.1.2 255.255.255.0
```

- SLB configuration on the APV appliances

```
#APV1
AN(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
AN(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
AN(config)#slb group method "slb_g1" rr
AN(config)#slb group member "slb_g1" "r1" 1 0
AN(config)#slb group member "slb_g1" "r2" 1 0
AN(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
AN(config)#slb policy default "v1" "slb_g1"
AN(config)#ip pool p1 192.168.1.3 192.168.1.3
```

```
AN(config)#slb proxyip group slb_g1 p1

#APV2
AN(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
AN(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
AN(config)#slb group method "slb_g1" rr
AN(config)#slb group member "slb_g1" "r1" 1 0
AN(config)#slb group member "slb_g1" "r2" 1 0
AN(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
AN(config)#slb policy default "v1" "slb_g1"
AN(config)#ip pool p1 192.168.1.4 192.168.1.4
AN(config)#slb proxyip group slb_g1 p1
```

- SSI configuration on the APV appliances

```
#APV1 & APV2
AN(config)#ha ssi on
AN(config)#ha unit APV_1 172.16.1.1 65521
AN(config)#ha unit APV_2 172.16.1.2 65521

AN(config)#ha group id 1
AN(config)#ha group port 1 bond1 port1
AN(config)#ha group port 1 bond1 port2
#configuring the priority of a specified floating IP group on all units in the SSI domain
AN(config)#ha group priority ssi 1 100
AN(config)#ha group enable 1

AN(config)#monitor network gateway APV_1 192.168.1.30 GATEWAY_1 1000 3 3
AN(config)#monitor network gateway APV_2 192.168.1.30 GATEWAY_1 1000 3 3
AN(config)#ha decision rule "GATEWAY_1" Group_Failover 1
AN(config)#ha on
```

6.5.2 SSI Configuration for Switch Failover

To achieve the configuration effect in Switch Failover Work Flow, the following configuration steps are required:

- Switch configuration
- Bond and interface IP configuration, and heartbeat interface configuration to include the APV appliances into one broadcast domain
- SLB configuration
- SSI configuration

The detailed configuration is as follows:

- Switch configuration
  - Connect port1 and port2 of APV1 to the current switch, and port3 and port4 to the standby switch.
  - Connect port1 and port2 of APV2 to the current switch, and port3 and port4 to the standby switch.
  - Connect port5 and port6 of APV1 to the intranet switch.
  - Connect port5 and port6 of APV2 to the intranet switch.
  - Add the current switch ports connecting with port1 and port2 of APV1 and APV2 into a bond.
  - Add the standby switch ports connecting with port3 and port4 of APV1 and APV2 into a bond.
  - Enable the LACP protocol on the bond of the current and standby switches.
  - Configure flow distribution (for example, hash based on the source IP and port) on the bond of the current and standby switches.
  - Add the intranet switch ports connecting with port5 and port6 of APV1 into a bond.
  - Add the intranet switch ports connecting with port5 and port6 of APV2 into a bond.
- IP configuration of bond and interface on the APV appliances

```
#APV1
AN(config)#bond interface bond1 port1
AN(config)#bond interface bond1 port2
AN(config)#bond interface bond1 port3
AN(config)#bond interface bond1 port4
AN(config)#bond interface bond2 port5
AN(config)#bond interface bond2 port6
AN(config)#ip address bond1 100.8.1.1 255.255.255.0
AN(config)#ip address bond2 192.168.1.1 255.255.255.0
#heartbeat configuration for SSI domain
AN(config)#ip address port7 172.16.1.1 255.255.255.0

#APV2
AN(config)#bond interface bond1 port1
AN(config)#bond interface bond1 port2
AN(config)#bond interface bond1 port3
AN(config)#bond interface bond1 port4
AN(config)#bond interface bond2 port5
AN(config)#bond interface bond2 port6
AN(config)#ip address bond1 100.8.1.1 255.255.255.0
AN(config)#ip address bond2 192.168.1.2 255.255.255.0
```

```
#heartbeat configuration for SSI domain
AN(config)#ip address port7 172.16.1.2 255.255.255.0
```

- SLB configuration on the APV appliances

```
#APV1
AN(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
AN(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
AN(config)#slb group method "slb_g1" rr
AN(config)#slb group member "slb_g1" "r1" 1 0
AN(config)#slb group member "slb_g1" "r2" 1 0
AN(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
AN(config)#slb policy default "v1" "slb_g1"
AN(config)#ip pool p1 192.168.1.3 192.168.1.3
AN(config)#slb proxyip group slb_g1 p1

#APV2
AN(config)#slb real tcp "r1" 192.168.1.100 80 65535 tcp 3 3
AN(config)#slb real tcp "r2" 192.168.1.200 80 65535 tcp 3 3
AN(config)#slb group method "slb_g1" rr
AN(config)#slb group member "slb_g1" "r1" 1 0
AN(config)#slb group member "slb_g1" "r2" 1 0
AN(config)#slb virtual tcp "v1" 100.8.1.20 80 arp 0
AN(config)#slb policy default "v1" "slb_g1"
AN(config)#ip pool p1 192.168.1.4 192.168.1.4
AN(config)#slb proxyip group slb_g1 p1
```

- SSI configuration on the APV appliances

```
#APV1 & APV2
AN(config)#ha ssi on
AN(config)#ha unit APV_1 172.16.1.1 65521
AN(config)#ha unit APV_2 172.16.1.2 65521

AN(config)#ha group id 1
AN(config)#ha group port 1 bond1 port1
AN(config)#ha group port 1 bond1 port2
AN(config)#ha group priority ssi 1 200
AN(config)#ha group enable 1

AN(config)#ha group id 2
AN(config)#ha group port 2 bond1 port3
AN(config)#ha group port 2 bond1 port4
AN(config)#ha group priority ssi 2 100
AN(config)#ha group enable 2
```

```
AN(config)#monitor network gateway APV_1 192.168.1.30 GATEWAY_1 1000 3 3
AN(config)#monitor network gateway APV_2 192.168.1.30 GATEWAY_1 1000 3 3
AN(config)#ha decision rule "GATEWAY_1" Group_Failover 1
AN(config)#ha on
```

Chapter 7 Server Load Balancing (SLB)

7.1 Overview

SLB (Server Load Balancing) allows you to distribute load and traffic to specific groups of servers or to a specific server. The APV 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 with SLB policies.

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

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

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

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

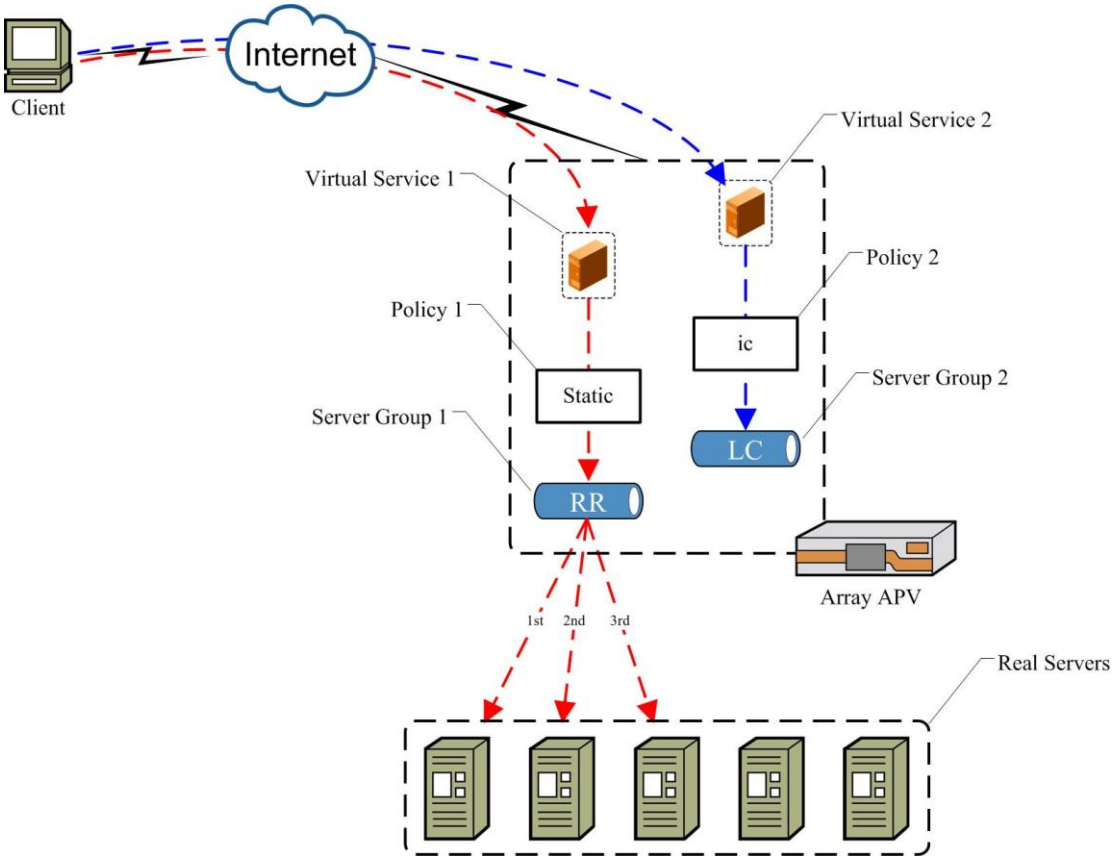


Figure 7 - 1 SLB Architecture

7.2 Understanding SLB

7.2.1 Virtual Service

A virtual service consists of a virtual IP address (VIP) and a port. The user will send the request to this virtual IP and port pair. It is called a virtual service because the actual request is not responded to here, but is instead forwarded by some policy to a backend server in the background service group. In other words, the virtual service is an image of the real server on the appliance, as shown in the picture below.

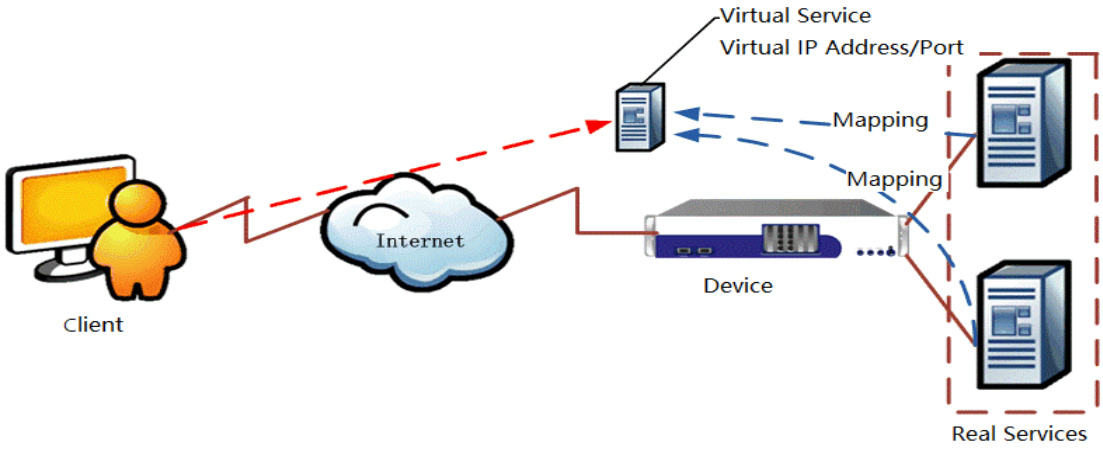


Figure 7–2 Virtual Service

Note: The virtual IP address cannot be the same as the system IP address on the same network adapter. If the virtual service is not bound to the real service group through a policy or there is no working service in the bound real service group, the response "503 Service Unavailable" will be sent to the user to indicate that the service is unavailable.

7.2.1.1 Proxy Protocol

The system supports obtaining the real IP address of the client through Proxy Protocol packets.

Virtual services of TCP, TCPS, HTTP, and HTTPS protocol types can be set as the receiver of v1 or v2 Proxy Protocol packets. After receiving Proxy Protocol packets, the appliance can obtain the source IP address of the client from the packets. For HTTP and HTTPS virtual services, this function can be used together with the "**http xforwardedfor on**" command to forward the obtained client IP address to the real service.

➢ **Configuration Example**

- Set the virtual service as the receiver of Proxy Protocol packets.

```
AN(config)#slb virtual settings proxyprotocol vs1
```

7.2.2 Real Service Group

Real service group is a set of real services that provide the same type of service. Client requests hitting this real service group will be forwarded to the real services selected based on the group method.

This section covers the commands for creating the real service group, setting the group method, adding group members, and configuring other group options.

7.2.2.1SLB Methods

The APV 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 7–1 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.
Global Round Robin (grr)	In terms of the working mechanism, the grr method is the same as the rr method, but it is recommended to use them in different environments. The rr method applies to the single CPU environment. The grr method

	applies to the multi-core CPU environment, in which the rr method cannot guarantee the accuracy of request distribution.
Least Connections (lc)	This metric tells SLB to select the real server with the fewest number of active connections.
Shortest Response (sr)	The server with the shortest response time will get the next request. Using this metric you can intermix fast servers with slow servers and the fast servers will get more hits initially. As load increases, and response time increases and the slower servers will start to field more requests.
Persistent IP (pi)	This metric ties the source IP of the request to the real server processing the request. An example application for this metric is when doing e-commerce transactions and a specific server needs to maintain state about the client's transaction. Keep in mind that if a large ISP deploys a Mega-Proxy, one real server could service thousands of requests. (A Mega-Proxy is used to proxy all client requests from a single IP.)
Persistent Cookie (pc)	Persistent cookie is used to associate a cookie name/value pair with a single real server on your backend. When setting up cookie based policies keep in mind that you need a default policy for requests that do not have cookies in the HTTP header.
Insert Cookie (ic)	Dynamically inserts cookies to allow ArrayOS to maintain persistence to a server.
Rewrite Cookie (rc)	Rewrite Cookie rewrites server side cookies on the fly thereby allowing the backend servers to maintain persistence to a client.
Proximity (prox)	This method is based on SDNS proximity info and used by redirect policy only. It directs SLB to select the real server, which has lowest proximity distance with the request IP.
SNMP (snmp)	This method is based on the real server's SNMP (Simple Network Management Protocol) MIB information regarding the server's status and availability, for example, 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.
Least Bandwidth (lb)	This method directs traffic to the real service with the least bandwidth based on the bandwidths and weight of real services.

Additional Load Balancing methods include: Persistent URL (pu), Persistent Hostname (ph), Hash Cookie (hc), Hash Header (hh), SSL SID (sslsid), Hash IP (hi), Consistent Hash IP (chi), Consistent Hash RADIUS User Name (radchu), Consistent Hash RADIUS Session ID (radchs), radpsun（RADIUS Persistence Session by Username）, Individual Session Persistence (persistence), SIP Call ID (sipcid), SIP User ID (sipuid), SIP CallID Persistence Session (sipcidps), SIP UserID Persistence Session (sipuidps) and Persistent TCP Option (pto).

For more information on these additional SLB methods, please consult the APV CLI Handbook.

7.2.2.2 Proxy Protocol

The system supports obtaining the real IP address of the client through Proxy Protocol packets.

Real service group of TCP, TCPS, HTTP or HTTPS Protocol type can be set as the sender of Proxy Protocol packets. The system supports sending v1 and v2 Proxy Protocol packets.

➢ **Configuration Example**

- Set the real service group as the sender of v2 Proxy Protocol packets.

```
Demo(config)#slb group settings proxyprotocol g1 v2
```

7.2.3 SLB Policies

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

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

7.2.3.1.1.1 SLB Policies

Basic Policy	Persistent Policy	QoS Policy
DoH Redirect Static Default Backup	Persistent URL Persistent Cookie Rewrite Cookie Insert Cookie Hash URL RADIUS Username RADIUS Session ID SIP Domain SIP Call	QoS Cookie QoS Hostname QoS URL QoS Network QoS Clientport QoS Body QoS Dnsdomain QoS Dnsqtype QoS Diameter QoS Diameterappid Regular Expression Header

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

7.2.3.1.1.2 SLB Policy Priority

Priority	Policy
a	DoH
b	Redirect

Priority	Policy
c	Static
d	DNSSEC
e1	QoS Clientport
	QoS Network
	Persistent URL
	Rewrite Cookie
	Insert Cookie
	Persistent Cookie
	QoS Cookie
	QoS Hostname
	QoS URL
	QoS Body
	QoS Dnsdomain
	QoS Dnsqtype
	Regex
	Header
	Hash URL
e2	RADIUS Username
	RADIUS Session ID
	Filetype
	QoS Diameter
	QoS Diameterappid
	SIP Domain
	SIP Call
h	Default
i	Backup

For Layer 4 SLB, only three SLB policies (QoS Network, QoS Clientport, QoS Dnsdomain and QoS Dnsqtype) can be used besides the Static, Default and Backup policies.

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*>

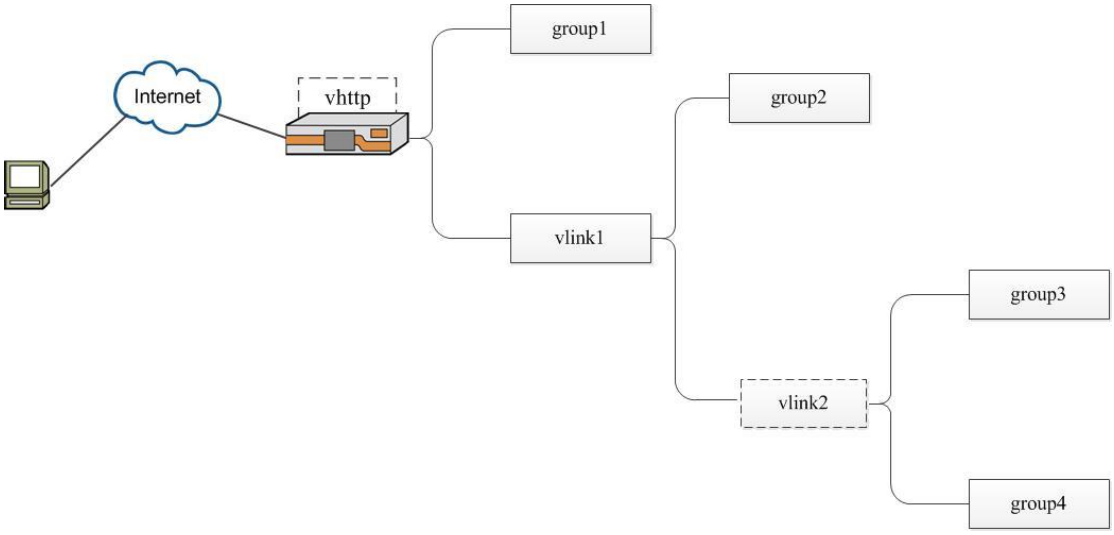
7.2.3.2Policy Nesting

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

In traditional SLB policy mechanisms, one policy is defined to bind a virtual service with a group. For example, in the command "**slb policy qos url policy1 vs1 group1 'news' 1**", the virtual

service vs1 and group1 are associated by policy1. The requests that match the policy1 will be forwarded to group1.

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



7.2.3.2.1.1.1 Policy Nesting

Limitations of SLB Policy Nesting:

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

Note: The DoH policy does not support vlink.

7.2.4 SLB Session Persistence

The SLB module of the APV appliance introduces a new, generally applied, and session ID-based method for session persistence. This persistence method can either operate independently on

Layer 4/7 SLB or work in collaboration with an existing Layer 7 SLB persistence policy. In the latter case, the existing policy extracts the session ID and the persistence method implements the session persistence.

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

7.2.4.1Session ID Types

The persistence method supports the following types of session IDs:

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

7.2.4.2Obtaining the Session ID

Direct extraction by the Persistence Method

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

- Client IP address or IP address and port number from the client request
- SSL session ID extracted from the client request
- A specified string from the HTTP URL query, HTTP cookie, HTTP header, or HTTP body field in the client request (configured by the "**slb group persistence request**" command).
- A specified string from the HTTP cookie, HTTP header, or HTTP body field in the server response. In this case, you also need to configure the APV appliance to obtain the matching specified string from the HTTP URL query, HTTP cookie, HTTP header, or HTTP body field in the client request (configured by the "**slb group persistence response**" command).

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

Indirect extraction through an existing SLB Persistence Policy

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

- Header
- Persistent cookie
- Persistent url
- Qos body

Note: For details about the Session ID matching mechanism with the SLB persistence method and SLB persistence policy, please contact Array Customer Support for related documents.

For more information about the collaboration of the persistence method (configured by the "**slb group method** <*group_name*> **persistence**" command) with existing SLB persistence policies, refer to example "Persistence Method Collaborating with an SLB Persistence Policy".

7.2.4.3Session ID Timeout Management

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

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

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

In addition, the persistence method can statically reserve a session ID (configured by the "**slb group persistence session static**" command). Requests matching a statically reserved session ID will be consistently sent to the same real server.

7.2.5 SLB Health Check

The APV appliance provides four types of health checks:

Basic health check

Basic health checks determine the application, service or server availability (Up/Down) status using a specified protocol format. APV supports basic health check types that include ARP, ICMP (ping), TCP, TCPS, DNS, HTTP, HTTPS and etc. The default health check is assigned with the individual real server (for example, representing any backend physical or virtual server in the server farm) configuration. APV 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. APV supports configuring additional health check name, which helps administrators easily configure and identify the additional health check. See the "**health relation**" command.

Script health check

There are many applications and non-standard protocols (in addition to standard TCP/IP based protocols) that follow specific request/response sequence for communication purposes. To support custom application availability check, customers can make use of scripted health check. APV 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 APV and applications.

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

Group health check

The group health check is used to determine the health status of each real service of the associated group. The health check type supported by the group health check is the same as that of the basic health check. In application, administrators can configure the group health check to obtain the status of each real service without configuring health check for each real service, which simplifies the configurations of the health check.

Besides, the group health check supports the Server Load Balancing in the open cloud environment. For more details about how to configure the group health check in the open cloud environment, contact Array Customer Support for related documents.

Note:

- The relationship of the basic health check and the group health check is "AND".

- The SSL/TLS type basic health check, additional health check and group health check support the configuration of SSL/TLS protocol versions and cipher suites.
- For a non-SSL/TLS basic health check, the SSL/TLS type additional health check will be configured with all SSL/TLS protocol versions and cipher suites supported by the system by default.
- Without the SSL host configured, the SSL/TLS type basic health check, additional health check and group health check can still function. The ClientHello message will carry all SSL/TLS protocol versions and cipher suites supported by the system by default.

7.2.5.1Methods of Health Check

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

ICMP Health Check

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

TCP Health Check

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

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

TCPS Health Check

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

HTTP & HTTP2 Health Check

The basic built-in HTTP and HTTP2 health check opens a TCP connection and sends an HTTP request with one of the HTTP methods (such as GET, POST, etc.) pre-defined in the health request table. The APV 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”. The system supports configuring basic health check, additional health check and the group health check of HTTP and HTTP2 type.

HTTPS & HTTPS2 Health Check

HTTPS and HTTPS2 health check provides an SSL health check for real servers, while HTTPS2 health check does not implement certificate verification. If the SSL handshake succeeds, the APV 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 and HTTPS2 health check, users should pre-define HTTP requests/methods and corresponding expected responses for matching purposes. For HTTPS health check, the imported client certificate must be encoded by DER rules during client authentication.

Note: To make the system work properly, it is recommended to configure 500 HTTP2 health checks at most, including basic, additional, and group health checks.

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 APV expects a special DNS response. The DNS request and response are not configurable because they are unchangeable. If the required conditions are satisfied, then server will be marked as “UP”, otherwise, the server will be marked as “DOWN”.

Radius-Auth Health Check & Radius-Acct Health Check

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

LDAP Health Check

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

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

RTSP-TCP Health Check

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

SIP-UDP & SIP-TCP Health Check

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

POP3 Health Check

POP3 health check conducts a series of handshakes with the real service to check its health state. First, the appliance establishes a TCP connection and checks if the real service sends the POP packet that indicates the ready status. After that, the appliance sends the USER request and PASS request to the real service to authenticate client identity. Then, the appliance sends the STAT request or the customized request (configured with the "health request" command) to see if the service can return the expected response (configured with the "health response" command). After receiving the correct response, the appliance sends the QUIT request. If the real service sends back the expected response, it will be marked as "up". Otherwise, it will be marked as "down".

SNMP Health Check

SNMP health check conducts a series of handshakes with the real service to check its health state. The appliance sends a request to query the SNMP node. If the real service sends back the expected response, it will be marked as "up". Otherwise, it will be marked as "down".

SMTP Health Check

SMTP health check conducts a series of handshakes with the real server to check its health state. The appliance establishes a TCP connection with the real server and then checks if the real server sends the SMTP Ready packet. After that, the appliance sends a NOOP request or user-customized request and then the QUIT request. If the real server sends back expected responses one by one, the server will be marked as "up". Otherwise, it will be marked as "down".

Database Health Check

Database health check checks the health status of real database servers by using a series of health checks between the APV appliance and the real database servers. The APV appliance first establishes a TCP connection with the real database server, and then it logs into the real database server and send a database query to the real database server. If the real database server responds with an expected response, it will be marked as "up"; otherwise, it will be marked as "down".

Currently, the system supports three types of database health checks: Oracle, MySQL and MsSQL. MySQL database health check includes MySQL-DB and MySQL-DBS, and MySQL-DBS is an

SSL-based MySQL database health check. MsSQL database health check includes MsSQL-DB and MsSQL-DBS, and MsSQL-DBS is an SSL-based MsSQL database health check.

7.2.5.2HC 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 APV appliance does health check. An HC checker is defined as one transaction of health check. It consists of sending one message and receiving one response. A list of HC checkers can compose an HC checker list, which is identified by the HC checker list name.

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

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

health list <*list_name*>

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

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

7.2.5.3HTTP Requests and Responses

By default ArrayOS 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:

```
AN(config)#health on
AN(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
AN(config)#show health response
Row    Response:
0       200 OK
1       200 OK
2       200 OK
3       200 OK
4       200 OK
5       200 OK
6       200 OK
7       200 OK
8       200 OK
9       200 OK
10      200 OK
11      200 OK
12      200 OK
13      200 OK
14      200 OK
15      200 OK
```

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

```
HEAD / HTTP/1.0
```

The response needed from the real server is:

```
200 OK
```

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

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

Changing the HTTP Health Request/Response

You can define your own HTTP requests and the responses to be used by the HTTP health check. For example, you may simply change the request to get a CGI script that returns an HTTP status 200 OK when the database server is up and a 404 NOT FOUND when the database server is "down".

Below are the commands to use to perform this task as well as an example from our network model.

health request <*request_index*> <*request_string*>

health response <*response_index*> <*response_string*>

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

For our network example:

1. Define your own HTTP request.

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

Set the request to index 1, the response to index 0 for server2http.

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

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

The health check provided by the ArrayOS 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 **arrayadmin**

1. Add a real service with HTTP health check.

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

Configure HTTP health check.

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

Associate the configured HTTP health check with the real service.

```
AN(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 (for example, simplified Chinese, traditional Chinese).

7.2.5.4TCP-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 APV 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.

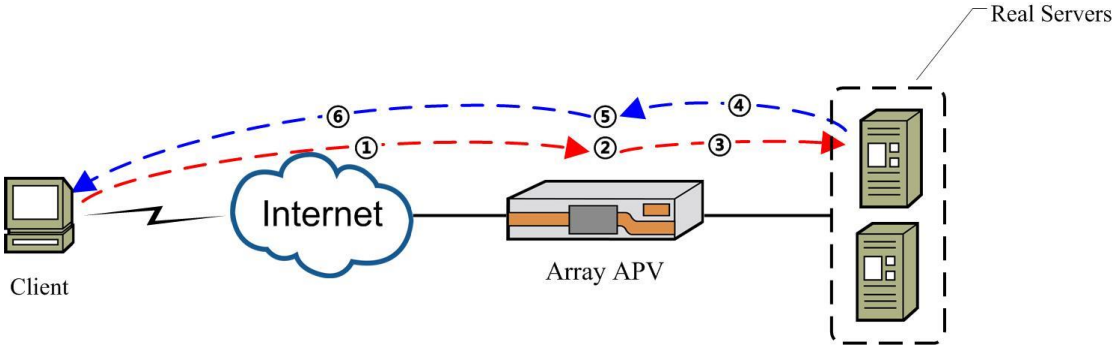
7.2.5.5Health Failover

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

7.2.6 Transparent, Reverse Proxy and Triangle Transmission

7.2.6.1Transparent Transmission

For transparent transmission, the APV appliance will direct the connections toward specified real servers directly when forwarding requests from clients. That is, the server is aware of the client’s IP address and therefore knows which client accesses the server. In transparent mode, the source IP stays unchanged, while the source port might change. If administrators want to keep the source port unchanged under transparent mode, the difference value between virtual service port number and real service port number must be a multiple of APV CPU core numbers.



7.2.6.1.1.1.1 Transparent Transmission

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

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

- Advantages of transparent transmission

  The server can record the IP addresses of the clients.

- Limitations of transparent transmission

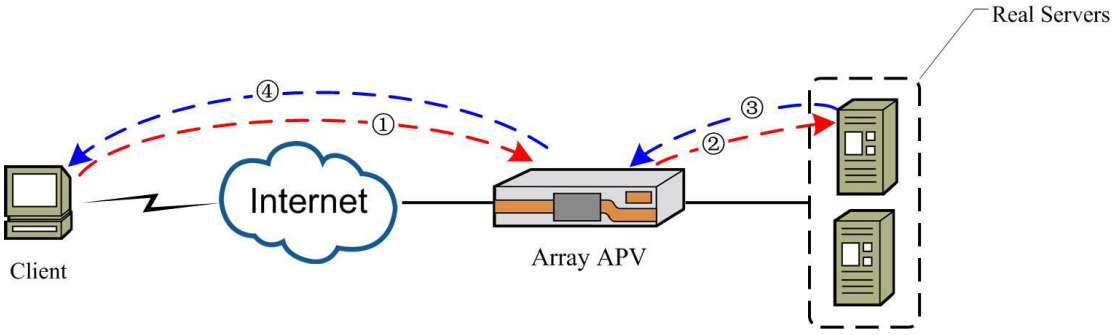
  Structure/router design requires responses from source servers to pass through the APV appliance.

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

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

7.2.6.2Reverse Proxy Transmission

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



7.2.6.2.1.1.1 Reverse Proxy Transmission

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

- Advantages of reverse proxy transmission

  One-armed deployment is applicable.

  Connection pool technology can be used to enhance system performance.

- Limitations of reverse proxy transmission

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

- Solution

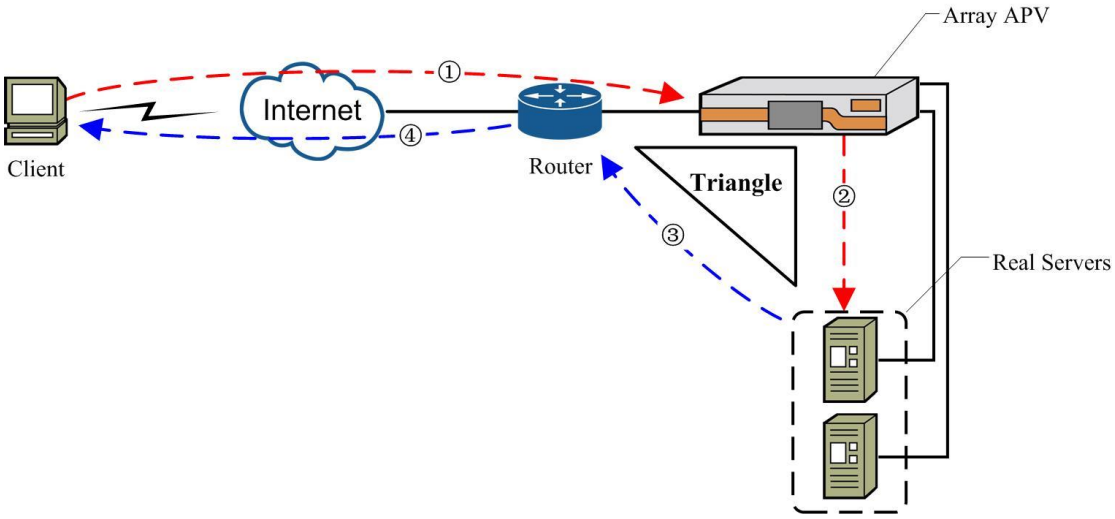
  The APV appliance can insert a field “X-Forwarded-For” into the header of the HTTP request of the client to trace the client IP.

7.2.6.3Triangle Transmission

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

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

The following shows how the Triangle Transmisison works.



7.2.6.3.1.1.1 Triangle Transmission

1. Client sends a request to a virtual IP on the APV appliance via the router.

The APV appliance forwards the request to a real service.

The real service returns a response to the router directly. Since the default route IP on the real service is set to be port2 interface IP of the router, the response will be sent to the router directly.

The router forwards the response to the client.

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

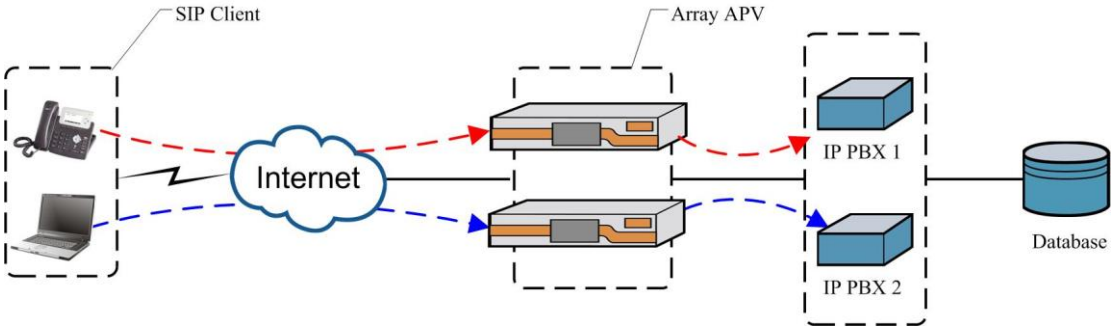
7.2.7 Packet based UDP Load Balancing

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

7.2.8 SIP Load Balancing

What is SIP (Session Initiation Protocol) Load Balancing

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



7.2.8.1.1.1.1 SIP Load Balancing

SIP Load Balancing performs stateful inspection of SIP messages to scan and hash calls based on a SIP Call-ID header destined for a SIP server. Stateful inspection means that a packet is inspected not only for its source and destination information found in the header, but also packet contents

found at Layer 7 (the Application Layer). Once the APV appliance has identified the Call-ID which identifies a specific SIP session, it sends future messages from the same Call-ID to the same SIP server.

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

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

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

SIP load balancing supports the following policies:

- SIP Domain policy: With the virtual service associated with the real service group by the SIP Domain policy, when a SIP request matches the domain name and call flow type specified in the SIP Domain policy, the SIP request will be forwarded to the associated group. SIP requests accessing the same domain name can be forwarded to the same group persistently based on SIP Domain policies.

  Besides the domain name, the SIP Domain policy can be triggered based on three types of call flows: outgoing call flow, incoming call flow and other call flows.

- SIP Call policy: SIP Call policies are triggered based on the domain name, call flow type and call number type.

  - Call flow types: Compared with the SIP Domain policy, the SIP Call policy is triggered based on two types of call flows: outgoing call flow and incoming call flow.

  - Call number type: The types of call numbers include the caller number and the callee number. The caller number can be triggered based on the PAI (logical caller number) or FROM (source caller number) header field, which supports prioritization. The callee number can be triggered based on the request line (logical callee number) or To (source callee number) header field, which also supports prioritization.

The SIP Call policy must be used together with the call number full matching rule (configured by the "**slb sip callnum list import**" command) or the call number regex matching rule (configured by the "**slb sip callnum regex**" command). After a SIP Call policy is configured to associate the virtual service and real service group, the administrator must configure a call number full matching rule or a call number regex matching rule for the policy. The two types of matching rules can be configured for both the caller number and the callee number. Therefore, there are a maximum of four types of matching rules that can be configured for each policy:

  - Caller number full matching rule

  - Callee number full matching rule

  - Caller number regex matching rule

- Callee number regex matching rule

When all of the four types of matching rules are configured for a policy, the policy is hit if any one of these rules is matched.

SIP Call policy configuration example:

1. Configure a SIP TCP virtual service “siptcpvs".

```
AN(config)#slb virtual siptcp siptcpvs 10.8.6.138
```

Configure a real service group “g1” to use the sipcid method.

```
AN(config)#slb group method g1 sipcid
```

Configure a SIP TCP real service “rs1” and “rs2” and add them to the group “g1”.

```
AN(config)#slb real siptcp rs1 192.169.0.126
AN(config)#slb real siptcp rs2 192.169.0.127
AN(config)#slb group member g1 rs1
AN(config)#slb group member g1 rs2
```

Configure a SIP Call policy “p1” for outgoing call flows.

```
AN(config)#slb policy sipcall p1 siptcpvs g1 abc.com 12 calling from to
```

Configure a callee number regex matching rule for the policy “p1”.

```
AN(config)#slb sip callnum regex p1 calling “[1][3578]\\d{9}”
```

Configure a real service group “g2” to use the rr method.

```
AN(config)#slb group method g2 rr
```

Configure a SIP TCP real service “rs3” and add it to the group “g2”.

```
AN(config)#slb group member g2 rs3
```

Configure a default policy.

```
AN(config)#slb policy default siptcpvs g2
```

Based on the configurations above, the SIP TCP virtual service will forward the SIP request meeting the following conditions to the group “g1”:

- The SIP request is an outgoing call flow;
- The requested domain name is “abc.com”;
- The callee number matches the regular expression “[1][3578]\\d{9}”.

The group “g1” will forward the SIP requests to “rs1” and “rs2” based on the sipcid method. The SIP TCP virtual service will forward all other SIP requests to the group “g2” based on the default policy.

7.2.9 RTSP Load Balancing

What is RTSP (Real Time Streaming Protocol)

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

What is RTSP Load Balancing

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

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

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

- Redirect mode

  The media stream will not pass through APV appliance. After APV 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 APV appliance. After APV 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.

7.2.10Tuxedo Load Balancing

Transactions for Unix Extended for Distributed Operation (Tuxedo) provides a middleware between the client and the server used to build reliable and distributed enterprise applications on a large scale. It allows an enterprise to implement simplified development and deployment while keeping existing assets unchanged. The Tuxedo server processes transactions using two connections:

1. A workstation client (WSC) requests to establish a TCP connection with the workstation listener (WSL).

2. After the WSL validates the request, it will assign a WSH process to handle the request and sends the IP address and port number of the WSH process to the WSC.
3. The WSC requests to connect the WSH. Subsequent requests and responses will be transferred over this connection.

In Tuxedo load balancing scenarios, the APV appliance is deployed between the WSC and the WSL. It receives a WSC's requests on the Tuxedo virtual service and then sends the requests to the corresponding Tuxedo group based on the configured policy. The APV appliance maintains a mapping table for each Tuxedo group, which records the mapping relationships between the WSH port number and the real service.

- Upon receiving a WSC's request, the group will search the mapping table for the real service corresponding to the WSH port number. If no entry exists, the Tuxedo group will select a real service based on the first choice method. When the real service returns the IP address and port number of the WSH, the APV appliance will add the correspondence between the real service and the port number to the mapping table.
- If the WSH port number accessed by the WSC is available in the mapping table, the APV appliance will obtain the corresponding real service and send the request to it.

Note:

1. The WSH port numbers assigned to each Tuxedo server must not overlap.
2. The IP addresses of all WSH processes must be the same as the IP address of the Tuxedo virtual service.
3. All WSLs can only assign WSH processes on the local host.

7.2.10.1 Configuration Example

1. Configure a Tuxedo virtual service.

```
AN(config)#slb virtual tuxedo vs1 172.16.73.155 0
```

Configure a Tuxedo real service.

```
AN(config)#slb real tuxedo rs1 172.16.72.145 1000 icmp 3 3
```

Configure a Tuxedo real service group.

```
AN(config)#slb group method g1 tuxedo lc 10
```

Configure a default policy.

Tuxedo load balancing supports the Static policy, Default policy, QoS Clientport policy and QoS Network policy.

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

7.2.11WebSocket Load Balancing

The WebSocket protocol provides a mechanism for browser-based applications that need two-way communication with servers without opening multiple HTTP connections and it enables the server to push messages to clients. It has two parts: a handshake and the data transfer. A WebSocket connection is established by upgrading the HTTP protocol to WebSocket during the handshake. The subsequent WebSocket messages will be transferred back and forth over this connection. In this process, the APV system can function as a reverse proxy server to forward WebSocket (WS) and WebSocket Secure (WSS) traffic running on top of HTTP/1.1 between a client and a server.

In a scenario where the system forwards WS and WSS traffic to a group of servers, it supports the load balancing of it according to the configured HTTP or HTTPS SLB policies and methods. With the exception of the QoS Body policy, all policies and methods supported by HTTP and HTTPS SLB are applicable to the load balancing of WS and WSS traffic.

7.2.12DNS over HTTPS（DoH）

DNS over HTTPS (DoH) is a protocol for performing remote Domain Name System (DNS) resolution via the HTTPS protocol. It prevents man-in-the-middle attacks and protects user privacy and security by using the HTTPS protocol to encrypt the data between the DoH client and the DoH-based DNS servers

SLB can help your network infrastructure to support DoH without upgrading or rebuilding backend DNS servers. SLB provides the DoH policy to associate a group of legacy DNS servers with the HTTPS virtual service, achieving the provision of DoH services.

- **Configuration Example**

1. Configure an HTTPS virtual service.

```
AN(config)#slb virtual https "vs1" 192.168.12.62 443 arp 0
```

Configure DNS real services and a group.

```
AN(config)#slb real dns dns_rs1 192.168.2.2
AN(config)#slb real dns dns_rs2 192.168.2.3
AN(config)#slb group method g1 rr
AN(config)#slb group member g1 dns_rs1
AN(config)#slb group member g1 dns_rs2
```

Configure the DoH policy to associate the HTTPS virtual service with the DNS group.

```
AN(config)#slb policy doh vs1 g1
```

7.2.13Layer 2 IP/MAC-based Load Balancing

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

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

Generally, in a scenario where Layer 2 SLB is applied, the system will forward packets whose destination IP is not a local IP address. To enable the Layer 2 SLB to forward packets whose destination IP matches a local IP (such as VIP, interface IP, etc.), the administrator needs to configure these local IPs as local exception IPs for Layer 2 SLB. For details, refer to 7.3.5.3 Local Exception IP Configuration for Layer 2 SLB.

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

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

7.2.14Layer 3 IP-based Load Balancing

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

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

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

7.2.15Port Range Load Balancing

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

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

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

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

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

7.2.16Terminal Server Load Balancing

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

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

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

routing token. APV appliance will reconnect the client to TS2 according to the IP address and port in the routing token.

7.2.17RADIUS Server Load Balancing

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

Note: Compared with the "radchu" method, the "radpsun" method enhances the RADIUS sesseion persistence, which means when the same RADIUS request hits the same service group, the timeout value of this RADIUS session will be refreshed.

7.2.18DirectFWD

DirectFWD is a new Layer 4 SLB function by utilizing a multi-thread and non-lock architecture based on a multi-core system. This new architecture has maximized the advantage of the multi-core system. Compared with the traditional Layer 4 SLB, DirectFWD provides remarkably better Layer 4 SLB performance. This function is enabled by the command "**slb directfwd on <virtual_service>**" and disabled by the command "slb directfwd off".

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

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

- It is only used for TCP SLB, and temporarily only supports static, default and backup policies. Please note that after enabling the DirectFWD function, the SR method will stop to take effect. All the other Layer 4 SLB methods are supported.
- 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.
- It cannot handle IP fragments.

Note:

- The more the interfaces used for DirectFWD, the better the performance.
- DirctFWD is not supported in Triangle Transmission SLB.

DirectFWD Syncache

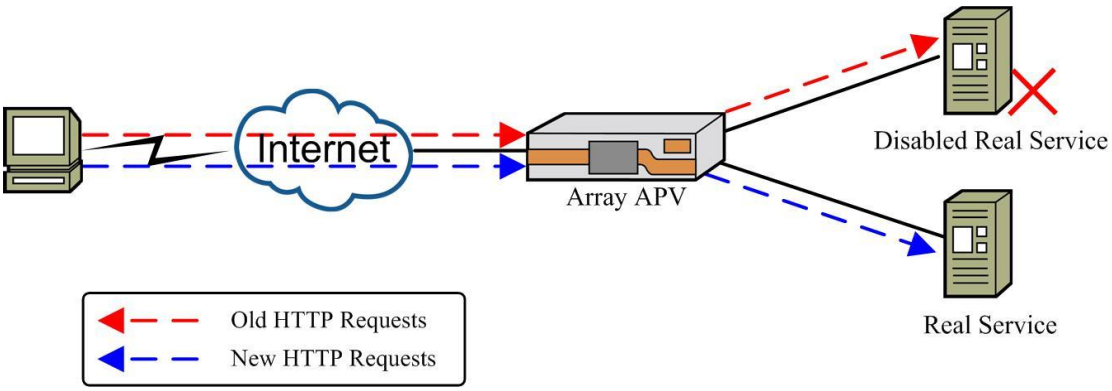
DirectFWD syncache effectively and efficiently protects the real servers from SYN flood DOS attacks. When DirectFWD syncache function is on, all the SYN packets from a client will not be forwarded to a real server directly. The APV appliance will hold the useful information in those SYN packets firstly,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

7.2.19Real Service Graceful Shutdown and Warm-up Activation

7.2.19.1 Real Service Graceful Shutdown

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



7.2.19.1.1.1.1 Graceful Shutdown of Real Services

The following gives an example of Graceful Shutdown:

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

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

```
AN(config)#show statistics slb real http service
Real service service 10.8.6.42 80 DOWN INACTIVE(waiting)
```

```
        Main health check: 10.8.6.42 80 tcp DOWN
        Max Conn Count:                1000
        Current Connection Count:    4572
        Outstanding Request Count: 4215
        Total Hits:                    311
        Total Bytes In:                39431
        Total Bytes Out:               53466
        Total Packets In:              7541
        Total Packets Out:             3252
        Average Bandwidth In:          543 Kbps
        Average Bandwidth Out:         748 bps
        Average Response time:         32.000 ms
```

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

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

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

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

7.2.19.2 Real Service Warm-up Activation

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

Right after a real service is enabled, no connection requests will be sent to the real service for a period of time. This period is called "recovery time". When the recovery time ends, the real

service enters the “warm-up time”. In this period of time, firstly only a small amount of connection requests are forwarded to the real service for processing. The number of the requests will be increased gradually. When it reaches the maximum number allowed for the real service, it means the real service works normally.

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

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

7.2.20Group Member Activation

The system allows the administrator to set the member activation policy, controlling which real services in the group process traffic.

The real service group supports two member activation modes:

- Priority-based activation mode: The system activates the allowed number of real services in a group in the descending order of priority.
- Priority subgroup-based activation mode: The system divides real services into subgroups according to their priorities and only real services in the highest-priority subgroup process the traffic.

In addition, the system supports monitoring the availability of the group and subgroup, and performing traffic switchover when the group or a subgroup is unavailable.

In priority-based activation mode, if the number of available real service is less than the minimum threshold, the group is regarded as unavailable and the traffic destined for it will be switched to the group associated with the backup policy.

In priority subgroup-based activation mode, if the number of available real services in this subgroup is less than the minimum threshold, this subgroup is regarded as unavailable and the subgroup with the second highest priority will process the traffic. If all priority subgroups are

unavailable, the traffic destined for the group will be switched to the group associated with the backup policy.

By default, the group uses the priority-based activation mode, allows all real services in the group to be used, and does not monitor the group availability.

➢ **Configuration Example**

Add real services to the real service group and set priorities for these members:

```
AN(config)#slb group member "g1" "r1" 1 3
AN(config)#slb group member "g1" "r2" 1 3
AN(config)#slb group member "g1" "r3" 1 3
AN(config)#slb group member "g1" "r4" 1 2
AN(config)#slb group member "g1" "r5" 1 2
AN(config)#slb group member "g1" "r6" 1 2
AN(config)#slb group member "g1" "r7" 1 1
AN(config)#slb group member "g1" "r8" 1 1
AN(config)#slb group member "g1" "r9" 1 1
```

Set the member activation policy for the group:

```
AN(config)#slb group activation g1 0 2 1
```

After the preceding configuration, the priority subgroup-based activation mode is enabled for g1. The subgroup with priority 3 (r1, r2, and r3) will process the traffic of this group.

7.2.21Real Service Management based on Host Node

A SLB host node is a logical object based on IP address or domain name, and is used to help identify and manage real services with the same IP address or domain name.

Real services with the same IP address or domain name will be assigned to the same host node. Administrators can enable or disable a specified host node. When a host node is enabled or disabled, all real services associated with the host node will be enabled or disabled, so that batch management of real services based on IP addresses or domain names can be realized.

Currently, the system supports the following two ways to create a SLB host node:

- Auto-generated host node: When a real service is created, the system will automatically create a host node based on the IP address or domain name of the real service. Auto-generated host nodes can only be automatically deleted when the real services are deleted. Auto-generated host nodes cannot be saved into the configuration.
- Manually created host node: Manually create a host node through the "**slb node name**" command. Manually created host nodes need to be deleted manually. Manually created host nodes can be saved into the configuration.

By default, all host nodes are enabled. The maximum number of manually configured host nodes that the system supports is the same as that of the real services.

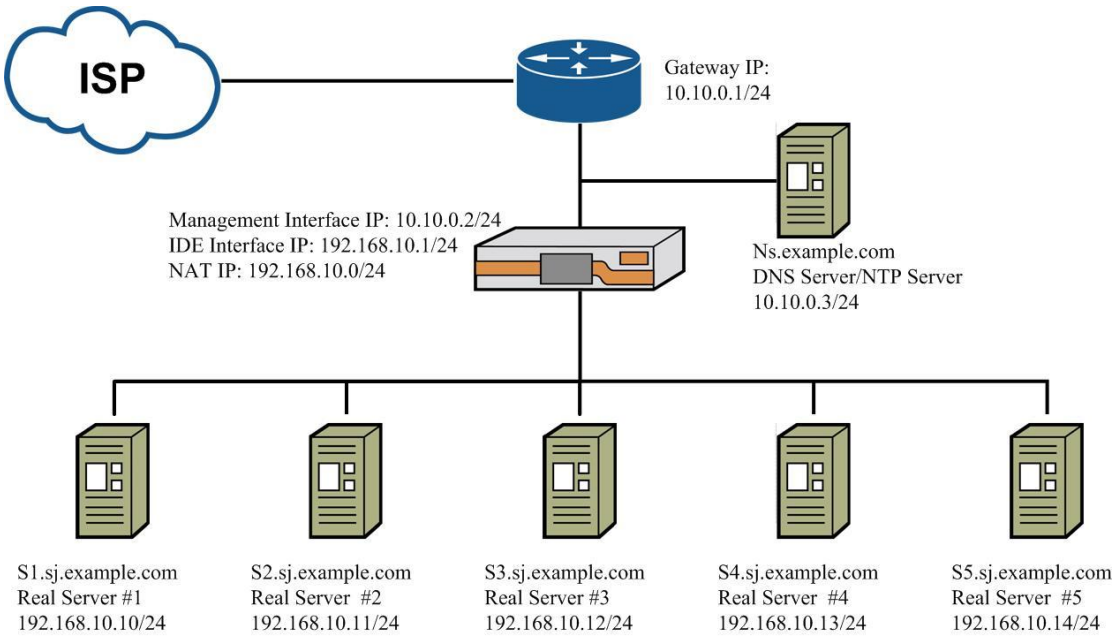
7.3 SLB Configuration

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

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

7.3.2.1Configuration Guidelines

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



7.3.2.1.1.1 SLB Network Architetcure

7.3.2.1.1.2 General Settings of SLB

Operation	Command
Configure real services	**slb real tcp** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **slb real ftp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real udp** *<real_service> <ip> <port> [max_connection] [hc_up]*

Operation	Command
	[hc_down] [timeout] [hc_type] **slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real tcps** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **slb real dns** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]* **slb real health** *<add_hc_name> <real_service> <ip> <port> [hc_type] [hc_up] [hc_down]*
Define group methods	**slb group method** *<group_name>* **[rr|pu|sr]** **slb group method** *<group_name>* **hc** *[first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **ic** *[cookie_name] [path_attribute] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **rc** *[cookie_name] [offset] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **lc** *[threshold_granularity] [yes|no]* **slb group method** *<group_name>* **hh** *<header_name> [first_choice_method] [threshold_granularity] [prefix] [delimiter]* **slb group method** *<group_name>* **ec** *<cookie_name> [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **lb** *[threshold_granularity]* **slb group method** *<group_name>* **pto** <tcp_option_kind> [*first_choice_method*]
Add the real servers into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]* **slb health** *<group_hc_name> [type] [interval] [timeout] [hc_up] [hc_down] [http_method] [url_path] [expected_codes]* **slb group health** *<group_name> <group_hc_name>*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]* **slb virtual ip** *<virtual_service> <vip> [max_connection]* **slb virtual l2ip** *<virtual_service> <vip> [gateway_ip]*
Bind the group (or a real service) to the virtual service	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy qos url** *<policy_name> {virtual_service|vlink_name} {group_name|vlink_name} <url_string> <precedence>* **slb policy qos cookie** *<policy_name> {virtual_service|vlink_name}*

Operation	Command
	{group_name|vlink_name} *<policy_string>* *<precedence>* **slb policy persistent cookie** *<policy_name>* *{virtual_service|vlink_name}* *<group_name>* *<cookie_name>* *<precedence>* **slb policy qos hostname** *<policy_name>* *{virtual_service|vlink_name}* *{group_name|vlink_name}* *<host_name>* *<precedence>* **slb policy redirect** *<policy_name>* *<virtual_service>* *<group_name>* *<redirected_from_host>*

7.3.2.2Configuration Example via CLI

1. Define real services.

The first step in setting up your network architecture with the APV 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:

```
AN(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: 0
- Health Check: tcp
- Consecutive up health check results before server is marked up: 3
- Consecutive down health check results before server is marked down: 3

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

```
AN(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.

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

Now all five real services are defined for the APV 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:

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

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

Maintaining the Real Services

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

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

Now verify our configuration change:

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

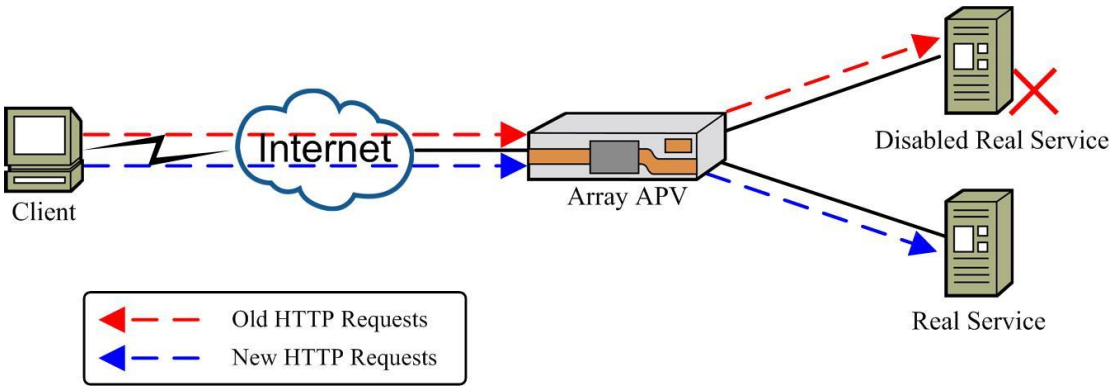
To re-enable the real server just use the "**slb enable**" command.

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

By default, when a real service is disabled or deleted, the APV 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.



7.3.2.2.1.1.1 Graceful Shutdown of Real Services

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

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

```
AN(config)#slb real disable r1
```

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

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.

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

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

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

(Optional) Configure group health check:

Define a group health check condition.

```
AN(config)#slb health check1 http 30 3 3 3 GET / "200"
```

Associate the group health check condition with the specified group.

```
AN(config)#slb group health rrgroup check1
```

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:

```
AN(config)#slb group member rrgroup service1http
AN(config)#slb group member rrgroup service2http
AN(config)#slb group member rrgroup service3http
AN(config)#slb group member rrgroup service4http
AN(config)#slb group member rrgroup service5http
```

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".

Note:

- The VIP address cannot be the same IP as any management IP address.
- The VIP address configured must be within the same subnet with any system interface on the appliance (except the 0.0.0.0 and noarp cases). For the VIP address

that does not match the subnet of any interface, the APV appliance will not allow it to be configured.

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

To establish virtual services:

```
AN(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.

Define policies.

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

```
AN(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 APV appliance. You should be able to test the configuration by typing the HTTP URL: "http://10.10.0.10/" in a Web browser.

For an existing virtual service or real service, the system supports direct modification of its IP addresses, port number and maximum number of connections. It is not mandatory that the existing one be deleted for re-configuration.

Note:

1. L2ip virtual service (configured by the "**slb virtual l2ip**" command) does not support direct modification of the IP address.
2. After the modification of a real service's IP address or port number, all "Total…" statistics of this real service which can be viewed in the "**show statistics slb real**" command output will be reset to zero; If the real service is an HTTP or HTTPS type, the "HTTP response code" statistics of it which can be viewed in the "**show statistics slb real**" statistics will also be reset to zero; If the real service joins in a group, the hit statistics of it as a group member which can be viewed in the "**show statistics slb group**" command output will also be reset to zero.

For an existing real service group, administrators can freely modify its method as required before it is associated with a virtual service and assigned a member. For example, if group "g1" is configured to use the pu method but it has not any members, you can change it to use any other method that the system supports.

After a group is associated with a virtual service via a policy or is assigned a member, whether its method can be modified depends on the method type, protocol type and other factors, and the system will display relevant prompts.

7.3.2.3Policies-based SLB Configuration Example

QoS URL

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

Group 1: URL: .jpg

Members:

- S1.sj.example.com
- S2.sj.example.com

Group 2: URL: 'english'

Members:

- S3.sj.example.com
- S4.sj.example.com

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

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

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

Add the real services into the groups (the real services are defined in the last example).

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

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

```
AN(config)#slb policy qos url url_pol_1 virtual1http group1 ".jpg" 1
AN(config)#slb policy qos url url_pol_2 virtual1http group2 english 2
```

Define a default policy.

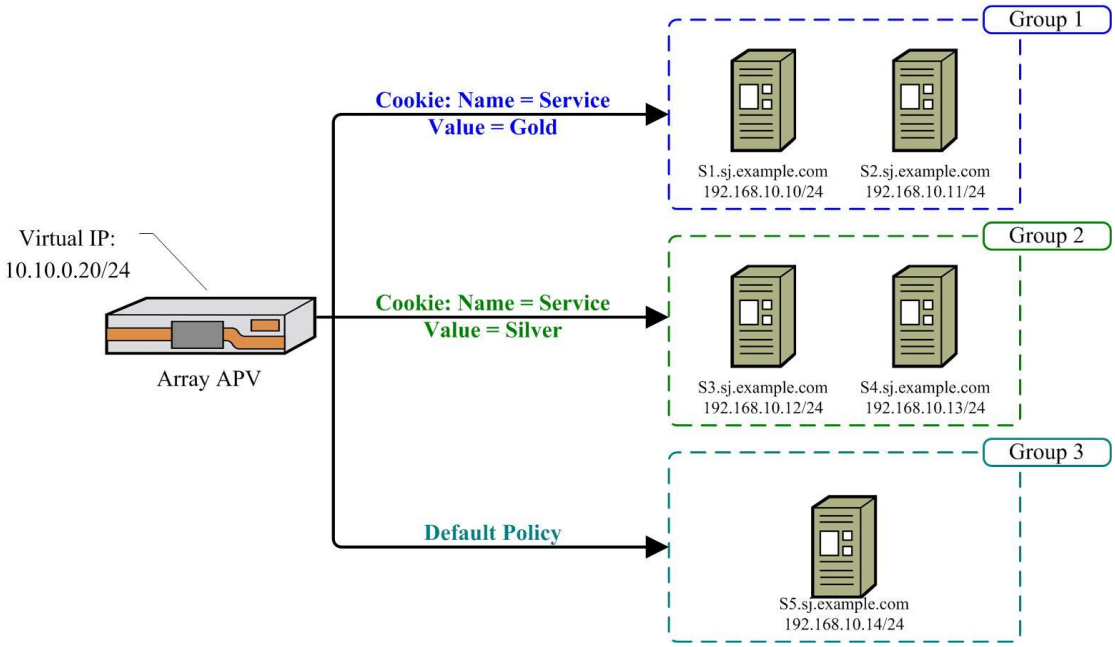
If we receive a request that has neither .jpg or 'english' in the URL, then the APV appliance will returne a 503 Service Unavailable since it does not know how to handle the request. This is fine if it is guaranteed every URL will always contain one of the strings. If not, it is best to define the

default policy. In our case we are just going to direct any request that does not contain one of the strings to go to the “english” servers belonging to group 2.

AN(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.



7.3.2.3.1.1.1 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:

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

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

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

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

1. Define three groups.

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

Add the real services into the groups.

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

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

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

Set the default policy for group “cookie_set_group”, where cookies will be set.

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

Set the QoS Cookie policy for group “gold_group” and “silver_group”.

```
AN(config)#slb policy qos cookie gold_policy virtual1http gold_group "service=gold" 1
AN(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.

1. Define two groups.

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

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

Add the real services into the groups.

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

Set the default policy for "group2", where cookies will be set.

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

Set the persistent cookie policy for "group1".

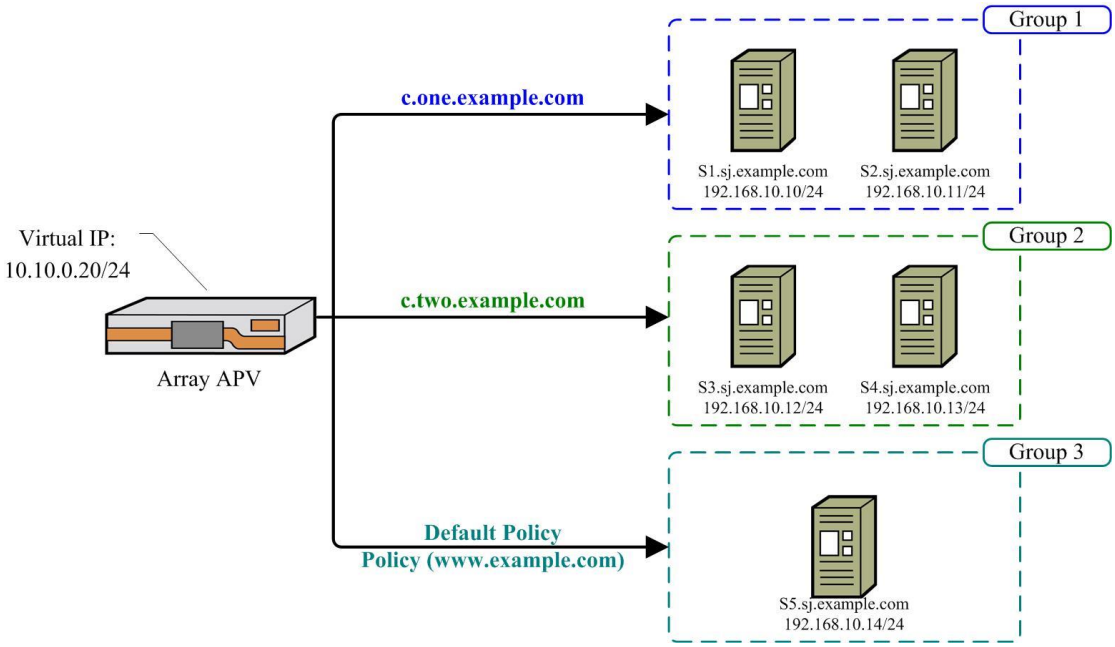
```
AN(config)#slb policy persistent cookie perst_pol virtual1http group1 Service 1
```

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.



7.3.2.3.1.1.2 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

1. Define the groups.

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

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

Add the real services into the groups.

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

Define the policy for “www_group”.

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

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

Define the QoS Hostname policy for “c_one_group” and “c_two_group”.

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

Persistent URL

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

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

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

1. Create the group for Persistent URL policy.

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

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

Add the real services into the group.

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

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

Setup the policies.

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

Insert Cookie

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

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

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

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

Define the Insert Cookie properties for the group.

```
AN(config)#slb group option ic ic_group "expires=300 secure=yes"
```

Add real services into the Insert Cookie group.

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

Set up the SLB policies to associate the group with virtual services.

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

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

```
AN(config)#show slb group method
slb group method ic_group "yqv" 1 rr
```

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

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

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

CookieExampleName=server1http

Rewrite Cookie

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

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

1. Create the group for Rewrite Cookie policy.

```
AN(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.

Add the real services into the group.

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

Setup the policies.

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

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

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

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

Create a group.

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

Add the real services into the group.

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

Use the “**slb policy redirect**” command to associate an existing virtual service with the group.

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

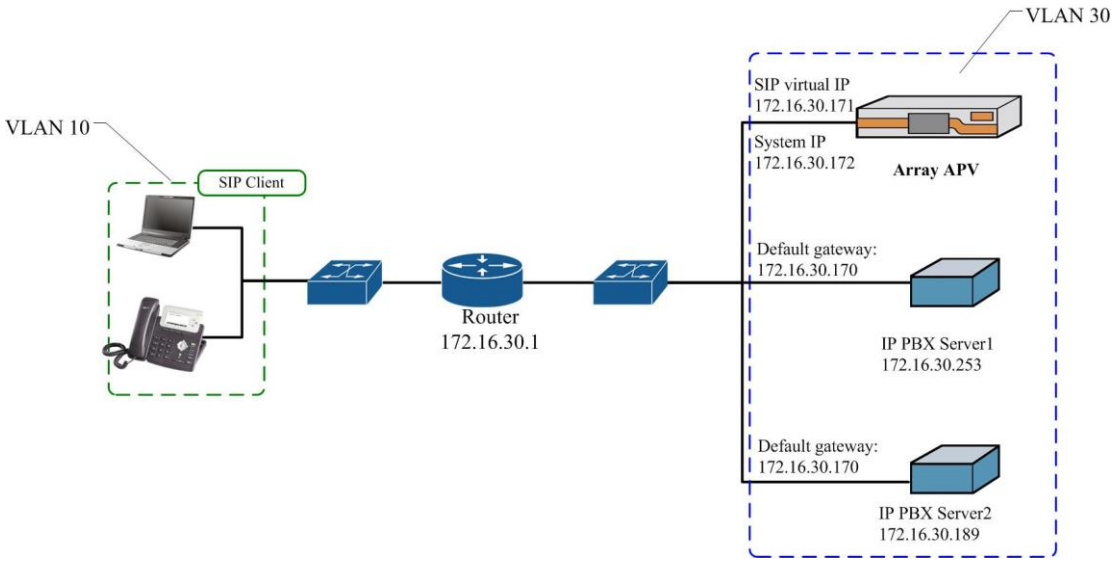
Currently, the Redirect policy supports rr (Round Robin), lc (Least Connection), sr (Shortest Response Time) and prox (Proximity) load balancing methods.

7.3.3 SIP Load Balancing

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

7.3.3.1Configuration Guidelines

In this section, the example is a one-arm case. The default Gateway of two servers is APV 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.



7.3.3.1.1.1 SIP Load Balancing

7.3.3.1.1.2 General Settings of SIP Load Balancing

Operation	Command
Configure real services	**slb real siptcp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real sipudp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name>* **{sipcid|sipuid|sipcidps|sipuidps}** *[first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** [threshold_granularity] [yes|no] **slb group method** *<group_name>* **pi** *[hash_bits] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **snmp** *[snmp_mode] [community] [oid_count] [oid1] [oid1_weight] [oid2] [oid2_weight] [check_interval]*
Add the real servers into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]*
Bind the group to the virtual	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>*

Operation	Command
service	**slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}*

7.3.3.2Configuration Example via CLI

1. Define SIPUDP real services.

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

Create a group for SIP load balancing by using the "**slb group method**" command.

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

Add SIPUDP real services into the group.

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

Create virtual services.

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

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

Associate the group to the virtual service for SIP SLB.

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

Configure SIP Multi-register.

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

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

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.

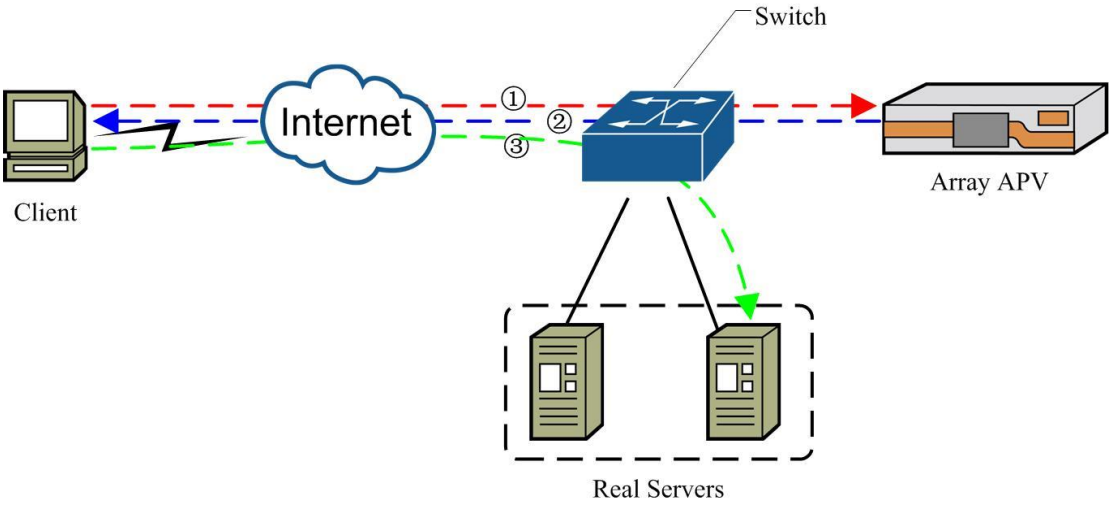
```
AN(config)#sip nat 172.16.30.171 5060 172.16.32.253 5060 udp 60 callid
AN(config)#sip nat 172.16.30.171 5060 172.16.32.189 5060 udp 60 callid
```

7.3.4 RTSP Load Balancing

7.3.4.1Configuration 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). APV appliance chooses a real service according to some policy and method. In redirect mode, APV appliance responds the client with the chosen real server’s URL “rtsp://audio2.array.com:554/test.mp3”. The APV appliance and the client get disconnected, and the client begins to communicate with the real server “audio2.array.com:554” In this mode, all the real servers should have public IP addresses which can be accessible from Internet clients.



7.3.4.1.1.1 RTSP Load Balancing - Redirect Mode

7.3.4.1.1.2 General Settings of RTSP Load Balancing

Operation	Command
Configure real services	**slb real rtsp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** *<group_name>* **{rr|pu|sr}** **slb group method** *<group_name>* **lc** [threshold_granularity] [yes|no] **slb group method** *<group_name>* **pi** *[hash_bits] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **snmp** *[snmp_mode] [community] [oid_count] [oid1] [oid1_weight] [oid2] [oid2_weight] [check_interval]*
Add the real servers into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]*
Bind the group to	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}*

Operation	Command
the virtual service	**slb policy static** *<virtual_service> <real_service>* **slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy filetype** *<policy_name> <vs_name> <group_name > <filetype>*

Configuration Example for Redirect Mode via CLI

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]".

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

Define RTSP real service groups.

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

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

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

Add the real services into the groups.

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

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

Define an RTSP virtual service.

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

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

Define a filetype policy to choose a group by file extension.

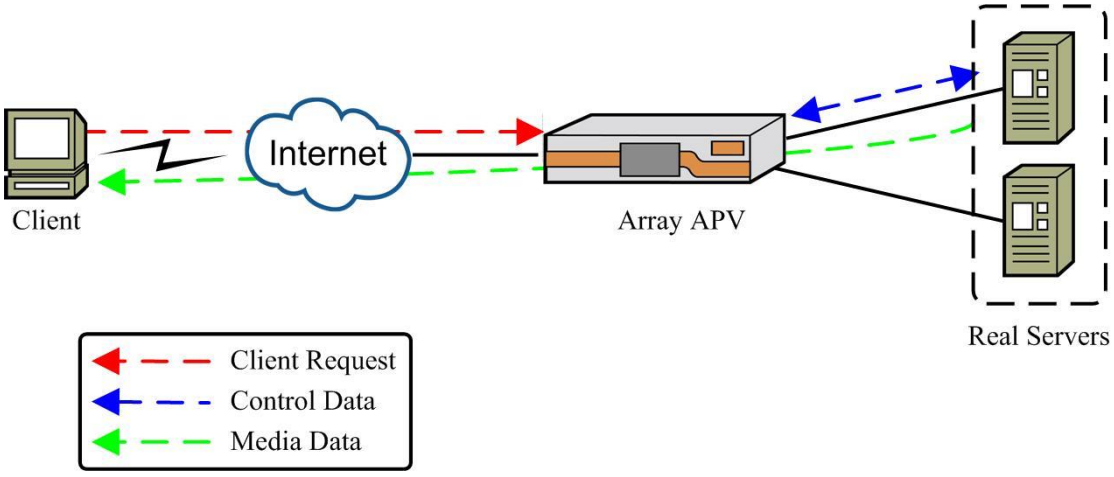
If you add default policy, you will choose that group when you can not find available real services by filetype policy.

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

7.3.4.2Configuration in Dynamic NAT Mode

Configuration Guidelines

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



7.3.4.2.1.1.1 RTSP Load Balancing - Dynamic NAT Mode

7.3.4.2.1.2 General Settings of RTSP Load Balancing

Operation	Command
Configure real services	**slb real rtsp** <*real_name*> <*ip*> *[port] [max_conn] [rtsp-tcp|tcp|icmp|script-tcp|script-udp|dns] [hc_up] [hc_down] [timeout]*
Define group methods	**slb group method** <*group_name*> **{rr|pu|sr}** **slb group method** <*group_name*> **lc** *[threshold] [yes|no]* **slb group method** <*group_name*> **pi** *[hash_bits] [rr|sr|lc|lb] [threshold]* **slb group method** <*group_name*> **hi** *[hash_bits]* **slb group method** <*group_name*> **chi** *[hash_bits]* **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]*

Operation	Command
	slb virtual rtsp *<virtual_name> <vip> [vport] [mode] [arp|noarp] [max_conn]* **slb virtual {ftp|ftps}** *<virtual_name> <vip> [vport] [max_conn]*
Bind the group to the virtual service	**slb policy default** *{virtual_name|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_name> <real_name>* **slb policy backup** *{virtual_name|vlink_name} {group_name|vlink_name}* **slb policy filetype** *<policy_name> <vs_name> <group> <filetype>*

Configuration Example via CLI

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

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

Define RTSP real service groups.

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

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

Add the real services into the group.

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

Define RTSP virtual services.

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

Set the policy.

Static policy has higher priority than the default policy.

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

7.3.5 Layer 2 IP/MAC-based Load Balancing

7.3.5.1Two-arm Network Topology

Configuration Guidelines

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

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

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

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

Client route:

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

Server route:

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

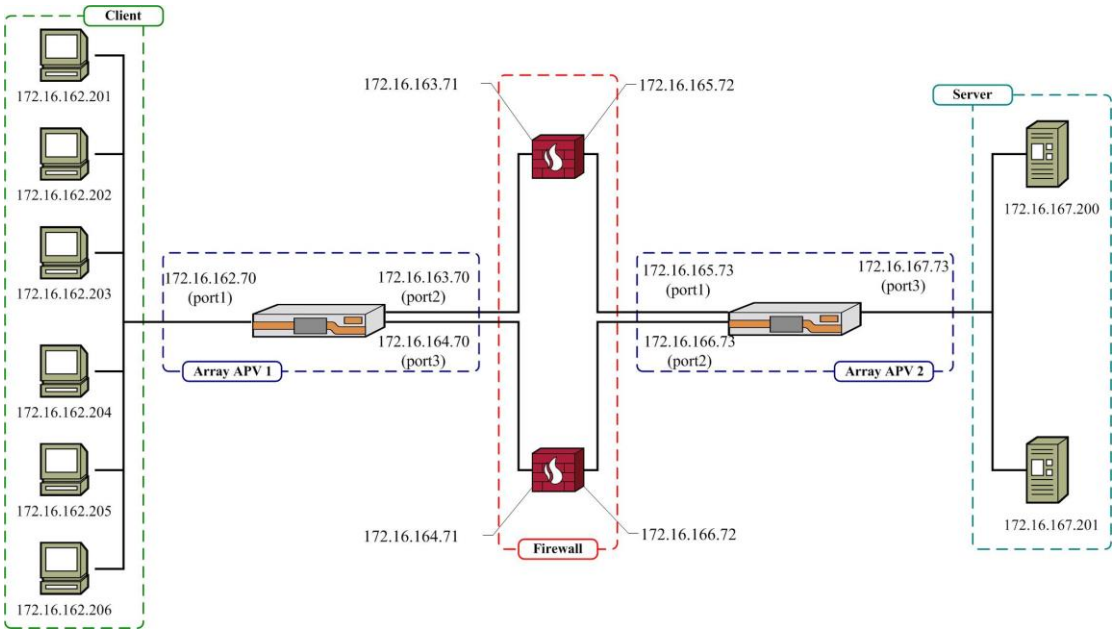
Firewall1 routes:

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

Firewall2 routes:

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

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



7.3.5.1.1.1 Layer 2 IP/MAC-based Load Balancing - Two-arm Network

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

7.3.5.1.1.2 General Settings of Layer 2 IP/MAC-based Load Balancing

Operation	Command
Configure real services	**slb real l2ip** *<real_service> <real_ip>* **slb real l2mac** *<real_service> <real_mac> <output_interface>*
Define group methods	**slb group method** *<group_name>* **{hi|rr|chi}** *[hash_bits]*
Add the real servers into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
Define virtual services	**slb virtual l2ip** *<virtual_service> <vip> [gateway_ip]*
Bind the group to the virtual service	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}*
Define the additional health check	**slb real health** *<add_hc_name> <real_service> <ip> <port> [hc_type] [hc_up] [hc_down]*
Configure reflector	**health ipreflect** *<reflector_name> <ip_address> <port> [protocol]*

Configuration Example via CLI

Then we begin to configure the left box APV1 according to the above figure.

1. Assign IP address to relative interfaces.

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

Define Layer 2 real services.

We can use IP address to define a real service.

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

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

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

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

Define the group for the real service and add its members to the group.

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

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

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

Add the real services to the group.

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

Define Layer 2 virtual service.

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

Define the SLB policy.

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

Note: Layer 2 SLB only supports default policy.

Add the additional health check on the backend server.

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

Here, we have finished the configuration on APV1 for the Layer 2 IP/MAC based SLB. Now we will begin to configure APV2:

1. Assign IP address to relative interfaces.

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

Define Layer 2 real services.

```
AN(config)#slb real l2ip rs1 172.16.165.72
AN(config)#slb real l2ip rs2 172.16.166.72
```

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

Define the group for the real service.

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

Add its members to the group.

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

Define Layer 2 virtual service.

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

Define the SLB policy.

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

Add the additional health check on the backend server.

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

Configure reflector for Layer 2 SLB TCP health check.

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

7.3.5.2Single-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:

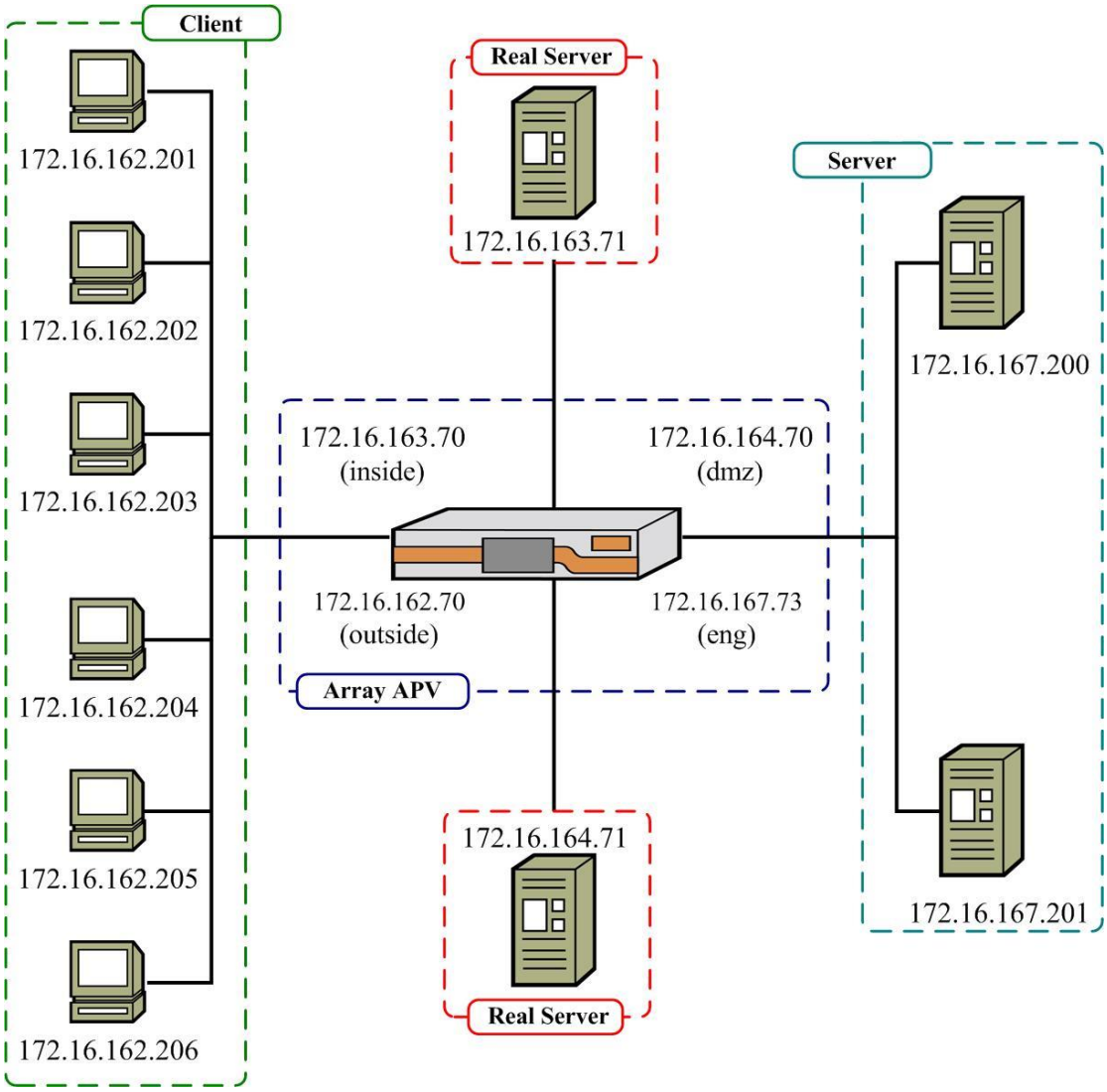
```
route ADD 172.16.162.0 MASK 255.255.255.0 172.16.167.73
route ADD 172.16.163.0 MASK 255.255.255.0 172.16.167.73
route ADD 172.16.164.0 MASK 255.255.255.0 172.16.167.73
```

Firewall1 route:

```
route add default 172.16.163.70
```

Firewall2 route:

```
route add default 172.16.164.70
```



7.3.5.2.1.1.1 IP/MAC-based Load Balancing - Single-arm Network

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

Configuration Example via CLI

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

1. Assign IP address to relative interfaces.

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

Define Layer 2 real services.

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

Define the group for the real service.

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

Add the real services into the group.

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

Define Layer 2 virtual service.

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

Define the SLB policy.

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

Add the additional health check on the backend server.

```
AN(config)#slb real health a1 r1 172.16.163.71 0 icmp
AN(config)#slb real health a1 r2 172.16.164.71 0 icmp
or
AN(config)#slb real health a1 r1 172.16.163.70 0 icmp
AN(config)#slb real health a1 r2 172.16.164.70 0 icmp
```

7.3.5.3Local Exception IP Configuration for Layer 2 SLB

Generally, in a scenario where L2 SLB is applied, the system will forward packets whose destination IP is not a local IP address. Packets whose destination IP matches a local IP (such as VIP, interface IP, etc.) will skip the L2 SLB and the L4/L7 load balancing will be performed.

The APV appliance supports local exception IP configuration, allowing the local IP (such as VIP, interface IP, etc.) that requires orchestration to be configured as an exception IP for L2 SLB, so that the system can still perform L2 load balancing on it and forward its traffic. After a local exception IP is configured, when a packet whose destination IP matches the configured local IP hits the L2IP virtual service, the packet will still be forwarded by L2 SLB (such as forwarding to a security device), and then be returned to the APV appliance to accept the L4/L7 load balancing.

- **Configuration Example via CLI**

Add a local exception IP for the specified L2IP virtual service.

```
AN(config)#slb l2vs ipexception 172.16.163.71 vs1
```

7.3.6 Layer 3 IP-based Load Balancing

7.3.6.1Configuration Guidelines

The commands used to configure Layer 3 SLB are summarized in the following table:

7.3.6.1.1.1 General Settings of Layer 3 IP-based Load Balancing

Operation	Command
Configure real services	**slb real ip** *<real_service> <ip> [max_connection] [hc_type] [hc_up] [hc_down] [udp_timeout]*
Define group methods	**lb group method** *<group_name>* **{rr|sr}** **slb group method** *<group_name>* **lc** *[threshold_granularity] [yes|no]* **slb group method** *<group_name>* **pi** *[hash_bits] [first_choice_method] [threshold_granularity]* **slb group method** *<group_name>* **hi** *[hash_bits]* **slb group method** *<group_name>* **chi** *[hash_bits]* **slb group method** *<group_name>* **snmp** *[snmp_mode] [community] [oid_count] [oid1] [oid1_weight] [oid2] [oid2_weight] [check_interval]*
Add the real servers into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
Define virtual services	**slb virtual ip** *<virtual_service> <vip> [max_connection]*
Bind the group (or the real service) to the virtual service	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}*

7.3.6.2Configuration Example via CLI

1. Create real services.

```
AN(config)#slb real ip rip0 10.3.14.10
```

```
AN(config)#slb real ip rip1 10.3.14.20 1000 none
```

The health check of rip0 defaults to “icmp”.

Define a group for Layer 3 load balancing by using the “**slb group method**” command.

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

Add the real services into the group.

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

Create a virtual service.

```
AN(config)#slb virtual ip vip0 10.3.14.56
```

Associate a group with a virtual service.

You may associate the group or the real service to the virtual service of Layer 3 load balancing by using the command “**slb policy default**” or associate the real service to the virtual service by the command “**slb policy static**”.

```
AN(config)#slb policy default vip0 gip0
Or
AN(config)#slb policy static vip0 rip0
```

7.3.7 Port Range Load Balancing

7.3.7.1Configuration Guidelines

The commands used to configure Port Range SLB are summarized in the following table:

7.3.7.1.1.1 General Settings of Port Range Load Balancing

Operation	Command
Configure real services	**slb real tcp** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real udp** *<real_service> <ip> <port> [max_connection] [hc_up] [hc_down] [timeout] [hc_type]* **slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real tcps** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **slb real dns** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]*
Define group	**slb group method** *<group_name> [method]*

Operation	Command
methods	
Add the real servers into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]*
Define the port range of a virtual service	**slb virtual portrange** *<virtual_service> <start_port> <end_port> [protocol] [port_type]*
Bind the group (or the real service) to the virtual service	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy backup** *{virtual_service|vlink_name} {group_name|vlink_name}*

7.3.7.2Configuration Example via CLI

Herein we use HTTP protocol as an example. The configurations of other protocols are similar.

1. Define static or port range real services.

When the real services are static:

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

The port of HTTP type real service defaults to 80.

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

The port of “rhttp1” is specified to 90.

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

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

Define a group.

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

Add the real services into the group.

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

Create a virtual service.

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

Define the port range for the virtual service.

At most three port ranges can be defined for an SLB virtual service.

```
AN(config)#slb virtual portrange vhttp0 80 90
AN(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.

Associate a group or a real service with a virtual service.

Associate the group to the port-range virtual service with the command "**slb policy default**" and associate the real service to the port-range virtual service with the command "**slb policy static**".

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

7.3.8 Terminal Server Load Balancing

7.3.8.1Configuration Guidelines

The commands used to configure Terminal Server Load Balancing are summarized in the following table:

7.3.8.1.1.1 General Settings of Terminal Server Load Balancing

Operation	Command
Create RDP real services	**slb real rdp** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]*
Create RDP groups	**slb group method** *<group_name>* **rdprt** *[first_choice_method]*
Add the real services into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
Create RDP virtual services	**slb virtual rdp** *<virtual_service> <vip> [vport] [arp_support] [max_connection]*
Associate the real server group with virtual services	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}*

7.3.8.2Configuration Example via CLI

1. Create RDP real services.

The default port number for RDP real services is 3389.

```
AN(config)#slb real rdp rs1 172.16.69.191 3389 1000 icmp 3 3
AN(config)#slb real rdp rs2 172.16.69.192 3389 1000 icmp 3 3
```

Note: For the RDP real services, only the “icmp” and “tcp” types of health check can be used.

Create RDP groups.

```
AN(config)#slb group method g1 rdprt rr
```

Add the real service into the group.

```
AN(config)#slb group member g1 rs1
AN(config)#slb group member g1 rs2
```

Create RDP virtual services.

The default port number for RDP virtual services is 3389.

```
AN(config)#slb virtual rdp vs1 172.16.69.171 3389 arp 0
```

Associate the RDP group with the virtual services.

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

Note: RDP only supports the Default group policy.

7.3.9 Policy Nesting

7.3.9.1Configuration Guidelines

The commands used to configure Policy Nesting are summarized in the following table:

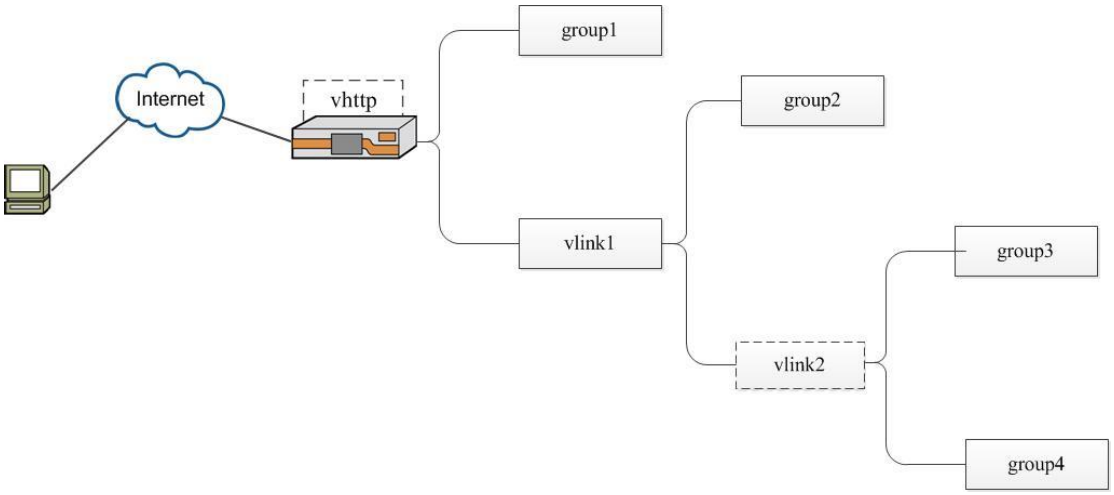
7.3.9.1.1.1 General Settings of Policy Nesting

Operation	Command
Configure real services	**slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]*
Define group methods	**slb group method** *<group_name> [method]*

Operation	Command
Add the real servers into the group	**slb group member** *<group_name> <real_service> [weight|cookie_value|url_tag_value] [priority]*
Define virtual services	**slb virtual {http|https|dns|siptcp|sipudp}** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb virtual {tcp|tcps|udp}** *<virtual_service> <vip> <vport> [arp_support] [max_connection]* **slb virtual rtsp** *<virtual_service> <vip> [vport] [mode] [arp_support] [max_connection]* **slb virtual {ftp|ftps}** *<virtual_service> <vip> [vport] [max_connection]*
Create the vlink	**slb vlink** *<vlink_name>*
Bind the group to the virtual service or the vlink	**slb policy default** *{virtual_service|vlink_name} {group_name|vlink_name}* **slb policy static** *<virtual_service> <real_service>* **slb policy icookie** *<policy_name> {virtual_service|vlink_name} <group_name> <precedence>* **slb policy rcookie** *<policy_name> {virtual_service|vlink_name} <group_name> <precedence>* **slb policy persistent cookie** *<policy_name> {virtual_service|vlink_name} <group_name> <cookie_name> <precedence>* **slb policy persistent url** *<policy_name> {virtual_service|vlink_name} <group_name> <url_tag> <precedence>*

7.3.9.2Configuration Example via CLI

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



7.3.9.2.1.1.1 Policy Nesting

1. Define the real services.

Six HTTP real services are configured in this example:

```
AN(config)#slb real http "rhttp1" 172.16.85.109 80 1000 http 3 3
```

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

Create the groups.

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

Add the real services into the groups.

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

Create a virtual service.

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

Create the vlinks.

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

Associate a group to a virtual service or a vlink.

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

7.3.10SLB Session Persistence Configuration

7.3.10.1 Independently Applying the Persistence Method for Session Persistence (ip)

➢ **Configuration purpose:**

To implement session persistence by independently using the persistence method with the client IP address as the session ID. Once configured, the APV appliance will:

- Send the first access request from a client to the real server selected by the first choice method (rr in this example).
- Obtain and record the client IP address as the unique session ID.
- Send subsequent requests from the client to the same real server. If the client remains idle (for example, no new client requests are detected) for 5 minutes, the APV appliance will terminate the client's session and clear the session ID.

The detailed configuration procedure is as follows:

➢ **Prerequisites:**

- Layer 4 or 7 real services r1 and r2 are already defined. In this example, real services are of the HTTP type.
- A Layer 4 or 7 virtual service v1 is already defined. In this example, the virtual service is of the HTTP type.

➢ **Configuration example via CLI:**

1. Execute the following command to configure the service group and persistence method:

slb group method <*group_name*> **persistence** <*session_id_type*> *[rr|sr|lc] [threshold]*

For example:

AN(config)#**slb group method g1 persistence ip rr**

Execute the following command to add real services to the service group:

slb group member <*group_name*> <*real_name*> *[weight]*

For example:

AN(config)#**slb group member g1 r1 1 0** AN(config)#**slb group member g1 r2 1 0**

Execute the following command to bind the service group and virtual service with the default policy:

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

For example:

AN(config)#**slb policy default v1 g1**

Execute the following command to configure the time out mode:

slb persistence timeout <*timeout _ minutes*> *[group_name] [idle|timeout]*

For example:

AN(config)#**slb persistence timeout 5 g1 idle**

7.3.10.2 Independently Applying the Persistence Method for Session Persistence (string)

➢ **Configuration purpose:**

To implement session persistence by independently using the persistence method with a specified string (obtained from the HTTP response cookie) as the session ID. Once configured, the APV appliance will:

- Send the first access request from a client to the real server selected by the first choice method (rr in this example).
- Obtain the "mycookie" value from the real server response and record it as the session ID.
- When receiving subsequent requests from the client, check whether the value of "telnum" in the request URL Query matches with the session ID. If yes, send the request to the same real server. If the client remains idle (for example, no new client requests are detected) for 5 minutes, the APV appliance will terminate the client's session and clear the session ID.

The detailed configuration procedure is as follows:

➢ **Prerequisites:**

- Layer 4 or 7 real services r1 and r2 are already defined. In this example, the virtual service is of the HTTP type.
- A Layer 4 or 7 virtual service v1 is already defined. In this example, the virtual service is of the HTTP type.

➢ **Configuration example via CLI:**

1. Execute the following command to configure the service group and persistence method:

slb group method <*group_name*> **persistence** <*session_id_type*> *[rr|sr|lc] [threshold]*

For example:

AN(config)#**slb group method g1 persistence string rr**

Execute the following command to add real services to the service group:

slb group member *<group_name> <real_name> [weight]*

For example:

AN(config)#**slb group member g1 r1 1 0** AN(config)#**slb group member g1 r2 1 0**

Execute the following commands to obtain the value of "mycookie" in the response from the real server as the session ID:

slb group persistence request urlquery *<group_name> <query_name>* **slb group persistence response cookie** *<group_name> <cookie_name>*

For example:

AN(config)#**slb group persistence request urlquery g1 telnum** AN(config)#**slb group persistence response cookie g1 mycookie**

Execute the following command to bind the service group and virtual service with the default policy:

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

For example:

AN(config)#**slb policy default v1 g1**

Execute the following command to configure the time out mode:

slb persistence timeout *<timeout_minutes> [group_name] [idle|timeout]*

For example:

AN(config)#**slb persistence timeout 5 g1 idle**

Note: Before configuring the APV to implement session persistence based on the HTTP cookie, you need to specify the cookie in the real service configurations to ensure that the HTTP response can carry the corresponding cookie.

7.3.10.3 Persistence Method Collaborating with an SLB Persistence Policy

- **Configuration purpose:**

Implement session persistence by applying both the persistence method and a Layer 7 SLB persistence policy. In this case, the session ID is obtained through the policy.

The following table describes the configuration for the method and policy:

7.3.10.3.1.1 Configurations of the Method and Policy

Item	Configurations
Policy	• The header policy is configured to obtain the content of "x-up-calling-line-id" in the HTTP request header as the target string. • The default policy is configured.
Persistence method	• The first choice method is round robin (rr). • The session ID is of the string type. • Whether to obtain the session ID from the client request or server response is not specified. • The offset and ID length for obtaining the session ID are specified.

Once configured, the APV appliance will:

- If the request from the client matches with the header policy, the APV appliance will obtain the session ID based on the offset and ID length from the content of "x-up-calling-line-id".
- If the request from the client matches with the default policy, the APV appliance will use the first choice method to direct the request to a real server and not implement session persistence.

➢ **Prerequisites:**

- Layer 4 or 7 real services r1 and r2 are already defined. In this example, the real services are of the HTTP type.
- A Layer 4 or 7 virtual service v1 is already defined. In this example, the virtual service is of the HTTP type.

➢ **Configuration example via CLI:**

1. Execute the following command to configure the service group and persistence method:

slb group method <*group_name*> **persistence** <*session_id_type*> *[rr|sr|lc] [threshold]*

For example:

AN(config)#**slb group method g1 persistence string rr**

Execute the following command to add real services to the service group:

slb group member <*group_name*> <*real_name*> *[weight]*

For example:

AN(config)#**slb group member g1 r1 1 0**
AN(config)#**slb group member g1 r2 1 0**

Execute the following command to bind the service group and virtual service with the header and default policies:

slb policy header *<policy_name> {virtual_name|vlink_name} {group_name|vlink_name} <header_name> <header_string> <precedence>*
slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

For example:

```
AN(config)#slb policy header p1 v1 g1 "x-up-calling-line-id" "^1" 1
AN(config)#slb policy default v1 g1
```

Execute the following command to configure the offset and length for the session ID:

slb group persistence value *<group_name> <offset> [session_id_length]*

For example:

```
AN(config)#slb group persistence request header g1 abc user y 0
AN(config)#slb group persistence value g1 4 3
```

Execute the following command to configure the time out mode:

slb persistence timeout *<timeout_minutes> [group_name] [idle|timeout]*

For example:

```
AN(config)#slb persistence timeout 5 g1 idle
```

7.4 SLB Summary

SLB Type	Priority (1 is the highest)	Virtual Service	Real Service	Health check	Scenarios
Layer 7 HTTP/HTTPS	2	IP + Port + proto (HTTP, HTTPS)	IP + Port + proto (HTTP, HTTPS)	None HTTP HTTPS TCP TCPS ICMP Additional Script	1. Balance traffic according to application protocol headers, for example HTTP headers 2. Cache feature is needed
Layer 7 DNS	2	IP + Port + proto (DNS)	IP + Port + proto (DNS)	None DNS ICMP Additional Script	DNS requests DNS cache feature can be applied for better performance
Layer 7 FTP	2	IP + Port + proto (FTP)	IP + Port + proto (FTP)	None TCP ICMP Additional Script	FTP traffic
Layer 7 SIP	2	IP + Port + proto (SIP-TCP, SIP-UDP)	IP + Port + proto (SIP-TCP, SIP-UDP)	None TCP TCPS ICMP Additional Script SIP-TCP SIP-UDP	Balance VOIP traffic
Layer 7 RTSP	2	IP + Port + proto (RTSP)	IP + Port + proto (RTSP)	None TCP ICMP Additional Script RTSP-TCP	Balance real time media traffic
Layer 4	2	IP + port	IP + Port	None TCP TCPS ICMP Additional Script	1. Balance traffic according to TCP/UDP headers. 2.TCP port or UDP port is specified to determine a particular

SLB Type	Priority (1 is the highest)	Virtual Service	Real Service	Health check	Scenarios
					service
Port range (for Layer 7)	3	Layer 7 VS + Port range	Layer 7 RS Layer 7 RS (0 port)	Non-zero port RS: Layer 7 health check Zero port RS: ICMP Additional	In addition to Layer 7 SLB, cross-port and dynamic port application traffic balance is supported
Port range (for Layer 4)	3	Layer 4 VS + Port range	Layer 4 RS Layer 4 RS (0 port)	Non-zero port RS: Layer 4 health check Zero port RS: ICMP Additional	In addition to Layer 4 SLB, cross-port and dynamic port application traffic balance is supported
Layer 3	4	IP	IP	None ICMP Additional	In addition to port range SLB, cross-protocol application traffic balance is supported. Currently, only TCP and UDP protocol are supported
Layer 2	1	IP + port ranges	IP, MAC	ARP Additional (only ICMP)	1. The backend real services do not have usable IP addresses so that the traffic cannot be balanced according to IP addresses; 2. The backend real services are not the destination of the input traffic (for example, virus scanners check every packet before forwarding it to the real destination).

		Basic Methods						IP-based Methods				Header/Request-based Methods										Cookie-based Methods					General	Others					
		grr	rr	lc	snmp	sr	lb	pi	hi	hip	chi	hh	ph	chh	pto	po	hq	salsid	sipcid	sipuid	sipcidps	sipuidps	diametersid	rc	ec	ic	pc	hc	persistence	tuxedo	rdprt	radchs	radchs	radpsun	radchi
Basic Policies	Static	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕
	Default	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	★	★
	Backup	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	★	★
Persistent Policies	Persistent URL	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	✕	✕	✕
	Hash URL	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	✕	✕	★	★	✕	✕	✕	✕	✕
	Persistent Cookie	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	★	★	✕	✕	✕	✕	✕	✕
	Rewrite Cookie	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕
	Insert Cookie	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	✕	✕	✕	✕	✕	✕	✕
	Radname	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	★	✕	✕	✕
	Radsid	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	✕	★	✕	✕
	SIP Domain	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	★	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕
	SIP Call	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	★	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕
	Filetype	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	✕
	DNSSEC	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	✕	✕	✕	✕
QoS Policies	QoS Cookie	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕
	QoS Hostname	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	✕	✕	★	★	✕	✕	✕	✕	✕
	QoS URL	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	✕	✕	★	★	✕	✕	✕	✕	✕
	QoS Network	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	✕	✕	★	★	✕	✕	✕	✕	✕
	QoS Clientport	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	✕	✕	★	★	✕	✕	✕	✕	✕
	Regex	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	✕	✕	★	★	✕	✕	✕	✕	✕
	Header	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕	★	★	★	★	★	★	✕	✕	★	★	✕	✕	✕	✕	✕
	QoS Dnsdomain	★	★	★	✕	★	★	★	★	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	✕	✕	✕	✕
	QoS Body	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	★	★	★	★	★	★	★	★	★	★	✕	✕	✕	✕	✕
	QoS Diameter	★	★	★	★	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕
	QoS Diameterappid	★	★	★	★	★	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕
Others	Redirect	★	★	★	✕	★	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕	✕

grr-Global Round Robin
rr-Round Robin
lc-Least Connections
snmp-SNMP
sr-Shortest Response
lb-Least Bandwidth

pi-Persistent IP
hi-Hash IP
hip-Hash IP and Port
chi-Consistent Hash IP

hh-Hash Header
ph-Persistent Hostname
chh-Consistent Hash Header
pto-Persistent TCP Option
po-Persistent URL
hq-Hash Query

salsid-Persistent SSL SID
sipcid-SIP Call ID
sipuip-SIP User ID
sipcidps-SIP CallID Persistence Session
sipuidps-SIP UserID Persistence Session
diametersid-Persistent Diameter SID

rc-Rewrite Cookie
ec-Embed Cookie
ic-Insert Cookie
pc-Persistent Cookie
hc-Hash Cookie
persistence-Persistence

tuxedo-Tuxedo
rdprt-RDP Routing Token
radchs-Consistent Hash RADIUS Username
radchs-Consistent Hash RADIUS Session ID
radpsun-RADIUS Persistence Session by Username
radchi-RADIUS Consistent Hash IP

★ The policy can collaborate with the group method.表示策略可以和该算法配合使用。
✕ The policy cannot collaborate with the group method.表示策略不能和该算法配合使用。

Note: After enabling the DirectFWD function, the SR method will stop to take effect. All the other Layer 4 SLB methods are supported.

8 Application Security Orchestrator

8.2 Overview

In the traditional series-type connection architecture, all network security devices are deployed in the Internet access area in the form of physical series or logical series, and provide services in the mode of active and standby deployment. The traditional network security architecture has the following problems:

- Tight coupling of security architecture and network architecture: To meet the new security needs, the existing network architecture needs to be adjusted constantly, while the transformation cost is highand the current architecture can notmeet the needs of business changesor provide differentiated and personalized security policies for different businesses.
- Performance bottleneck of SSL encryption and decryption: With the increasing proportion of encrypted traffic, security devices need to rely on their own encryption and decryption functions to carry out seven layers detection and defense, which not only increases the burden of security devices, but also leads to low efficiency of business access and time delay.
- Low utilization rate of security resources and poor scalability: All traffic pass through all security devices, resulting in serious waste of resources. In addition, security resources cannot be pooled, and performance expansion can only be carried out by replacing the device.
- High investment and difficult maintenance: Users usually need to purchase a large number of high-end devices to meet security and performance needs, and the later capacity expansion and maintenance costs are high. With the increase of security devices in series, the cost of handling network fault raises.

The application security orchestration function aims to solve the problems and disadvantages of the traditional serial security architecture, provide intelligent orchestration scheme, realize personalized and customized orchestration based on the user's security policy requirements, and provide a series of functions for the security architecture, such as resource pooling, load balancing, SSL encryption and decryption, traffic visualization, health check, fault tolerance and so on. This function has the following values:

- Improve the utilization efficiency of security devices and the efficiency of application access;
- Improve the flexibility of security resources to meet the demand of flexible expansion and contraction;
- Realize personalized security policies customization, so that security and other resources can provide better service to meet business differentiation needs;
- Realize device security fault tolerance, decouple with security business, and reduce the impact of fault business.

8.3 Deployment Scenarios

As an orchestrator, APV is connected to the network as a three-layer network device, which arranges and processes the traffic (requests and responses) routed by upstream and downstream devices, and then sends it according to the routing rules. The orchestrator supports personalized and customized orchestration based on the security policy requirements of users. The orchestrator supports three deployment scenarios:

- **Inbound Proxy**

The orchestrator can provide virtual services for client access, perform load balancing on the back-end server, and handle plaintext or SSL traffic.

In this scenario, the orchestrator supports:

- Traffic encryption and decryption
- Traffic load balancing
- Forwarding traffic to different security service chains for processing according to security requirements.

- **Inbound transparency**

The orchestrator is deployed as a three-layer gateway. The client accesses external resources and completes DNS resolution. The client can handle plaintext or SSL traffic, and can be used in combination with NAT and outbound LLB.

In this scenario, the orchestrator supports:

- Traffic encryption and decryption
- Forwarding traffic to different security service chains for processing according to security requirements.

- Outbound transparency

The orchestration device is deployed as a three-layer gateway. The client accesses external resources and completes DNS resolution. The client can handle plaintext or SSL traffic, and can be used in combination with NAT and outbound LLB.

In this scenario, the orchestration function supports:

- Traffic encryption and decryption (through SSL interception)
- Forwarding traffic to different security service chains for processing according to security requirements.

8.4 Functional Principle

The orchestration function is mainly realized through traffic listener, security service chain, security service, security device, orchestration policy, SSL encryption and decryption and health check.

8.4.2 Traffic Listener

The traffic listener is composed of a group of uplink and downlink interfaces, which serves as the starting and ending points of orchestrator.

8.4.3 Security Service Chain

The security service chain defines the path of security services through which the traffic needs to pass in turn. It can include L2 security services, L3 security services, L4 security services and TAP services. When the traffic hits the security service chain, it will be sent to each security service in turn in the service chain for processing, such as external SSL encryption and decryption, security check, security audit and so on.

8.4.4 Security Service

Security service is a component of the security service chain, which is usually provided by one or more security devices. When the traffic hits the security service, it supports the balanced load of traffic to multiple security devices. When the security service is unavailable, it supports the processing of denying client access or bypassing the security service. According to the type, security services can be classified as L2 security services, L3 security services, L4 security services and TAP service.

8.4.5 Security Devices

Security devices are devices that actually handle traffic safely and are components of security services. According to the deployment mode, security devices can be classified into L2 security devices, L3 security devices and L4 security devices.

Note: the TAP security service only supports one TAP device, which will be configured together when defining the TAP service.

8.4.6 Orchestration Policy

The orchestration policy defines the path the traffic needs to pass through. The system supports configuring security policies for traffic listeners, security service chains, and real service groups.

8.4.6.1Orchestration Policy for Traffic Listeners

According to this kind of orchestration policy, the traffic received by the traffic listener will be distributed to the specified security service chain or virtual service, or be blocked.

In orchestration scenario, a policy must be configured for the traffic listener, otherwise the traffic received by the traffic listener will be processed as normal business.

8.4.6.2Orchestration Policy for Security Service Chain

According to this kind of orchestration policy, the traffic processed by the security service chain will be distributed to the next security service chain or virtual service. This kind of orchestration policy does not support blocking traffic.

8.4.6.3Orchestration Policy for Real Service Group

This kind of orchestration policy distributes the traffic that hits the real service group to other security service chains. This kind of orchestration policy is usually used to connect the previous orchestration policy whose distribution object is a virtual service.

8.4.6.4Network Orchestration Rules

Network orchestration rules define the hit conditions of orchestration policy. When the five-tuples of traffic (source IP, source port, destination IP, destination port and protocol) match the orchestration rules, the traffic will hit the orchestration policy that associated with the orchestration rules.

8.4.6.5Orchestration Policy Priority

The higher the priority value of the orchestration policy, the higher the priority when matching.

8.4.7 Encryption and Decryption of Secure Traffic

In the inbound deployment scenario, the orchestrator supports employing the SLB&SSL function of the device to encrypt and decrypt traffic. For the inbound proxy, it also supports the encryption and decryption of traffic by external SSL gateway devices.

To apply an external SSL gateway device for encryption and decryption, the external gateway should be defined as an L4 security service.

In the outbound deployment scenario, the orchestration function supports the use of SSL listening function to encrypt and decrypt traffic.

8.4.8 Health Check

The orchestration function supports health check for L2/L3 security devices. Health check for L4 security appliances uses the health check configuration for the related TCP-type SLB virtual service.

The health check mode supported by L2 security device is bi-direct mode. In this mode, the health check traffic passes through the security devices and through the uplink and downlink at the same time.

The health check methods supported by L3 security device are:

- Bi-direct mode: In the bi-direct mode, the traffic for health check pass through the security device and through the uplink and downlink at the same time. Bidirect mode supports TCP-type health check.
- Uni-direct mode: In the unidirect mode, the traffic for health check is forwarded to the security device and pass through uplink and downlink respectively.

Unidirect mode supports ICMP, TCP, Half-TCP, HTTP and HTTPS health check. The ICMP-type health check examines both uplink and downlink; the TCP, Half-TCP, HTTP and HTTPS health check only examine the uplink.

The Half-TCP health check establish TCP Half-Open connection with the security device to examine its health status. If the Half-TCP health check successfully sends the TCP request and receives the correct response from the device, the device will be reported up.

L3 security device also supports additional health check. Administrator can configure health check for the uplink, downlink or both links as needed. Supported L3 device health check types are ICMP, TCP, Half-TCP, HTTP and HTTPS.

8.5 Configuration Guide

When using the orchestrator, you need to define the traffic path according to the security requirements, draw orchestration topology, and then configure the nodes in the orchestrator topology one by one.

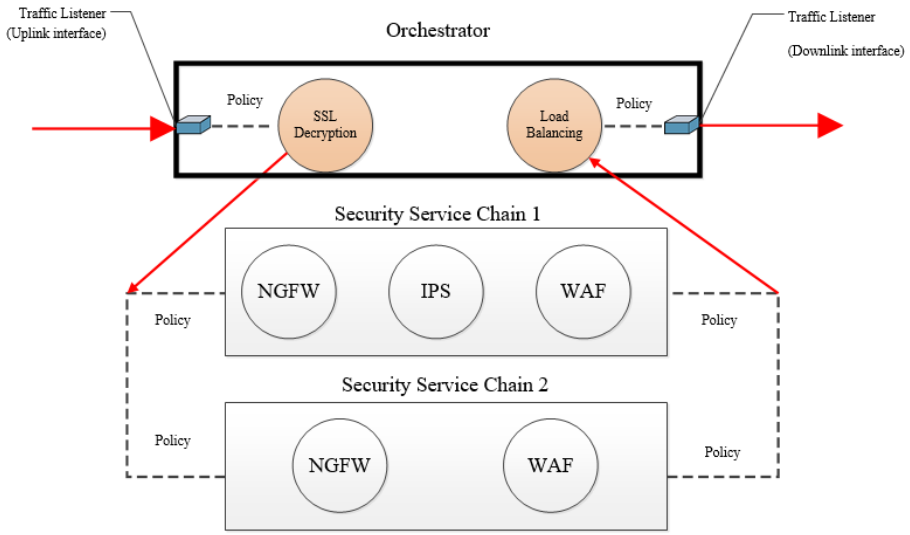


Figure 1-1 **Typical Orchestrator Topology**

In the above figure, the orchestration processes are:

1. After receiving the traffic, the orchestrator forwards the traffic to the SLB virtual service for SSL decryption according to the orchestration policy of the traffic listener. After decryption, the destination IP address of the traffic remains unchanged and the destination port changes.

2. The orchestrator distributes traffic to different security service chains for processing according to different orchestration policies in real service group.

3. After being processed by the security service chain, the orchestrator forwards the traffic to SLB virtual service for load balancing according to the orchestration policy of the security service chain, and then distributes the traffic to the real server according to SLB load balancing policy and method.

In the actual business processing scenario, users can customize the orchestration process according to the actual business needs. For configuration examples, please refer to the "Orchestrator Deployment Guide".

9 Reverse Proxy Cache

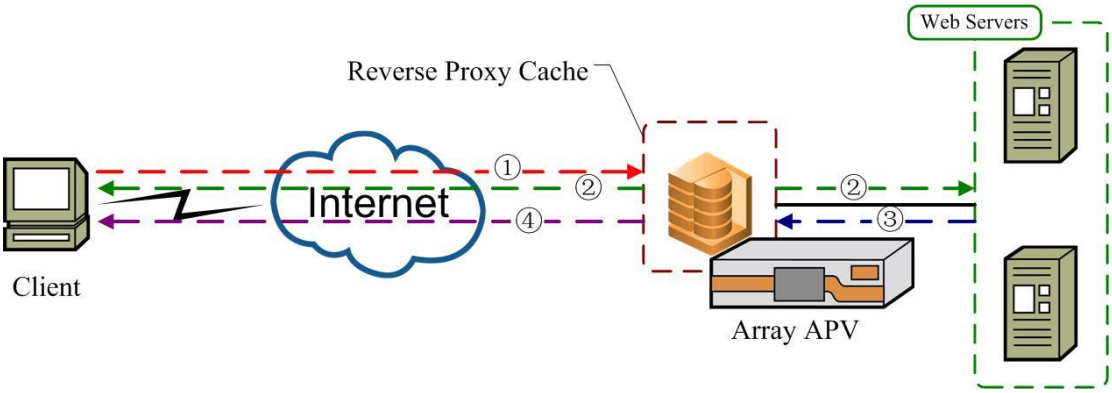
9.2 Overview

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 APV appliance.

9.3 Understanding Reverse Proxy Cache

9.3.2 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.



9.3.2.1.1.1.1 Reverse Proxy Cache Working Mechanism

1. The client sends a request to the APV appliance, requesting for a file on the Web server. The APV appliance will forward the request to the cache module for processing.
2. If the requested content has been cached on the APV appliance and the cache does not expire (cache hit), the APV 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 APV appliance with the requested content.
4. The APV 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 APV 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 APV appliances. See the table below:

9.3.2.1.1.2 Max Size of Cache Object

System Memory	Max Size of Cache Object
2 GB	1024 KB（1 MB）
4 GB	10,240 KB（10 MB）
8 GB	20,480 KB（20 MB）
16 GB/24 GB/32 GB/64 GB	40,960 KB（40 MB）

9.3.3 Advantages of Reverse Proxy Cache

Compared with traditional cache functions, the Reverse Proxy Cache, without making any compromise on the overall stability and performance, provides a smarter, high-efficient and personalized configuration platform for administrators to more flexibly adjust the APV appliances to addressing the demands of different websites. This helps administrators improves the response ability of the websites, reduces the load of Web servers and delivers perfect user experience of visiting the websites.

The Reverse Proxy Cache function is mainly featured with the following advantages:

- **Improved performance**
  - The cache function is an independent module in the ArrayOS. 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 APV to send compressed contents to appropriate clients without having to involve the compression engine. This greatly enhances compression throughput.
- **Outstanding stability**
  - If any error occurs to the cache module, administrators can turn off the module immediately, which will not affect the running of other system functions.
  - All cache tuning parameters now use the "**cache filter**" mechanism, and the global control parameters are reduced. This new approach gives administrators more flexibility and control and minimizes confusion during configuration.
- **Intelligent monitoring**
  - The APV process monitor also monitors the cache (in addition to the reverse proxy). If it detects any issues (or if the cache process crashes), it will restart the cache after appropriate cleanup.
- **Flexible configuration**

- 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 APV is making caching decisions. This should help the customer tune the APV 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.

9.3.4 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.

9.3.4.1Cacheable 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 APV appliance will re-validate the content with the origin server and update the content when it is requested next time.

9.3.4.2Non-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.

9.3.5 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 CLI configurations examples in this chapter and the APV CLI Handbook.

- 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.

9.3.6 Cache Expiration Time

Three types of cache expiration time are involved during the cache process:

- Cache expiration time defined by the cache-related field (the Expires and Cache-Control headers) in the HTTP responses
- Cache expiration time set by the "**cache settings expire**" command
- Cache expiration time specified by the "ttl" parameter in the "**cache filter rule**" command

The three types of cache expiration time are prioritized as follows:

1. The cache expiration time set by the "ttl" parameter in the "**cache filter rule**" command will be preferentially used. If the "ttl" parameter is not specified, the cache expiration time specified by the "**cache settings expire**" command applies.
2. For the cache content that does not match any cache filter rule, the expiration time defined in the cache-related headers of the HTTP response applies.
3. If the cache expiration time is unavailable in the cache-related headers of the HTTP response, the cache expiration time set by the "**cache settings expire**" command applies.

9.3.7 Cache Image Format Optimization

The system supports the cache image optimization function, which compresses the images by converting the image format. When the client requests the image again, the system will reply the image with the optimized format, so as to save the traffic and speed up the Webpage access.

Currently, the system supports format optimization of source images in BMP, JPEG, and PNG formats, and source images will be converted into WebP or JPEG images.

You can set the minimum size of source image allowed to be optimized for this function. Source images smaller than the minimum size will no longer be optimized. You can also configure image optimization exclusion policies to skip the image optimization when clients request specified URLs. When the URL in an HTTP request sent to the specified virtual service matches the policy, the appliance will not optimize the image in the corresponding response.

9.4 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 APV 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.

9.4.2 Configuration Guidelines

9.4.2.1.1.1 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** <*expire_time*>
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]*
View contents of cache objects	**show cache content** <*host_name*> <*url_regex*>
Remove all cache cache objects by force	**clear cache content**
Enable cache filter	**cache filter {on|off}**

Configure cache filter rule	**cache filter rule** <*host_name*> <*url*> <*cache_behavior*>
View cache filter configuration	**show cache filter status**
View all cache filters about the specified host name	**show cache filter hostname** <*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** *[host_name]*
Enable the image optimization function	**cache imgopt {on|off}** *[virtual_service]*
Set minimum size of images allowed to be optimized	**cache imgopt sizelimit** <*min_size*>
Set target image format for image optimization	**cache imgopt format** *[target_format]*
Display image optimization settings	**show cache imgopt settings**
Configure an image optimization exclusion policy	**cache imgopt urlexclude** <*virtual_service*> <*url*>
Delete a specified image optimization exclusion policy	**no cache imgopt urlexclude** <*virtual_service*> <*url*>
View image optimization exclusion policies	**show cache imgopt urlexclude** *[virtual_service]*

Clear all image optimization exclusion policies	**clear cache imgopt urlexclude** *[virtual_service]*

9.4.3 Configuration Example via CLI

The Cache function for each virtual service works independently. By default, the Cache function is turned off. When Cache is turned off, no objects are stored in cache and all requests will go to the backend servers through the server load balancing mechanism.

1. Enable the cache function.

To use cache, we need to first enable the Cache function for the specified virtual service.

In this example, we enable the Cache function for the virtual service "virtual_MOSS".

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

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

```
AN(config)#show cache status
reverse proxy cache:                enable
per-vs status "virtual_MOSS":       enable
```

Configure basic cache settings.

We start to define basic cache rules for APV 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.

```
AN#show cache settings
Cache Configuration:
        Cache Default Expiration:       82800 seconds
        Maximum Cacheable Object Size:  5120 KB
```

The above cache settings are the default values, which are the optimal values. If your application has some special requirements, you can make the above cache settings as your needs determine.

To set the global cache expiration time, we can use the "**cache settings expire**" command. The default value is 82800 seconds (23 hours). The global default expiration time will be used as the expiration time for an object in cache only if it is impossible to calculate the expiration time using the Expiration Model specified in Section 13.2 of RFC 2616.

```
AN(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.

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

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

```
AN(config)#show cache settings
Cache Configuration:
        Cache Default Expiration:          43200 seconds
        Maximum Cacheable Object Size:  1000 KB
```

Configure cache filter.

First, enable the cache filter function by using the command "**cache filter {on|off}** <*virtual_service*>". By default, the cache filter function is disabled.

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

Then, define cache filter rules by using the command "**cache filter rule** <*host_name*> <*url*> <*cache_behavior*>". 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:

```
AN(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**".

```
AN(config)#show cache filter all
cache filter rule "www.xyz.com" "./*.jpg " "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.xyz.com" ".*\.bmp" "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.xyz.com" ".*\.gif" "cache=yes" "urlquery=yes" "ttl=200000"
cache filter rule "www.test.com" "example" "cache=yes" "urlquery=yes" "ttl=150000"
cache filter rule "www.test.com" ".*\.jpg" "cache=yes" "urlquery=yes" "ttl=200000"
```

Configure the cache image optimization function

Use the following commands to configure the image optimization function:

Set the minimum size of the source image allowed to be optimized:

```
AN(config)#cache imgopt sizelimit 5
```

Set the target format for the image optimization function:

```
AN(config)#cache imgopt format webp
```

Enable the image optimization function:

```
AN(config)#cache imgopt on
```

Configure an HTTP image optimization exclusion policy:

```
AN(config)#cache imgopt urlexclude vs01 /request/image
```

Show cache statistics.

Once you've configured your cache functions, the ArrayOS will allow you to view the running status of the appliance as it pertains to the caching requirements you've configured.

```
AN(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
```

10HTTP Content Rewrite

10.2 Overview

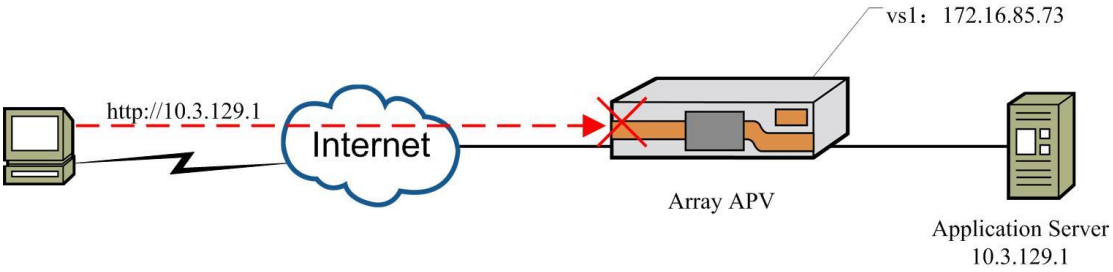
The HTTP Content Rewrite feature allows end users to visit the HTTP contents on the Web servers behind the APV appliance. This feature aims to reduce network latency and improve user experience.

This chapter will cover the theories and configurations of the HTTP Content Rewrite feature.

10.3 Understanding HTTP Content Rewrite

10.3.2How HTTP Content Rewrite Works

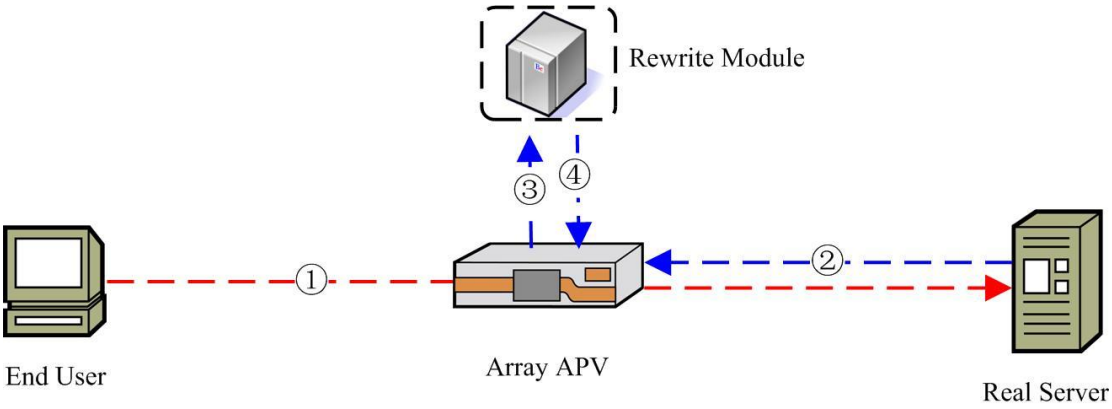
When a company places the APV appliance in front of the application servers, users can access the applications through APV. However, the Web pages that the applications generate contain links with private IP addresses or internal domain names. So if the user fetches the pages with those internal links through APV, the browser will not be able to retrieve the images and other objects on the HTML page since the HTML links pointing to them do not point back to the APV appliance.



10.3.2.1.1.1.1 Users Failed to Access Internal Web Pages

The solution to the above problems is to rewrite contents of the HTML pages before it is sent to the client. To be more precise, the contents of every HTML page will be processed and all the HTML links will be found and rewritten so that the end user can access the pages via a browser. In addition, the rewritten pages can be cached by the APV appliance so that APV does not need to rewrite the same pages over and over again.

Upon receiving the responses from the real server, the APV appliance will communicate with the uproxy module to read the response data that needs to be rewritten, and send these data to the rewrite module for rewriting. The IP addresses and domain names in the files that need rewriting according to configured rules will be rewritten. Finally the APV appliance responds the rewritten data to the end user, and further caches the rewritten data.



10.3.2.1.1.1.2 HTTP Content Rewrite Working Mechanism

1. The end user sends an HTTP request to the real server via the APV appliance.

The response data coming from the real server reaches the APV appliance.

The APV sends the data to the rewrite module.

The rewrite module reads the data that needs rewriting, and rewrites the data.

The APV appliance sends the rewritten data to the end user, and caches the data.

10.3.3Advantages of HTTP Content Rewrite

- **Improve the user experience**

  The HTTP Content Rewrite help external end users from visiting the internal real server hidden behind the APV appliance. It rewrites the Web page file into the valid format to end users. The APV appliance also supports to rewrite the multi-byte character files such as Chinese and Japanese.

- **Reduce the effect to the performance**

  Comparing with the URL rewrite method, the HTTP Content Rewrite feature causes less communication with the real server to reduce the effect to the performance.

- **Decrease the response time**

  In addition, the rewritten pages can be cached by the APV appliance so that APV does not need to rewrite the same pages over and over again. When the client requests the real server for the same Web page, the APV appliance will respond the client with the cached data to decrease the response time.

- **Easy to maintain**

  The HTTP Content Rewrite feature helps enterprise decrease the cost of the human resource. The administrator needs less monitoring because the HTTP Content Rewrite feature rewrites the Web page file by the rewrite driver and the rewrite module automatically.

10.3.4Working Principles of HTTP Content Rewrite

The following introduces the working principles of HTTP Content Rewrite:

➢ Global or per virtual service content rewrite

The administrator can enable/disable the HTTP content rewrite globally or per virtual service.

Note:

- By default, the HTTP content rewrite function is disabled globally, while enabled per virtual service.
- Only with the global HTTP content rewrite enabled, will the per virtual service HTTP content rewrite enabling and configurations take effect.

➢ Define the global content rewrite rule

The HTTP content rewrite function allows administrators to define global rewrite rules to rewrite the IP addresses, domain names or other strings in the Web page files into new strings as pre-defined.

Two kinds of rewrite rules are supported:

- **ProxyHTMLURLMap**

Only rewrite the URLs inside the HTML tags. The other strings will not be rewritten. This kind of rewrite rules can only be applied to rewriting of HTML and XHTML files.

For example:

The source Web page file:

```
<p><a href="10.3.129.1">10.3.129.1</a></p>
<p><a href="http://10.3.129.1/">http://10.3.129.1</a></p>
<p><a href="https://10.3.129.1/">https://10.3.129.1</a></p>
```

The rewritten Web page file:

```
<p><a href="172.16.85.74">10.3.129.1</a></p>
<p><a href="http://172.16.85.74/">http://10.3.129.1</a></p>
<p><a href="https://172.16.85.74/">https://10.3.129.1</a></p>
```

The IP address “10.3.129.1” in the URLs inside the HTML tags has been rewritten into “172.16.85.74”, while others remain unchanged.

- **Substitute**

Rewrite URLs inside and outside the HTML tags.

For example:

The source Web page file:

```
<p><a href="10.3.129.1">10.3.129.1</a></p>
<p><a href="http://10.3.129.1/">http://10.3.129.1</a></p>
<p><a href="https://10.3.129.1/">https://10.3.129.1</a></p>
```

The rewritten Web page file:

```
<p><a href="172.16.85.74">172.16.85.74</a></p>
<p><a href="http://172.16.85.74/">http://172.16.85.74</a></p>
<p><a href="https://172.16.85.74/">https://172.16.85.74</a></p>
```

All the “10.3.129.1” strings have been rewritten into “172.16.85.74”.

Note:

- The configuration strings of the parameter “rule” must be framed in double quotes.
- The configuration strings of “ProxyHTMLURLMap” and “Substitute” are strictly case-sensitive.
- When both “ProxyHTMLURLMap” and “Substitute” rules are configured, the “ProxyHTMLURLMap” rules will be applied first, and then the “Substitute” rules. If the “ProxyHTMLURLMap” and “Substitute” rules have been configured to map to the same regex, the “Substitute” rules will overwrite the “ProxyHTMLURLMap” rules.
- To change the rewrite rules, the currently running rewrite operations will fail, and APV will reset the related connections. Therefore, it is suggested not change the rewrite rules while the APV appliance is processing network traffic.
- If the HTTP content rewrite function is enabled and content rewrite rules are configured, the length of each line in responses cannot be greater than 1MB; otherwise, the APV appliance will send a RST packet to the client to abort the TCP connection.

➢ **Specify the type of the files to be rewritten**

The HTTP Content Rewrite function allows administrators to specify the type of the files to be rewritten. The following are the supported file types:

- text/html
- text/plain
- text/richtext
- text/xml
- application/xml
- application/xhtml+xml
- text/css
- text/javascript
- application/javascript

By default, only the files in “text/html” type are allowed to be rewritten.

- **Define the URL regex for per virtual service HTTP content rewrite**

The administrator can define the URL regex to permit or deny rewriting of the files that match the URL regex per virtual service.

To specify the URL regex, the administrator should first define a URL list, and then add URL regexes into the URL list. The URL regex can be defined as the extension name, the file content or a part of the file name. Then the administrator need to associate the URL list with a virtual service through a permit/deny rule. After all these are done, the files that match any URL regex in the URL list will be rewritten according to the associated permit/deny rules.

Multiple URL regexes can be added into a URL list, in “OR” relationship. That is to say, the permit/deny rule will work as long as any of the extension name, file name or file contents matches the URL regex.

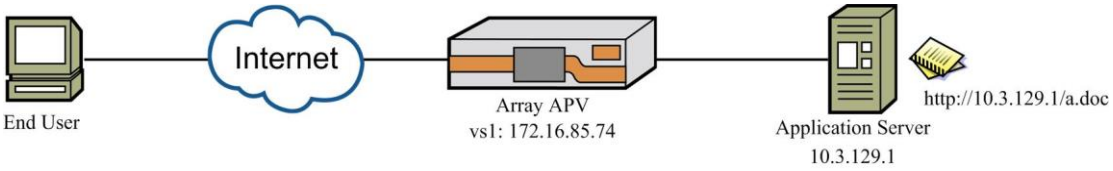
- **Define the HTTP response status code**

The HTTP Content Rewrite function also supports rewriting the Web page files that contain specific HTTP response status code. The “200” HTTP response status code is supported by default. That is to say, the APV appliance will only rewrite the Web page files with the “200” HTTP response status code by default.

10.4 HTTP Content Rewrite Configuration

10.4.2Configuration Guidelines

Before you start to configure HTTP Content Rewrite, please take a moment to familiarize yourself with the network architecture for HTTP Content Rewrite configuration.



10.4.2.1.1.1 HTTP Content Rewrite Topology

As shown above, the real server is hidden behind the APV appliance, and the end user can only access the IP address “172.16.85.74” of the virtual service “vs1”. If the end user wants to visit the Web resource “a.doc” on the real server “10.3.129.1”, the source URL address “http://10.3.129.1/a.doc” should be rewritten into “http://172.16.85.74/a.doc”.

10.4.2.1.1.2 General Settings of HTTP Content Rewrite

Operation	Command
Enable HTTP Content Rewrite	**http rewrite body {on|off}** *[virtual_service]*
Configure HTTP Content	**http rewrite body rule** <*rule*> *[flags]*

Operation	Command
Rewrite rules	
Configure the file type to be rewritten	**http rewrite body mimetype** <*mime_type*>
Add a URL regex into a URL list	**http rewrite body url list** <*url_list*> <*url*>
Configure to permit rewriting the string that matches the URL regex in a URL list	**http rewrite body url permit** <*virtual_service*> <*url_list*>
Configure to deny rewriting the string that matches the URL regex in a URL list	**http rewrite body url deny** <*virtual_service*> <*url_list* >
Configure to rewrite the files that contain specific HTTP response status code	**http rewrite body statuscode** <*status_code*>

10.4.3Configuration Example via CLI

1. Configure the virtual service.

```
AN(config)#slb virtual http "vs1" 172.16.85.74 80 arp 0
```

Enable the HTTP Content Rewrite function.

```
AN(config)#http rewrite body on
```

View the status (enable/disable) of HTTP Content Rewrite.

```
AN(config)#show http rewrite body status
http body rewrite:              enable
per-vs status "vs1":              enable
```

Configure an HTTP Content Rewrite rule to rewrite the IP address “10.3.129.1” into “172.16.85.74”.

```
AN(config)#http rewrite body rule "ProxyHTMLURLMap 10.3.129.1 172.16.85.74" -R
```

View the defined HTTP Content Rewrite rules.

```
AN(config)#show http rewrite body rule
http rewrite body rule “ProxyHTMLURLMap 10.3.129.1 172.16.85.74” “-R”
```

Configure the type of files to be rewritten.

```
AN(config)#http rewrite body mimetype xml
```

11DNS Cache

11.2 Overview

The DNS SLB mechanism used by APV appliance supports DNS cache feature. Upon receiving any type “A” or “AAAA” DNS responses, which are mapping of host names to IP addresses, APV will save them in SLB DNS cache. Then, when the APV appliance receives any DNS requests for the cached “A” or “AAAA” records from clients, the appliance will directly send back the “A” or “AAAA” responses to the clients. If there is no records in cache that hit the requests, the APV appliance will communicate with the remote DNS server(s) directly, and then save the server responses in cache for responding to the coming requests.

11.3 DNS Cache Configuration

11.3.2Configuration Guidelines

11.3.2.1.1.1 General Settings of DNS Cache

Operation	Command
Define related SLB component	**slb real dns** <*real_service*> <*ip*> *[port] [max_connection] [hc_type] [hc_up] [hc_down] [timeout]* **slb virtual dns** <*virtual_service*> <*vip*> *[vport] [arp_support] [max_connection]* **slb policy static** <*virtual_service*> <*real_service*>
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*>

11.3.3Configuration Example via CLI

1. Configure necessary SLB component.

Since DNS cache is interdependent with SLB configuration strategies, please refer to Chapter 7 Server Load Balancing (SLB). Below is a configuration example for DNS cache deployment. First, the SLB component needs to be established.

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

Enable DNS cache.

To enable DNS cache, the "**dns cache {on|off}**" command should be used. The DNS cache is disabled by default.

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

Configure the DNS cache expiration time.

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

Establish hosts for the DNS cache.

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

12HTTP Compression

12.2 Overview

The APV appliance supports in-line compression of HTTP objects. By employing this licensed feature, administrators may maximize throughput to the desired site while end-users will experience quicker download speeds. This chapter describes the configuration of HTTP Compression capabilities which are part of ArrayOS platform. Configuration of HTTP Compression functionality can be divided into two main parts. The first part is the basic configuration and the second part is dedicated to advanced configuration.

12.3 Understanding HTTP Compression

HTTP compression, otherwise known as content encoding, is a publicly defined way to compress textual contents transferred from Web servers to browsers. HTTP compression uses public domain compression algorithms to compress XHTML, JavaScript, CSS, and other text files at the server.

By default, the following MIME types can be compressed by the APV appliance for all the browsers:

- Text (text/plain)
- HTML (text/HTML)
- XML (text/XML)

The following MIME types are able to be compressed by the APV 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 do not want to compress some textual contents from Web servers to browsers, they are able to configure a url-exclude HTTP compression rule for the client request via the the "**http compression policy urlexclude** <*virtual_service*> <*url*>" 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 (for example, 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.

12.4 HTTP Compression Configuration

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

12.4.2Configuration Guideline

12.4.2.1.1.1 General Settings of HTTP Compression

Operation	Command
Enable HTTP data compression	**http compression {on|off}** *[virtual_service]*
View HTTP data compression status	**show http compression settings**
Set advanced HTTP compression	**http compression policy useragent** <*user_agent_string*> <*mime_type*> **http compression advanced useragent on**
Set the url-exclude compression rule	**http compression policy urlexclude** <*virtual_service*> <*url*>

12.4.3Configuration Example via CLI

1. Enable HTTP Compression

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

This command enables the HTTP Compression functionality with default settings. By default, the APV appliance compresses the following MIME types for all the browsers:

- Text (text/plain)
- HTML (text/HTML)
- XML (text/XML)

Check the status of HTTP Compression

```
AN(config)#show http compression settings
```

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:

```
AN(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 APV 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**".

Array provides a tested list of browsers that can handle the compressed form of additional MIME types. The APV 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:

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

It activates the compression of Java Script and CSS types for IE 6, IE 7, IE 8 and Mozilla 5.0 browsers.

Configure url-exclude HTTP Compression rule.

```
AN(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.

```
AN(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.

```
AN(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.

AN(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.

13HTTP/HTTPS Routing

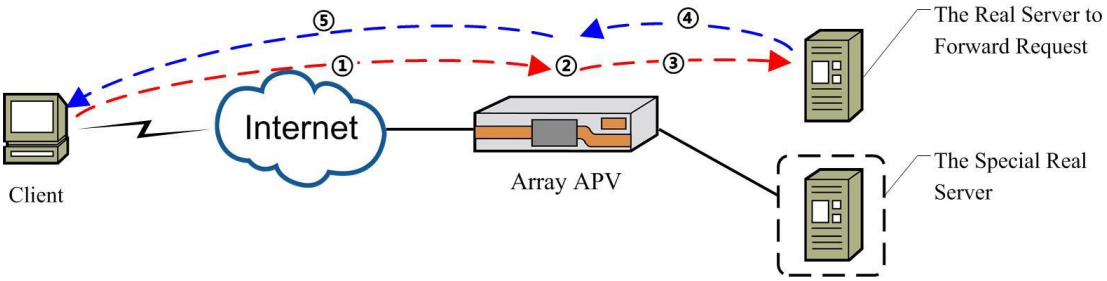
13.2 Overview

This chapter introduces the HTTP/HTTPS routing function of the APV appliance.

In the HTTP/HTTPS routing scenario, the APV appliance forwards the HTTP/HTTPS request of the client (front-end application server) to a specified real server in a forward proxy manner. To make the APV appliance a forward proxy server for forwarding HTTP/HTTPS requests, users need to manually configure the APV virtual service IP and port to be accessed in the front-end application server. When an HTTP/HTTPS request hits the virtual service, the APV appliance selects the real server based on the destination IP address and port carried in the HTTP/HTTPS request header (if the real server is of HTTPS-type, an SSL connection will be established) and forwards the request.

The configuration method of APV's HTTP/HTTPS routing function has been optimized. The administrator only needs to associate the virtual service with a special real server (whose IP and port values are both 0), and the appliance can forward the HTTP/HTTPS request that hits the virtual service according to the destination IP address carried in the request header. In addition to this special real server, the administrator does not need to configure other real servers, which can greatly reduce the administrator's configuration workload in response to the elastic changes in client access.

The APV HTTP/HTTPS routing process is shown in the figure below.



13.2.2.1.1.1.1 APV HTTP/HTTPS Routing Process

➢ **Precondition**

- The client (front-end application server) has been configured with the IP address and port of the virtual service to be accessed.
- The virtual service on the APV appliance has been associated with a special real server whose IP address and port values are 0.

➢ **Routing Process**

1. The client's HTTP/HTTPS request hits the virtual service deployed on APV appliance.

The APV appliance obtains the IP address and port of the destination real server through the host field in the HTTP/HTTPS header.

The APV appliance establishes an SSL connection with the destination real server (when the real server is of HTTPS type) and forwards the request.

After receiving the request, the server sends the response to the APV appliance.

The APV appliance sends the response to the client.

13.3 HTTP/HTTPS Routing Configuration

13.3.2Configuration Guidelines

The following table lists the CLI commands required for HTTP/HTTPS routing function. For the description of related commands, please refer to the ArrayOS APV CLI Handbook.

13.3.2.1.1.1 APV HTTP/HTTPS Routing Configuration Command

Operation	Command
Configure virtual service	**slb virtual http** <*virtual_service*> <*vip*> *[vport] [arp_support] [max_connection]* **slb virtual https** <*virtual_service*> <*vip*> *[vport] [arp_support] [max_connection]*
Configure real service	**slb real http** <*real_service*> <*ip|domain*> *[port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real https** <*real_service*> <*ip|domain*> *[port] [max_connection] [hc_type] [hc_up] [hc_down]*
Configure policy	**slb policy static** <*virtual_service*> <*real_service*>

13.3.3Configuration Example via CLI

In the following configuration example, we first define a special real server whose IP address and port values are 0, and then associate this real server to a virtual service using static policy.

After completing the configuration, there is no need to configure other real servers, and the APV appliance can forward the HTTP/HTTPS request according to the destination IP and port carried in the request header.

```
AN(config)#slb real http "rs_httproute" 0 0 1000 icmp 3 3
AN(config)#slb virtual http "vs_httproute " 10.1.61.100 53
AN(config)#slb policy static "vs_httproute" "rs_httproute"
```

14Secure Sockets Layer (SSL)

14.2 Overview

Now that the basic SLB and Caching are setup on the APV appliance, we can set up the SSL (Secure Sockets Layer) acceleration functionality to provide secure transactions with your clients. The SSL Accelerator works by decrypting the secure traffic and passing the unencrypted traffic to the original server. In an alternative mode the SSL accelerator can be used to decrypt the secure traffic, apply traffic management (SLB, caching, etc.) processing on decrypted traffic and then encrypt it back before passing it to SSL enabled origin server.

ArrayOS APV 10.4 versions provide complete support of SSL protocol versions, including SSLv3.0, TLSv1.0, TLSv1.1, TLSv1.2 and TLSv1.3. Adminsitrators can selectively configure the protocols versions based on security requirements of different applications.

14.3 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.

14.3.2Cryptography

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 cryptographic algorithms, symmetric and asymmetric algorithms.

- In symmetric algorithms, encryption and decryption employ the same secret key, which is negotiated by the communicating parties. The secret key may be intercepted by the third party during its delivery. Therefore, in actual applications, the secret key is always encypted by an asymmetric algorithm and then delivered. Commonly used symmetric algorithms include DES and AES.
- Asymmetric algorithm is also known as public key algorithm. It employs a pair of keys, which are the public key and the private key. The public key is open, while the private key is privately owned. Data encrypted by the public key can only be decrypted by the private key. Asymmetric algorithms supported by the APV appliance include RSA and ECC algorithms.

➢ **RSA algorithm**

RSA is the most commonly used asymmetric cryptography. Its security increases with the key length’s increase. Gradual increases in the RSA key length has slowed down the algorithm’s computation speed.

➢ **ECC algorithm**

Compared with the RSA algorithm, the ECC algorithm provides the same level of confidentiality with shorter key lengths.

14.3.3Digital Signatures

Digital signature is calculated using irreversible signature algorithms. Validating a digital signature helps confirm the identity of the information sender or signer and whether the information was distorted during transmission.

In SSL handshake process, SSL virtual host and real host support the negotiation of RSA, RSASSA-PSS and ECDSA signature algorithms and the validation of RSA, RSASSA-PSS and ECC digital certificates. In Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) key exchange section, the server will send the ephemeral public key via the Server Key Exchange message which carries a digital signature. The client will validate the digital signature to confirm the reliability and integrity of the public key.

In client certificate authentication section, the client will send a digital signature to the server via the Certificate Verify message. This digital signature is to enable the server to confirm the identity of client. The server will use the public key in the client certificate to validate the digital signature.

The following table lists the signature algorithms supported by SSL virtual hosts and real hosts.

14.3.3.1.1.1 Signature Algorithm

Type	SSL Virtual Host	SSL Real Hosts
RSA Signature Algorithm	SHA1RSA SHA224RSA SHA256RSA SHA384RSA SHA512RSA	SHA1RSA SHA224RSA SHA256RSA SHA384RSA SHA512RSA MD5RSA
RSASSA-PSS Signature Algorithms	rsa_pss_rsae_sha256 rsa_pss_rsae_sha384 rsa_pss_rsae_sha512 rsa_pss_pss_sha256 rsa_pss_pss_sha384 rsa_pss_pss_sha512	rsa_pss_rsae_sha256 rsa_pss_rsae_sha384 rsa_pss_rsae_sha512 rsa_pss_pss_sha256 rsa_pss_pss_sha384 rsa_pss_pss_sha512
ECDSA Signature Algorithm	SHA1ECDSA SHA224ECDSA SHA256ECDSA SHA384ECDSA SHA512ECDSA ecdsa_secp256r1_sha256 ecdsa_secp384r1_sha384 ecdsa_secp521r1_sha512	SHA1ECDSA SHA224ECDSA SHA256ECDSA SHA384ECDSA SHA512ECDSA ecdsa_secp256r1_sha256 ecdsa_secp384r1_sha384 ecdsa_secp521r1_sha512

14.3.4Digital Certificates

Certificates contain information identifying the user/device. They are digital documents that will attest to the binding of a public key to an individual or other entity. They allow verification of the claim that a specific public key does, in fact, belong to the specified entity. Certificates help prevent someone from impersonating the server with a false key. SSL uses X.509 standard certificates to validate identities. X.509 standard certificates contain information about the entity, including public key and name. A certificate authority then validates this certificate.

14.3.4.1.1.1 X.509 Certificate

Name	Meaning
Version	Indicates the certificate version number. Certificate format varies by versions.
Serial Number	Indicates the certificate serial number, which is unique among all certificates issued by an authority. If a certificate is revoked, its serial number will be added to the Certificate Revocation List (CRL).
Issuer	Indicates the authority that issues the certificate. Generally, a certificate is issued by a Certificate Authority (CA).
Valid from	Indicates the valid start date of the certificate.
Valid to	Indicates the valid expiration date of the certificate.
Subject	Indicates the person or enterprise identified by the certificate.
Public key	Indicates the public key of the certificate holder.
Signature algorithm	Indicates the signature algorithm used to generate the signature.
Thumbprint, Thumprint algorithm	Indicates the thumbprint and thumbprint algorithm. Thumbprint is a hashed value of the entire certificate calculated by the thumbprint algorithm and then encrypted by the private key of the CA.

The APV appliance supports two types of digital certificates: RSA (including RSAPSS) and ECC certificates. Digital certificates are issued by CA. To obtain a certificate from CA, individuals or enterprises need to send Certificate Signing Request (CSR) to CA, and then import and activate the certificate. The APV appliance supports generation of RSA, RSASSA-PSS, and ECC CSRs for a virtual host.

➢ **Generating a CSR**

- To apply for an RSA or RSAPSS certificate, execute the "**ssl csr**" command to generate a PKCSv1.5 RSA CSR or RSASSA-PSS CSR.
- To apply for an ECC certificate, execute the "**ssl ecc csr**" command to generate an ECC CSR.

CA will send back the desired certificate via Email. After receiving the certificate, administrators need to import it by using CLI commands.

➢ **Importing a certificate**

RSA, RSAPSS and ECC certificates are imported using the "**ssl import certificate**" command. RSAPSS certificates are used only for SSL hosts that support TLSv1.3. If an RSAPSS certificate is imported for an SSL host that does not support TLSv1.3, it cannot be activated and will not be used in SSL handshakes.

➢ **Activating a certificate**

All RSA, RSAPSS, ECC certificates are activated by using the "**ssl activate certificate** *<host_name> [certificate_index] [domain_name] [certificate_type]*" command. It allows you to activate all types of certificates at a time or only the specific type of certificate. Note that the "certificate_type" for both RSA and RSAPSS certificates is "rsa".

The number of certificates that can be imported and activated for an SSL host is as follows:

- If it is associated with one or more domain names, you can import three RSA certificates (including RSAPSS certificates) and three ECC certificates for each domain of the virtual host. Each domain of it can have one active RSA or RSAPSS certificate and one active ECC certificate.
- If it is not associated with any domain name, you can import a maximum of three RSA certificates (including RSAPSS certificates), three ECC certificates for it. It can have one active RSA or RSAPSS certificate, one active ECC certificate.

Note: The APV appliance running FIPS HSM does not support ECC algorithms. In FIPS HSM environment, each virtual host can have one active RSA certificate for each of its associated domain. Each real host can have only one RSA certificate.

The following is an example for importing and activate an RSA certificate and an ECC certificate with index 2 for real host "rhost".

1. Import the RSA certificate.

```
AN(config)#ssl import certificate rhost 2
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIIDPDCCAqWgAwIBAgIFAK+57hMwDQYJKoZIhvcNAQELBQAwgaoxCzAJBgNVBAYTA
lVTMQswCQYDVQQIEwJDQTERMA8GA1UEBxMITWlscGl0YXMxGzAZBgNVBAoTEkFy
cmF5TmV0d29ya3MgSW5jLjEUMBIGA1UECxMLQVBWIFByb2R1Y3QxHjAcBgNVBAMTF
Xd3dy5hcnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5
ldHdvcmtzLm5ldDAeFw0xNjA3MTEwOTMwMzhaFw0yNDA5MjcwOTMwMzhaMIGUMQsw
CQYDVQQGEwJVUzEQMA4GA1UECAwHRmxvcmlkYTEQMA4GA1UEBwwHRmxvcmlkY
TEOMAwGA1UECgwFQXJyYXkxCzAJBgNVBAsMAlBPMQswCQYDVQQLDAJQTzELMA
```

```
kGA1UECwwCUE8xDjAMBgNVBAMMBXZob3N0MRowGAYJKoZIhvcNAQkBFgthYmNA
MTI2LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMlV4+6s/0zYho
akPUNNwGdh1HCtOyKWpXl7vw6YLxQlKIvlEel69HsfIL1hKV+YmZOuY1istLPqBdxPqLniy
VIcttX9QpPJJYiRnaU/OjylWLKQt61yLGPrsTzrDKnv2KE9RTpe8MWEwgzy42pmW3Wptl6Y3
Xoox1k521gYVoDSfU+mqzwEFftclx0CD35v8TFbS0AP6js3iikZIddcZiMmtm87oSgQ0Q9Tn8jv
jNYXpLJ/BcEnGHFsAP1S5MFC8Wj7jNJ5wHNSPPpdePDnQ49j+E7Ah9XFY502svxbgUjEt1z
wu2CAVt4jvKTEU3tbWE+tGIjJpFtkFa4f3urUXvkCAwEAATANBgkqhkiG9w0BAQsFAAOBg
QB/CD9eCsXPllVqmCasL0XpeOF/pAgon6pgC4BVf1quZEb7C78IATQWrQJfHdYSZVQwXws
bg6YLHnpI+SdpoNwjsrd25K6AvCmBMANtqFighIwjcbNJqINGWWxZUIquHVmyD+L/53x2Ni
h4qJ5zjMfWzPbR7KbwF0cDJs1/YfMEwg==
-----END CERTIFICATE-----
...
PEM format.
Certificate import successful !!!
```

Import the ECC certificate.

```
AN(config)#ssl import certificate rhost 2
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIICIzCCAcqgAwIBAgIFAI88UzkwCgYIKoZIzj0EAwIwgaoxCzAJBgNVBAYTAlVTMQswC
QYDVQQIDAJDQTERMA8GA1UEBwwITWlscGl0YXMxGzAZBgNVBAoMEkFycmF5TmV
0d29ya3MgSW5jLjEUMBIGA1UECwwLQVBWIFByb2R1Y3QxHjAcBgNVBAMMFXd3dy5h
cnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5ldHdvcm
tzLm5ldDAeFw0xNjA3MTEwOTMxMjdaFw0yNDA5MjcwOTMxMjdaMIGHMQswCQYDVQ
QGEwJDTjELMAkGA1UECAwCQ04xCzAJBgNVBAcMAkNOMQswCQYDVQQKDAJDTjE
LMAkGA1UECwwCQ04xCzAJBgNVBAsMAkNOMQswCQYDVQQLDAJDTjEOMAwGA1U
EAwwFdmhvc3QxGjAYBgkqhkiG9w0BCQEWC2FiY0AxNjIuY29tMFkwEwYHKoZIzj0CAQY
IKoZIzj0DAQcDQgAECXCDImdSY1/eq400+rReCE5qLfL9VeIHygJR8lAOzFTG58stV9kBjKR
BTBL5p5tdZqFX1DbxZ0bT7+mCuBvS7jAKBggqhkjOPQQDAgNHADBEAiAfDVKFeeeyEq9
HvOmXEGEueaYDCMoVg1zm2T396BOBVQIgKZUTOqn+Kb0Nh64b9mS0Fr8mtTqps5Fl7Q/
v2YO4MqQ=
-----END CERTIFICATE-----
...
PEM format.
Certificate import successful !!!
```

Activate the imported RSA and ECC certificates.

```
AN(config)#ssl activate certificate rhost 2 "" all
Do you want to activate all Certificates #2? [YES/(NO)]: YES
```

Warning: RSA certificate chain is incomplete for rhost. Please add interca or rootca certificate.
RSA Certificate #2 is activated successfully!

Warning: ECC certificate chain is incomplete for rhost. Please add interca or rootca certificate.
ECC Certificate #2 is activated successfully!

7.2.1.1 Client Certificate Authentication

In client certificate authentication, a client sends its certificate to a server for authentication upon receiving a Client Certificiate Request message from the server. The APV appliance supports using the Array certificate parser (Array patent) to verify the client certificate in a fast way.

- When the APV appliance functions as an SSL proxy client (SSL real host), if client certificate authentication is enabled for the SSL real host, it will send its certificate to the real service. In this scenario, if the SSL real host has both RSA and ECC certificates activated, it will select the certificate to be sent according to the following principles:
  - If RSA certificate type is specified in the Client Certificate Request message, the RSA certificate will be sent to the real service.
  - If only ECC certificate type is specified in the Client Certificate Request message, and the negotiated cipher suite is of the “ECDHE…” type, the ECC certificate wil be sent to the real service.
  - If only ECC certificate type is specified in the Client Certificate Request message, but the negotiated cipher suite is not an “ECDHE…” one, the SSL real host will reset the connection.
- When the APV appliance functions as an SSL proxy server (SSL virtual host), if client certificate authentication is enabled for the SSL virtual host, it will request SSL client to provide a certificate for authentication.

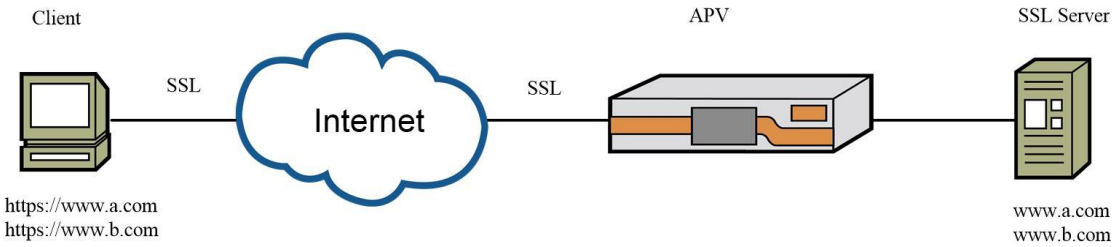
14.3.5Server Name Indication (SNI)

SNI provides the SSL authentication for each website running on an SSL server by using the domain name of the website. The SNI function provide the SSL authentication for multiple websites on an SSL server by associating multiple domain names with a virtual host instead of assigning an IP address for each website and associating a unique virtual service, which simplifies virtual service configurations and lessens IP address exhaustion.

Configuration Example

➢ **Prerequisites:**

- Two websites www.a.com and www.b.com run on an SSL server.
- A private key and the corresponding certificate have been imported for the specified SSL virtual host, and the certificate has been activated.



14.3.5.1.1.1.1 SNI Architecture

- **CLI**

1. Configure SLB.

Add the real service “rhost1” and “rhost2”. Associate the real service “rhost1” and “rhost2”with group “vhg1” and “vhg2” respectively. Add the virtual service “vshost1”. To select the hit real service based on the domain, configure QoS hostname policy for the group “vhg1” and “vhg2”.

```
AN(config)#slb real http "rhost1" 172.16.78.22 80 65535 tcp 3 3
AN(config)#slb real http "rhost2" 172.16.78.23 80 65535 tcp 3 3
AN(config)#slb group method "vhg1" rr
AN(config)#slb group method "vhg2" rr
AN(config)#slb group member "vhg1" "rhost1" 1 0
AN(config)#slb group member "vhg2" "rhost2" 1 0
AN(config)#slb virtual https "vshost1" 192.168.12.62 443 arp 0
AN(config)#slb policy qos hostname "spa" "vshost1" "vhg1" "www.a.com" 100
AN(config)#slb policy qos hostname "spb" "vshost1" "vhg2" "www.b.com" 200
AN(config)#slb policy default "vshost1" "vhg1"
```

Configure an SSL virtual host and associate with the virtual service “vshost1”.

```
AN(config)#ssl host virtual vshost vshost1
```

Associate the domain “www.a.com” and “www.b.com” with the SSL virtual host.

```
AN(config)#ssl sni vshost www.a.com
AN(config)#ssl sni vshost www.b.com
```

Import a private key for the domain “www.a.com”.

For how to get the private key and certificate and other ways to import the private key and certificate, please refer to section 14.4.3 Configuration Example via CLI.

```
AN(config)#ssl import key vshost1 www.a.com
You may overwrite an existing key file. This may require you
to purchase a new certificate. Type YES to continue: YES
Enter key, use "..." on a single line, without quotes
to terminate import
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,7B26776D2C14EB1F
```

```
WwU2hyREMjGi+6k5nq5gecnTSJSTk8q3ori+1jooOfgvRV72x/e3hGRFIysKb15J
1xeyUHFjtNJIHlEVjNg2XURXIRDO+qatI0cnvW1sYbuQodat0jGzpP/oOxGVcRsg
kTuP8xObpUou0RevvZVbfYHWsdWZnY5qhLjje1UGKUu8HwaMp3Q7OJ7pper7TOAX
CkiVn1zHgeFkB4p2lGwCHxbdNJg7ee5U2HFiSoC/8P8G64kIkEfxuMFwzkaVlia+
Anf40WlBcvixZaNZqHkZdcfJGlqKnqnpcJWiyLd8oPr3npYu6+c1PyHiyEua67nE
zK5G4KVFacW0POPLHfj/Q1yMqYcWySyecJvtq0jxYLujGH6qckLhMMltdptCfFvZ
6gfncRanNWEU+2v+nc509XKv9zJeR58WsAh81AN7aChuEBj69cjL4HwOtk3nttVm
pr5+cluF6mpbk2rJuZ14l+Hwc63XRAaHXV0bIQfo+rQBIWHkItvN4S2a0G23IC9N
32Y1qTqQgoj9qxSXvw1oKmV+UcVjtt78GyT3m7pYPtNc/wdehUPrR4Lc8hSiV3m2
9ZGIoNpdhVgCGw/MM65tr3dRZfYcG8cPJ4pk/3WKq9OfznlugAxo4KWebea51CaT
4YBMO5nETHT9WesX9viiKtbhPaAIKQOd4Rwcs5FK0ETHSfBBpW0tsXw562s6QnNS
mnWuoksvqqXqCbnuNSz611z3dOhiLwHdqCqLEDNEhv231ielOk1qoUo+ad0eiM0E
TrP6Zyw+j6i9K71sSe28Zm4gVQMnSHxGyp0K+zK3cYs=
-----END RSA PRIVATE KEY-----
...
PEM format.
Enter passphrase for the private key:

Key import successful !!!
```

Import the corresponding certificate for the domain “www.a.com”.

```
AN(config)#ssl import certificate vshost 1 www.a.com
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIIDPDCCAqWgAwIBAgIFAK+57hMwDQYJKoZIhvcNAQELBQAwgaoxCzAJBgNVBAYTA
lVTMQswCQYDVQQIEwJDQTERMA8GA1UEBxMITWlscGl0YXMxGzAZBgNVBAoTEkFy
cmF5TmV0d29ya3MgSW5jLjEUMBIGA1UECxMLQVBWIFByb2R1Y3QxHjAcBgNVBAMTF
Xd3dy5hcnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5
ldHdvcmtzLm5ldDAeFw0xNjA3MTEwOTMwMzhaFw0yNDA5MjcwOTMwMzhaMIGUMQsw
CQYDVQQGEwJVUzEQMA4GA1UECAwHRmxvcmlkYTEQMA4GA1UEBwwHRmxvcmlkY
TEOMAwGA1UECgwFQXJyYXkxCzAJBgNVBAsMAlBPMQswCQYDVQQLDAJQTzELMA
kGA1UECwwCUE8xDjAMBgNVBAMMBXZob3N0MRowGAYJKoZIhvcNAQkBFgthYmNA
MTI2LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMlV4+6s/0zY
hoakPUNNwGdh1HCtOyKWpXl7vw6YLxQlKIvlEel69HsfIL1hKV+YmZOuY1istLPqBdxPqLn
iyVIcttX9QpPJJYiRnaU/OjylWLKQt61yLGPrsTzrDKnv2KE9RTpe8MWEwgzy42pmW3Wptl6
Y3Xoox1k521gYVoDSfU+mqzwEFftclx0CD35v8TFbS0AP6js3iikZIddcZiMmtm87oSgQ0Q9Tn
8jvjNYXpLJ/BcEnGHFsAP1S5MFC8Wj7jNJ5wHNSPPpdePDnQ49j+E7Ah9XFY502svxbgUjEt
1zwu2CAVt4jvKTEU3tbWE+tGIjJpFtkFa4f3urUXvkCAwEAATANBgkqhkiG9w0BAQsFAAOB
gQB/CD9eCsXPllVqmCasL0XpeOF/pAgon6pgC4BVf1quZEb7C78IATQWrQJfHdYSZVQwXw
```

```
sbg6YLHnpI+SdpoNwjsrd25K6AvCmBMANtqFighIwjcbNJqINGWWxZUIquHVmyD+L/53x2
Nih4qJ5zjMfWzPbR7KbwF0cDJs1/YfMEwg==
-----END CERTIFICATE-----
...
PEM format.
Certificate import successful !!!
```

Import a private key for the domain “www.b.com”.

```
AN(config)#ssl import key vshost1 www.b.com
You may overwrite an existing key file. This may require you
to purchase a new certificate. Type YES to continue: YES
Enter key, use "..." on a single line, without quotes
to terminate import
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,7B26776D2C14EB1F

WwU2hyREMjGi+6k5nq5gecnTSJSTk8q3ori+1jooOfgvRV72x/e3hGRFIysKb15J
1xeyUHFjtNJIHlEVjNg2XURXIRDO+qatI0cnvW1sYbuQodat0jGzpP/oOxGVcRsg
kTuP8xObpUou0RevvZVbfYHWsdWZnY5qhLjje1UGKUu8HwaMp3Q7OJ7pper7TOAX
CkiVn1zHgeFkB4p2lGwCHxbdNJg7ee5U2HFiSoC/8P8G64kIkEfxuMFwzkaVlia+
Anf40WlBcvixZaNZqHkZdcfJGlqKnqnpcJWiyLd8oPr3npYu6+c1PyHiyEua67nE
zK5G4KVFacW0POPLHfj/Q1yMqYcWySyecJvtq0jxYLujGH6qckLhMMltdptCfFvZ
6gfncRanNWEU+2v+nc509XKv9zJeR58WsAh81AN7aChuEBj69cjL4HwOtk3nttVm
pr5+cluF6mpbk2rJuZ14l+Hwc63XRAaHXV0bIQfo+rQBIWHkItvN4S2a0G23IC9N
32Y1qTqQgoj9qxSXvw1oKmV+UcVjtt78GyT3m7pYPtNc/wdehUPrR4Lc8hSiV3m2
9ZGIoNpdhVgCGw/MM65tr3dRZfYcG8cPJ4pk/3WKq9OfznlugAxo4KWebea51CaT
4YBMO5nETHT9WesX9viiKtbhPaAIKQOd4Rwcs5FK0ETHSfBBpW0tsXw562s6QnNS
mnWuoksvqqXqCbnuNSz611z3dOhiLwHdqCqLEDNEhv231ielOk1qoUo+ad0eiM0E
TrP6Zyw+j6i9K71sSe28Zm4gVQMnSHxGyp0K+zK3cYs=
-----END RSA PRIVATE KEY-----
...
PEM format.
Enter passphrase for the private key:

Key import successful !!!
```

Import the corresponding certificate for the domain “www.b.com”.

```
AN(config)#ssl import certificate vshost 1 www.b.com
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
```

```
-----BEGIN CERTIFICATE-----
MIICIzCCAcqgAwIBAgIFAI88UzkwCgYIKoZIzj0EAwIwgaoxCzAJBgNVBAYTAlVTMQswC
QYDVQQIDAJDQTERMA8GA1UEBwwITWlscGl0YXMxGzAZBgNVBAoMEkFycmF5TmV
0d29ya3MgSW5jLjEUMBIGA1UECwwLQVBWIFByb2R1Y3QxHjAcBgNVBAMMFXd3dy5h
cnJheW5ldHdvcmtzLm5ldDEoMCYGCSqGSIb3DQEJARYZc3VwcG9ydEBhcnJheW5ldHdvcm
tzLm5ldDAeFw0xNjA3MTEwOTMxMjdaFw0yNDA5MjcwOTMxMjdaMIGHMQswCQYDVQ
QGEwJDTjELMAkGA1UECAwCQ04xCzAJBgNVBAcMAkNOMQswCQYDVQQKDAJDTjE
LMAkGA1UECwwCQ04xCzAJBgNVBAsMAkNOMQswCQYDVQQLDAJDTjEOMAwGA1U
EAwwFdmhvc3QxGjAYBgkqhkiG9w0BCQEWC2FiY0AxNjIuY29tMFkwEwYHKoZIzj0CAQY
IKoZIzj0DAQcDQgAECXCDImdSY1/eq400+rReCE5qLfL9VeIHygJR8lAOzFTG58stV9kBjKR
BTBL5p5tdZqFX1DbxZ0bT7+mCuBvS7jAKBggqhkjOPQQDAgNHADBEAiAfDVKFeeeyEq9
HvOmXEGEueaYDCMoVg1zm2T396BOBVQIgKZUTOqn+Kb0Nh64b9mS0Fr8mtTqps5Fl7Q/
v2YO4MqQ=
-----END CERTIFICATE-----
...
PEM format.
Certificate import successful !!!
```

Activate the certificate for the domain “www.a.com” and “www.b.com”.

```
AN(config)#ssl activate certificate vshost 1 www.a.com
AN(config)#ssl activate certificate vshost 1 www.b.com
```

Enable the SSL virtual host “vshost”.

```
AN(config)#ssl start vshost
```

14.3.6HTTP/2 Support

When an SSL host is configured in compliance with specific requirements, SSL acceleration supports HTTP/2 running over TLS.

For an SSL virtual host, it must meet the following configurations to support running HTTP/2; otherwise, HTTP/1.1 will be used:

1. The SLB virtual service associated with the virtual host is enabled with HTTP/2.

The virtual host supports TLSv1.2.

The virtual host supports at least one of these cipher suites:
ECDHE-RSA-AES128-GCM-SHA256, ECDHE-RSA-AES256-GCM-SHA384, ECDHE-ECDSA-AES128-GCM-SHA256 and ECDHE-ECDSA-AES256-GCM-SHA384.

For an SSL real host, it must meet the following configurations to support running HTTP/2; otherwise, HTTP/1.1 will be used:

1. The SLB real service associated with the real host supports and is enabled with HTTP/2.

The virtual host supports running HTTP/2 (see the three conditions for HTTP/2 support by virtual host stated previously).

The real host supports TLSv1.2.

The real host supports at least one of these cipher suites: ECDHE-RSA-AES128-GCM-SHA256, ECDHE-RSA-AES256-GCM-SHA384, ECDHE-ECDSA-AES128-GCM-SHA256 and ECDHE-ECDSA-AES256-GCM-SHA384.

Note: In the deployment scenario where HTTP/2 runs over TLS, SSL renegotiation must be disabled. Use the "**ssl globals renegotiation off**" command to disable it globally, and the "**ssl settings reneg** <*virtual_host_name*>" command to disable it on individual virtual host.

14.4 SSL Acceleration Configuration

14.4.2Configuration Guidelines

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

Virtual Host: An SSL host associated with an SLB virtual. An SSL virtual host acts as an SSL server and is used to communicate by using SSL between browser and APV appliance.

Real Host: An SSL host associated with an SLB real. An SSL real host acts as an SSL client and is used to communicate by using SSL between APV appliance and backend origin server.

Origin Server: A backend server that will accept clear-text or encrypted requests.

Clear-text: Any traffic that is not encrypted.

Virtual Host Port: The port that SSL virtual host will listen on. Typically port 443 is used.

Key (private): A private key that is stored on the APV appliance for PKI (Public Key Infrastructure) authentication purposes. APV 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 APV appliance also maintains a list of Trusted Certificate Authorities to verify certificates.

For our example environment, we have a domain name of "www.example.com". For our SSL purposes we will be using "www.example.com" as our SSL virtual host. This SSL virtual host is associated with an SLB virtual host using IP 10.10.0.10 and port 443.

SSL virtual Host: www.example.com

SLB virtual Host IP: 10.10.0.10

SLB virtual Host Port: 443

There are two methods for setting up SSL acceleration. The first method applies if you have never set up SSL, and you will need to walk through the whole process of setting up the SSL virtual host and generation of a CSR to send to the CA of your choice. The CA will send you a signed certificate that you will then import. The second method applies if you already have a key and certificate, and you can skip the CSR step and import your key and certificate. Let's go ahead and setup SSL as though we have never set it up.

14.4.2.1.1.1 General Settings of SSL

Operation	Command
Create an SSL virtual host	**slb virtual https** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **ssl host {real|virtual}** *<host_name> [slb_service]*
Import certificate and key for SSL virtual host	**ssl csr** *<virtual_host_name> [key_length] [certificate_index] [signature_algorithm_index] [domain_name]* **ssl ecc csr** *<virtual_host_name> [curve_name] [certificate_index] [signature_algorithm_index] [domain_name]* **ssl import certificate** *<host_name> [certificate_index] [domain_name] [tftp_ip] [file_name]* **ssl import key** *<host_name> [key_index] [domain_name] [tftp_ip] [file_name]* **ssl activate certificate** *<host_name> [certificate_index] [domain_name] [certificate_type]*
Create an SSL real host	**slb real https** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb real tcps** *<real_service> <ip> <port> [max_connection] [hc_type] [hc_up] [hc_down]* **ssl host {real|virtual}** *<host_name> [slb_service]*
Advanced configuration for an SSL virtual host	**ssl sto***p <host_name>* **ssl settings ciphersuite** *<host_name> <cipher_string>* **ssl settings protocol** *<host_name> <version>* **ssl settings reuse** *< host_name>* **ssl settings clientauth** *<host_name>* **ssl settings certfilter** *<virtual_host_name> <condition_1> [condition_2]* **ssl import rootca** *[host_name] [domain_name] [tftp_ip] [file_name]* **ssl settings crl offline** *<host_name> <cdp_name> <cdp_url> [domain_name] [time_interval] [delay_time]* **ssl settings crl online** *<virtual_host_name>* **ssl settings ocsp** *<virtual_host_name> <ocsp_server_url>* **ssl settings minimum** *<virtual_host_name> <cipher_strength> <redirect_url>* **ssl start** *<host_name>* **ssl import error** *<error_code> <url> [virtual_host_name]* **ssl load error** *<error_code> [virtual_host_name]*

Operation	Command
Advanced configuration for an SSL real host	**ssl settings ciphersuite** <*host_name*> <*cipher_string*> **ssl settings protocol** <*host_name*> <*version*> **ssl settings reuse** <*host_name*> **ssl settings clientauth** <*host_name*> **ssl import rootca** *[host_name] [domain_name] [tftp_ip] [file_name]* **ssl settings servername** <*real_host_name*> <*common_name*>

14.4.3Configuration Example via CLI

- **Creating an SSL Virtual Host**

Firstly, execute the "**slb virtual…**" command to create an SLB TCPS or HTTPS virtual service. Secondly, run the "**ssl host**" command to create an SSL virtual host and associate with the virtual service.

```
AN(config)#slb virtual https virtual1https 10.10.0.10 443
AN(config)#ssl host virtual www.example.com virtual1https
```

In this example, "vitrual1https" is the created SLB virtual service, and "www.example.com" is the created SSL virtual host.

- **Importing Certificate and Key for the Newly Created SSL Virtual Host**

The following configuration illustrates an example for applying for, importing and activating an RSA key and certificate for virtual host "www.example.com".

1. Run the "**ssl csr**" command to generate an RSA CSR for "www.example.com". (To apply for an ECC certificate, run the "**ssl ecc csr**" command.

```
AN(config)#ssl csr www.example.com
Generating keys for "www.example.com"....please wait
We will now gather some required information about your SSL virtual host.
This information will be encoded into your certificate.
TWO-character country code for your organization (eg. US) :US
State or province []:CA
location or local city []:San Jose
Organization Name :Example.com
Organizational Unit :Example.com
Organizational Unit []:
Organizational Unit []:
Do you want to use the virtual host name "www.example.com"
as the Common Name? (recommended) [Y/N]: Y
email address of administrator []:admin@example.com
Do you want to add Subject Alternative Names? (recommended) [Y/N] N
-----BEGIN CERTIFICATE REQUEST-----
MIIC6jCCAdICAQAwgZUxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTERMA8GA1U
EBwwIU2FuIEpvc2UxFDASBgNVBAoMC0V4YW1wbGUuY29tMRQwEgYDVQQLDAtFeGF
```

```
tcGxlLmNvbTEYMBYGA1UEAwwPd3d3LmV4YW1wbGUuY29tMSAwHgYJKoZIhvcNAQkB
FhFhZG1pbkBleGFtcGxlLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggE
BAKfPzHgGQA2DKh7kkzSKczUO9RkMRvrMX+MssKiVwGUpwUZ3B0YlW5gqUQ0ieYqNQ
bYr4s7G+d+zHe7NtsyUADMJwgDKK4pQaiBuxVWzQtqQqIGEZc4NqIaltJzOpzZSMqY0SvbQ
3MJqrvVgZpdObeedW5SRVjn8zyebr2j/re4tTIJpm+lj9FiFf/yVHsJdXjJrOONYfsAcaI9c8a5BLqe
PavZEvzh3p+eiCwjGflv8n48O8ub+PIGU1W230gkfRdG5D6e1yWlXFdceveunuOlfuoL2yBgHu
yxxgu+RkQd6ZqV6eACLbv47OTMr9MUGuW6TuQ0TI+T24lY8DBn/TxcCAwEAAaAPMA0G
CSqGSIb3DQEJDjEAMA0GCSqGSIb3DQEBCwUAA4IBAQCRl4Mao7hBqsqH/+kU8IQK7aq
wdujSDj5KxO5rKkSutslaqfsIbpr85nGKFqxrBxpy0lFs6NegztSV0dCc/Dt3iVaAqLEgeVmdFA9Z
bcpwHecQmeg1D200GmpsU3T2xiqM0mDc7jmRywWenCJkRWmO3EWeO9N5mbbeoOUs4Kel
KvVayMe2k9YvArSmOa3NHzyTQ1Zhqc80Q6Jg7mSw6B9et0JpKIim+3Hw12ULOdhIDijLOa8
GiDdhuL4J5FBDW0wY8Jl+YKeW7r8GDldENP1bdvWDdDkI0zHhVuwPDOuAcwWj23gT7jLo
wcNYRNIVW5RGrXHjlfb9UXKqMboJmpp+
-----END CERTIFICATE REQUEST-----

Do you want the private key to be exportable [Yes/(No)]:Yes
Enter passphrase for the private key:

Confirm passphrase for the private key:

Warning: RSA certificate chain is incomplete for www.example.com. Please add interca or rootca
certificate.
```

Besides the RSA CSR, this command also generates an RSA key pair and a testing certificate for virtual host “www.example.com”. The testing certificate is used for testing purpose only. To use it for testing or demonstration, you can directly start the SSL virtual host.

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

Now, you can connect to the website securely via a web browser.

Forward CSR to a CA.

In the output of the “ssl csr” command, copy the line starting from “-----BEGIN CERTIFICATE REQUEST----” to “----END CERTIFICATE REQUEST-----” and send it to CA. CA will reply with a digital certificate in the Email. The following is an example certificate.

```
-----BEGIN CERTIFICATE-----
MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0
NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQ
QDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2Z
WxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgw
MTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEw
NET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIu
MC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQ
ADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIX
```

```
gRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6Yin
DTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEB
QADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4
pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnE
aiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cRQLU=
-----END CERTIFICATE-----
```

Note: It is imperative that you do not delete the SSL virtual host before importing the certificate received from CA. Otherwise, you will have to send another CSR to your CA to re-apply for a certificate. Fortunately, most CAs provide a 30-day trial period for getting another certificate in case that something goes wrong. If the trial period elapses, you might have to pay for another certificate.

Run the "**ssl import certificate**" command to import the received certificate for the SSL virtual host.

```
AN(config)#ssl import certificate www.example.com 1
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
Enter the certificate file in PEM format,
  use "..." on a single line, without quotes
  to terminate import
-----BEGIN CERTIFICATE-----
MIICnjCANgcANgEUMA0GCSqGSIb3DQEBBAUAMIG5MQswCQYDVQQGEwJVUzETMB
EGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxHDAaBgNVBAoTE0
NsaWNrQXJyYXkgTmV0d29ya3MxFDASBgNVBAsTC0RldmVsb3BtZW50MSMwIQYDVQ
QDExpkZXZlbG9wbWVudC5jbGlja2FycmF5LmNvbTEpMCcGCSqGSIb3DQEJARYaZGV2Z
WxvcG1lbnRAY2xpY2thcnJheS5jb20wHhcNMDIwMjEzMTgwMTI5WhcNMDMwMjA4MTgw
MTI5WjB0MQswCQYDVQQGEwJVUzEMMAoGA1UECBMDRE9EMQwwCgYDVQQHEw
NET08xCzAJBgNVBAoTAkRPMQswCQYDVQQLEwJETzETMBEGA1UEAxMKMTAuMTIu
MC4xNDEaMBgGCSqGSIb3DQEJARYLbWhAZGtkay5jb20wgZ8wDQYJKoZIhvcNAQEBBQ
ADgY0AMIGJAoGBAMx4r+ae4kTZggtyU047OsKUyqCt+V1MHgTPTpVxdtxYhSTSOZwYIX
gRqBEdJvs2/ua1XZRzLOCTa58VI/8I3derAPqz79WpBRsxD25rCT1rzmalfkTea3V8jHJYP6Yin
DTWKFKztxeUclkzukzPUZO6M0fI5ToXNuLEe+IwvOkfAgMBAAEwDQYJKoZIhvcNAQEEB
QADgYEAodV5O0LKUr/O0BbxOnwmyP/DkLj4bpe9XxQO6B4psDey/+xBHs6tgGKuy8spbcJ4
pQc+5KLydK1ZYcTkbxJq41K4RHM11OClXVjm3xRhqKQnjzNboExIvkZsKIBbfLkBrM1eBnE
aiYWXmsYGfxPkwdhKlQCLQgN+G3IKu2cRQLU=
-----END CERTIFICATE-----
…
```

You can also import the certificate from a remote TFTP server, which needs the TFTP server's IP address and certificate file name ("example.crt" in this case).

```
AN(config)# ssl import certificate www.example.com 1 10.10.13.82 example. crt
You may overwrite an existing certificate.
Type YES to continue, NO to abort: YES
```

Note: Available key and certificate files can be directly imported into an SSL virtual host using the "**ssl import key**" and "**ssl import certificate**" commands (without applying for a certificate using CSR). These commands support copy and paste of the key or certificate content into the command prompt or import of the files from a remote TFTP server. Whichever method is used, please note following principles:

- The key file must be imported first and then the certificate file.
- Keys and certificates in non-PEM format can only be imported using the TFTP method.

Run the "**ssl activate certificate**" command to activate the certificate.

```
AN(config)#ssl activate certificate www.example.com 1
```

Note: The system will check the certificate chain when the certificate is activated. A warning message, stating that the certificate chain is incomplete, will be printed if its root CA certificate or intermediate CA certificate cannot be found in the virtual host's global trusted CA file or intermediate CA file. These certificates can be imported by using the "**ssl import rootca**" and "**ssl import interca**" commands.

Start the SSL virtual host.

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

Till now, SSL acceleration configuration for "www.example.com" is completed. A client can access the virtual host via HTTPS.

➢ **Import Certificate and Key from IIS and NS iPlanet Web Servers**

• **IIS**

If you are using the Microsoft IIS server, the APV appliance will allow you to import the certificate from IIS versions 4 and 5 through TFTP mechanism. IIS stores the SSL key and certificate in the same file. This file is in .PFX format. You need to put this file onto a TFTP server in its TFTP root directory and rename it as <host_name>.crt. This file then can be imported into APV appliance through the "**ssl import certificate**" command. This command takes TFTP server IP as an extra argument.

```
AN(config)#ssl import certificate www.example.com 1 10.10.0.3
```

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

After importing the certificate successfully, you will get a response from the CLI prompt. Then you can activate the certificate via the command "**ssl activate certificate**". For example:

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

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

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

- **Netscape/iPlanet**

If you are using the Netscape or iPlanet servers, the APV 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.

```
AN(config)#ssl import key www.example.com 1 10.10.0.3 www.example.com.key
```

This command imports the key from 10.10.0.3 with the file name "www.example.com.key".

Note: You must first import the certificate and then import the key when importing an SSL cert/key pair from iPlanet.

```
AN(config)#ssl import certificate www.example.com 1 10.10.0.3 www.example.com.crt
```

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

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

After importing the certificate successfully, you will get a response from the CLI prompt. Then you can activate the specific certificate via the command "**ssl activate certificate**". For example:

```
SJ-Box1(config)#ssl activate certificate www.example.com 1
```

Then we can start the SSL subsystem:

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

Now the APV appliance is configured to take full advantage of the SSL functionality within the ArrayOS.

Note: In this section we have created an SLB virtual service and configured SSL for it. This SLB virtual service is now ready to be used, and need to be linked with one or more SLB real services so that the SLB module can complete the SSL requests coming to this SLB virtual. To get information on "How to associate an SLB virtual with an SLB real", please refer to the SLB configuration section.

- **Creating an SSL Real Host**

The APV appliance allows you to use the SSL subsystem to talk to SSL enabled real servers. This allows an encrypted transaction between the ArrayOS and the backend servers.

Configuration of SSL real host is very simple and can be explained as follows:

The first step in this procedure is to define the SLB real service. To do this first we will employ an SLB related command. This command will create the SLB real service. The second step is to use the "**ssl host**" command to define the SSL real host.

For the definition and meaning of each parameter supplied in this command, please refer to the SLB CLI section of CLI Handbook.

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

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

Now the APV appliance is configured to take full advantage of the SSL functionality while communicating with the backend server.

Note: In this section we have created an SLB real service and configured SSL for it. This SLB real service is now ready to be used, and need to be linked with an SLB virtual service so that the SLB module can direct the traffic to this SSL enabled SLB real service. To get information on "How to associate an SLB real with an SLB virtual" please refer to Chapter 7 Server Load Balancing (SLB).

14.4.3.1 Advanced SSL Configuration for SSL Virtual Host

1. Disable SSL virtual host.

```
AN(config)#ssl stop www.example.com
```

SSL virtual host must be disabled before you change its configurations.

Configure cipher suites for the SSL virtual host.

```
AN(config)#ssl settings ciphersuite "www.example.com" "DES-CBC3-SHA"
```

The following table lists the RSA and ECC (ECDHE-ECDSA…) cipher suites allowed for an SSL virtual host with different protocol versions. "Y" indicates that the cipher suite is supported. "N" indicates that the cipher suite is not supported.

Cipher Suites	Bits	SSL Protocol – Virtual Host				
		SSLv3.0	TLSv1	TLSv1.1	TLSv1.2	TLSv1.3
RC4-MD5	128	Y	Y	Y	Y	N
RC4-SHA	128	Y	Y	Y	Y	N
DES-CBC-SHA	64	Y	Y	Y	N	N
DES-CBC3-SHA	192	Y	Y	Y	Y	N

Cipher Suites	Bits	SSL Protocol – Virtual Host				
EXP-RC4-MD5	40	Y	N	N	N	N
EXP-DES-CBC-SHA	40	Y	N	N	N	N
AES128-SHA	128	Y	Y	Y	Y	N
AES256-SHA	256	Y	Y	Y	Y	N
AES128-SHA256	128	N	N	N	Y	N
AES256-SHA256	256	N	N	N	Y	N
AES128-GCM-SHA256	128	N	N	N	Y	N
AES256-GCM-SHA384	256	N	N	N	Y	N
ECDHE-RSA-AES128-SHA	128	Y	Y	Y	Y	N
ECDHE-RSA-AES256-SHA	256	Y	Y	Y	Y	N
ECDHE-ECDSA-AES128-SHA	128	Y	Y	Y	Y	N
ECDHE-ECDSA-AES256-SHA	256	Y	Y	Y	Y	N
ECDHE-RSA-AES128-SHA256	128	N	N	N	Y	N
ECDHE-RSA-AES256-SHA384	256	N	N	N	Y	N
ECDHE-RSA-AES128-GCM-SHA256	128	N	N	N	Y	N
ECDHE-RSA-AES256-GCM-SHA384	256	N	N	N	Y	N
ECDHE-ECDSA-AES128-SHA256	128	N	N	N	Y	N
ECDHE-ECDSA-AES256-SHA384	256	N	N	N	Y	N
ECDHE-ECDSA-AES128-GCM-SHA256	128	N	N	N	Y	N
ECDHE-ECDSA-AES256-GCM-SHA384	256	N	N	N	Y	N
TLS-AES128-GCM-SHA256	128	N	N	N	N	Y
TLS-AES256-GCM-SHA384	256	N	N	N	N	Y

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.

Configure protocol version for SSL virtual host.

AN(config)#**ssl settings protocol "www.example.com" "SSLv3:TLSv1:TLSv12"**

SSL protocols allowed to be set include SSLv3, TLSv1, TLSv11, TLSv12 and TLSv13.. To set multiple protocols, separate them with colons (:).

Note: Parameter value “TLSv11” stands for the TLSv1.1 protocol, “TLSv12” stands for the TLSv1.2 protocol, “TLSv13” stands for the TLSv1.3 protocol.

Configure session reuse for SSL virtual host.

AN(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.

Configure client authentication for SSL virtual host.

The APV appliance supports the SSL based client authentication. You can enable client authentication for an SSL virtual host. If enabled, the APV appliance will require each client to present an SSL certificate for authorization, before the client can access the SSL virtual host.

AN(config)#**ssl settings clientauth "www.example.com"**

Note: If you enable SSL client authentication for an SSL virtual host, you must provide a trusted CA certificate. This will be used by the APV appliance to verify client certificates.

AN(config)#**ssl import rootca "www.example.com"**

This command will prompt you to cut and paste the trusted authority certificate in PEM format. You may configure multiple trusted authorities for one SSL virtual host.

Furthermore, the SSL virtual host will check the client certificate based on the configured certificate filters (by using the command "**ssl settings certfilter**"). If the client certificate fails the certificate verification, the SSL host will reject the client's access. At most three pieces of "certfilter" configuration (by using the "**ssl settings certfilter**" command) can be configured for an SSL virtual host. The logical relationship among the three pieces of "certfilter" configuration is "OR". If the client certificate does not match any piece of "certfilter" configuration, the SSL virtual host will reject the client's access.

The filters can be configured with any of the supported RDNs on the APV appliances.

14.4.3.1.1.1 Supported RDN on APV

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

RDN	Standard Name
givenName	Given Name
Surname	Surname
DC	Domain Component
emailAddress	Email Address
{OID expression}	OID information, for example: 1.2.3.4

For example:

```
AN(config)#ssl settings certfilter vhost
"subject:/C=US/O=Organization/OU=QA/emailAddress=admin@abc.com"
"issuer:/C=US"
```

In this example, client certificates can pass the certificate verification only when the following conditions are both met:

- In the “subject” field, “C” is “US”, “O” is “Organization”, “OU” is “QA” and “emailAddress” is “admin@ abc.com”.
- In the “issuer” field, “C” is “US”.

Otherwise, the client will fail the authentication.

Two kinds of client authentication modes are supported: mandatory and non-mandatory. Client authentication mode defaults to mandatory. In non-mandatory client authentication mode, when the server sends a certificate request to the client, if the client has no matched certificate or cancels the authentication by clicking the Cancel button, the server will permit the client to access limited network resources instead of dropping the SSL connection. However, all the networks resources which can be published to non-authenticated clients need to be defined by using the “**http acl url**” command.

Configure CRL for SSL virtual host.

APV supports the CRL (Certificate Revocation List) functionality. You can configure the APV appliance to fetch the CRL file periodically from a CRL Distribution Point (CDP) by using HTTP or FTP.

For our example, let’s consider a case when you have put your CRL file (Array.crl) on an HTTP Web server (www.crldp.com) and you want to fetch it every one minute.

You can configure the APV appliance as follows:

```
AN(config)#ssl settings crl offline www.example.com cdp1 "http://www.crldp.com/Array.crl"
1
```

This will cause the APV 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.

```
AN(config)#ssl settings crl offline www.example.com cdp1 "ftp://ftp.crldp.com/Array.crl" 1
```

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

```
AN(config)#ssl settings crl offline www.example.com cdp1
"ldap://ldap.crldp.com/cn=company,dc=organization,dc=com" 1
```

Configure OCSP for SSL virtual host to check the certificate validation online.

The APV appliance supports the OCSP (Online Certificate Status Protocol) protocol. You may configure the APV 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 APV appliance as follows:

```
AN(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.

Configure redirect for clients without strong encryption support.

The APV 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:

```
AN(config)#ssl settings minimum www.example.com 168 "http://www.example2.com"
```

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.

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

14.4.3.2 Advanced SSL Configuration for SSL Real Host

1. Dsiable SSL real host.

```
AN(config)#ssl stop www.myreal.com
```

SSL real host must be disabled before you change its configurations.

Configure cipher suites for the SSL real host.

```
AN(config)#ssl settings ciphersuite "www.myreal.com" "DES-CBC3-SHA"
```

The following table lists the RSA and ECC (ECDHE-ECDSA…) cipher suites allowed for an SSL real host with different SSL protocol versions. “Y” indicates that the cipher suite is supported. “N” indicates that the cipher suite is not supported.

Cipher Suite	Bits	SSL Protocol – Real Host				
		SSLv3.0	TLSv1.0	TLSv1.1	TLSv1.2	TLSv1.3
RC4-MD5	128	Y	Y	Y	Y	N
RC4-SHA	128	Y	Y	Y	Y	N
DES-CBC3-SHA	192	Y	Y	Y	Y	N
AES128-SHA	128	Y	Y	Y	Y	N
AES256-SHA	256	Y	Y	Y	Y	N
AES128-SHA256	128	N	N	N	Y	N
AES256-SHA256	256	N	N	N	Y	N
AES128-GCM-SHA256	128	N	N	N	Y	N
AES256-GCM-SHA384	256	N	N	N	Y	N
ECDHE-RSA-AES128-SHA	128	Y	Y	Y	Y	N
ECDHE-RSA-AES256-SHA	256	Y	Y	Y	Y	N
ECDHE-ECDSA-AES128-SHA	128	Y	Y	Y	Y	N
ECDHE-ECDSA-AES256-SHA	256	Y	Y	Y	Y	N
ECDHE-RSA-AES128-SHA256	128	N	N	N	Y	N
ECDHE-RSA-AES256-SHA384	256	N	N	N	Y	N
ECDHE-RSA-AES128-GCM-SHA256	128	N	N	N	Y	N
ECDHE-RSA-AES256-GCM-SHA384	256	N	N	N	Y	N
ECDHE-ECDSA-AES128-SHA256	128	N	N	N	Y	N
ECDHE-ECDSA-AES256-SHA384	256	N	N	N	Y	N
ECDHE-ECDSA-AES128-GCM-SHA256	128	N	N	N	Y	N
ECDHE-ECDSA-AES256-GCM-SHA384	256	N	N	N	Y	N
TLS-AES128-GCM-SHA256	128	N	N	N	N	Y
TLS-AES256-GCM-SHA384	256	N	N	N	N	Y

Configure protocol version for SSL real host.

AN(config)#**ssl settings protocol "www.myreal.com" "SSLv3:TLSv1"**

SSL protocols allowed to be set include SSLv3, TLSv1, TLSv11, TLSv12 and TLSv1.3. To set multiple protocols, separate them with colons (:).

Configure session reuse for SSL real host.

This allows you to enable SSL session reuses between the APV appliance and backend servers. This feature is enabled by default.

AN(config)#**ssl settings reuse www.myreal.com**

Configure client authentication for SSL real host.

The APV appliance can use SSL client authentication while communicating with the backend server. If this setting is enabled, the APV appliance will submit the client certificate to the backend sever for authentication during SSL handshake.

AN(config)#**ssl settings clientauth www.myreal.com**

Note: If you want to enable client authentication for an SSL real host, you will need to import a certificate and key pair for the SSL real host. The SSL real host will present this certificate to the backend server for authentication. This may be accomplished by using the "**ssl import certificate**" and "**ssl import key**" commands for an SSL real host. These two commands work exactly the same for an SSL virtual host and an SSL real host. For detailed instruction on using these commands, please refer to the SSL virtual host configuration described earlier.

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:

AN(config)#**ssl settings servername www.myreal.com "Myreal Inc."**

Import trusted CA certificate for SSL real host.

Since the SSL subsystem acts like a client to the real server, it has several root CA certificates just like a common Web browser. If you are using a self-signed certificate, or a certificate issued by your own local CA on your origin servers, then you need to use the "**ssl import rootca**" command to import the self-signed certificate that is on the real server or the local CA certificate.

The certificate must be in PEM format and is imported the same way you import a PEM certificate. The APV appliance will prompt you to cut and paste the text to the terminal and enter "..." to accept the certificate.

AN(config)#**ssl import rootca**

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.

AN(config)#**ssl start www.myreal.com**

15SSL Interception

With the SSL interception function, the APV appliance can intercept SSL traffic, which is encrypted, and send the decrypted data to security devices such as the firewall, IPS and IDS. After the decrypted data is inspected by the security device, it will be sent to the APV appliance for re-encryption and then forwarded to the real application servers. This helps prevent attacks, intrusion and data exfiltration aroused by insecure information encrypted in SSL data. In the SSL interception process, the APV appliance can cooperate with security devices set up in L2 or L3 mode. Thereare three distinct stages in the SSL interception process:

1. The ingress node receives and decrypts SSL traffic coming from clients and sends the decrypted data to the security device.
2. The security device performs inspection on the decrypted data, filters out those containing security risks and sends the inspected data to the egress node.
3. The egress node encrypts the data again and then forwards it to the real destination.

Based on the distribution mode, the ingress and egress nodes can be deployed in the integrated mode and the distributed mode.

- Integrated mode: The ingress and egress nodes are integrated on one APV appliance. That is, one APV plays the role of both the ingress and egress nodes.
- Distributed mode: SSL interception is implemented on two APV appliances. One plays the role of the ingress node, and the other plays the role of the egress node.

The APV appliance implements SSL interception by simulating server certificates. Administrators need to install the certificates trusted by clients on the ingress node to obtain and decrypts SSL traffic coming from the clients. How to install the simulated certificates depends on whether the actual network uses the PKI system.

- On the networks where PKI is not deployed, the CA certificate and key must be generated on the ingress node (by using the "**ssli cacert…**" command). The client browser must also have the CA certificate installed. When the ingress node obtains the server certificate, it will re-sign the server certificate using the generated CA certificate to create a simulated certificate, and then use the private key corresponding to the public key in the simulated certificate to decrypt SSL traffic coming from clients.
- On a network where PKI is deployed, the ingress node and client browsers must have an intermediate CA certificate and key installed. This intermediate CA certificate must be issued by a CA that is trusted by the client. When the ingress node obtains the server certificate, it will re-sign the server certificate using the imported intermediate CA certificate, and then use the private key corresponding to the public key in the simulated certificate to decrypt SSL traffic coming from clients.

Each simulated certificate will be cached in the system and set with a timer (by the "**ssli settings certcache timeout**" command). When the timer ends, the simulated certificate will be removed

out of the cache. If a simulated certificate is hit again before the timer ends, the timer will be renewed.

15.2 Domain List

When the SSL interception function is enabled for a SSL virtual host, the system will intercept all SSL traffic of this virtual host by default. However, for privacy protection, some SSL traffic, for example the communications to sensitive websites such as banking and healthcare applications should be forwarded without decryption and inspection.

To protect privacy, the system supports the domain list function. The domain list function can be enabled or disabled by the "**ssli domainlist {on|off}**" command. When it is disabled, the system intercepts all SSL traffic by default. When it is enabled, the domain list function employs two types of domain lists to define SSL traffic that needs to be intercepted or not.

- Bypass domain lists: SSL traffic that accesses the domain strings or domain names contained in this domain list is allowed to pass through in encrypted format without being inspected by a security device.
- Interception domain lists: SSL traffic that accesses the domain strings or domain names contained in this domain list must be decrypted and then inspected by a security device before passing through.

A virtual host that has n domain list applied will perform the bypass or interception operations based on the type of the applied domain list. A virtual host can have only one type of domain lists applied. For example:

1. Create a bypass domain list "list_1".

```
AN(config)#ssli domainlist list list_1 bypass
```

Add domain string items "abc*" and "cde*" as items to "list_1".

```
AN(config)#ssli domainlist item list_1 "abc*"
```

Apply "list_1" to virtual host "vhost".

```
AN(config)#ssli domainlist apply list vhost list_1
```

For simplification purposes, the domain list function also supports the application of a single bypass or interception domain string to a virtual host via the "**ssli domainlist apply item**" command. For example, when only SSL traffic accessing domain name "xyz*" needs to be intercepted, you can run the following command to apply "xyz*" as an interception domain string to the virtual host. It is not necessary to add the domain string to an interception domain list and then apply the list to the virtual host.

```
AN(config)#ssli domainlist apply item vhost2 "xyz*" intercept
```

Note: When the ingress node attempts to obtain the server certificate, if the server requires the client certificate or if the SSL handshake encounters a failture, the SSL traffic will also

pass through without being decrypted.

15.3 URL Filtering

As stated in section “13.1 Domain List”, the SSL interception module supports manual configurations of interception domain lists to define SSL traffic to be intercepted. Administrator can also manually add bypass domain lists to define SSL traffic to be bypassed. The domain list function applies if clients access only a few types of websites and the manual configuration workload is light.

If the application scenario accommodates accesses to various types of websites, traffic filtering using domain lists configurations requires the administrator to manually configure a great number of domain lists. In addition, the administrator cannot fully understand the internet. Clients’ accesses to some websites may be unnecessarily intercepted. In this case, it is recommended to configure the website classification function to achieve intelligent URL filtering. Via website classification, the APV appliance supports recognition of 82 website categories. For details about these categories, please refer to: http://www.brightcloud.com/tools/change-request-url-categorization.php.

The website classification function is disabled for all virtual hosts by default. Administrators can use the “**ssli webclassify {on|off}**” command to enable or disable this function on each virtual host.

15.3.2Filtering Policy

Using this function, administrators do not need to define a traffic filtering policy for every website to be intercepted or bypassed. Instead, administrators can define interception or bypass policies for specific types of websites. For example, to protect user privacy, administrators can define bypass policies for accesses to financial service and healthcare sites.

Similar to the domain list function, the website classification function supports definition of either of the following filtering policies to distinguish SSL processing modes.

- Interception policy: all traffic accessing websites belonging to the configured website category will be decrypted and then sent to security devices for inspection. Traffic accessing website categories that are not defined by the policy will be forwarded transparently. Administrators can use the “**ssli webclassify url intercept**” command to define the website category to be intercepted.
- Bypass policy: all traffic accessing websites belonging to the configured website category will be transparently forwarded without being decrypted and inspected by security devices. Traffic accessing website categories that are not defined by the policy will be intercepted. Administrators can use the “**ssli webclassify url bypass**” command to define the website category to be bypassed.

On the same virtual host, interception policies and bypass policies cannot be configured simultaneously, that is, each virtual host can be configured with only one type of filtering policies.

When a virtual host configured with both the domain list and website classification fucntions receives a request or response, it will take actions according to the following principles:

- The virtual host will preferentially match the domain lists configured by the domain list function.
  - If a matching entry is found, it will process the request or response based on the control type (interception or bypass) of the domain list.
  - If no matching entry is found in the domain lists, it will look up the website's category using the website classification function.
- When trying to determine the website category using the website classification function, the virtual host will first search in the local cache and database.
  - If the local cache or database has category information for the website, the virtual host will process the request or response based on the control type (interception or bypass) of the filtering policy.
  - If no category information is available in the local cache and database, the system will connect to the Webroot server to query the website category online and save the acquired category information to the local cache.
  - If the Webroot server cannot recognize the website category, the system will intercept the request or response by default. This default behavior can be modified using the "**ssli webclassify defaction**" command.

15.3.3License

To use the website classification function to achieve intelligent URL filtering, please contact Customer Support and provide the device model and serial number to obtain a license. This function supports two types of licenses:

- Free trial license: 30-day free trial
- Formal license: 365-day validity period

After the license expires, the website classification function will be unavailable. The Webroot server will deny the APV appliance's access and the APV appliance will stop sending website category lookup queries to the server.

15.3.4Configuration Example

1. Import the website classification function license.

```
AN(config)#webclassify license
41ce070c-4baa4482-02bc5237-f62e21b8-b50b177c-00000000-00000001-20171220-20180119
```

Enable the global website classification function.

```
AN(config)#webclassify on
```

Enable the online website classification lookup function.

```
AN(config)#webclassify cloud on
```

Configure the virtual host to bypass traffic accessing unrecognized websites on the SSL interception module.

```
AN(config)#ssli webclassify defaction vhost 0
```

Define healthcare and financial service sites as bypass website categories.

```
AN(config)#ssli web url bypass vhost "Health & Medicine"
AN(config)#ssli web url bypass vhost "Financial Services"
```

Enable the website classification function for the SSL interception virtual host.

```
AN(config)#ssli webclassify on vhost
```

15.4 Load Balance of Security Devices

While cooperating with security devices to implement SSL interception, the APV appliance also supports the load balance of security devices, by which it distributes traffic evenly to multiple security devices. In addition to the SSL interception application scenario (forward proxy topology), this function is also supported in the reverse proxy topology.

In a reverse proxy topology, the APV appliance processes SSL traffic in the same way as that in an SSL acceleration scenario. As the SSL virtual host has the certificate and key of the real server installed, SSL traffic will be decrypted on the ingress node and then distributed to security devices based on a load balance method. Finally, the traffic will be sent to the real servers via the egress node.

In a forward proxy topology, administrators need to configure CA certificates on the ingress node. The ingress node will generate the simulated certificates to obtain and decrypt SSL traffic and then distributes the traffic to security devices in load balance mode.

For information about how to configure SSL interception and load balance of security devices in these scenarios, please contact Customer Support to obtain the SSL interception configuration guide.

16Secure Application Access（SAA）

To protect specific network resources, most websites will employ the authentication mechanism to restrict end user’s access. Only end users that have passed the authentication can access specific network services. Meanwhile, in some large-scale websites, one click of an end user may require the coordination of multiple sub systems. If the end user has to do a round of authentication in every subsystem, this will not only increase the complexity of the authentication logic, but also impact user experience. Thus, an authentication system is always required to support Single Sign-On (SSO) at the same time.

The Secure Application Access (SAA) function provides AAA authentication and authorization methods based on SAML, LDAP, RADIUS and OAuth and Web SSO, which meets the requirement for user identity authentication in multiple application scenarios.

16.2 AAA

AAA is a series of combined features and operations that provide authentication, authorization and accounting functions for all connections and transactions on the internet. AAA provides a security architecture that enables control over which users are allowed access which services and how much of resources they have used. This empowers the administrators to tightly control user access to content, grant users with specific authorities to internal resources and realize the charging for services.

Currently, the APV appliance supports the authentication and authorization functions of the AAA feature.

16.2.2AAA Server

The APV appliance performs authentication and authorization of end users by using AAA servers. An AAA server is any entity that is able to authenticate or authorize a user by using AAA. The APV appliance supports the following types of AAA servers:

- Lightweight Directory Access Protocol (LDAP)
- Remote Authentication Dial In User Service (RADIUS)
- Security Assertion Markup Language (SAML) SP
- Open Authorization (OAuth)

16.2.2.1 SAML

SAML defines an XML-based framework, in which every party creates and exchanges authentication, authorization and attribute information. On the APV appliance, SAML authentication is implemented based on the Security Assertion Markup Language (SAML) 2.0 standards. There are three entities in the SAML framework: subject, Identity Provider (IdP) and Service Provider (SP).

Note: The timezone, date and time settings of the IdP and SP servers must be the same (at least in the minute level).

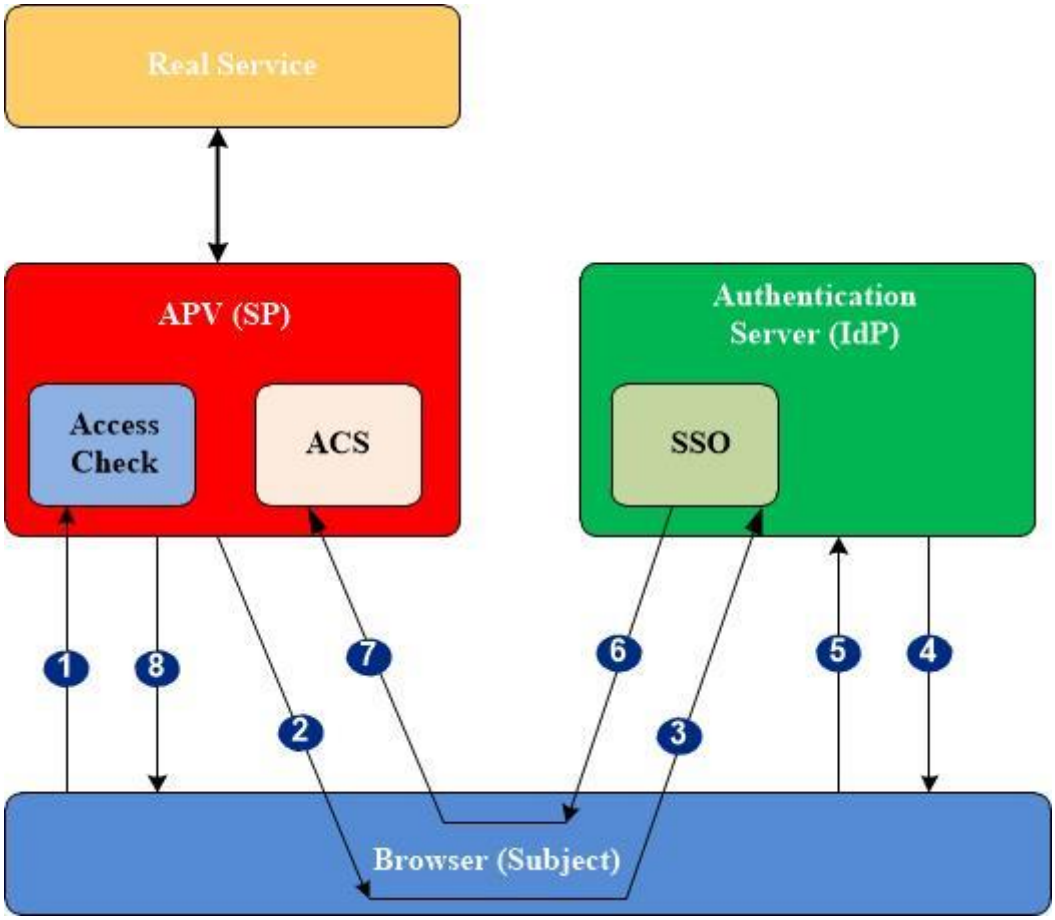
16.2.2.1.1 Metadata

Metadata defines how necessary configuration information is shared between two entities. Before starting authentication based on SAML, SP and IdP must own each other's metadata file. When any of the metadata files is updated, the administrator must import the new one into the entity. A metadata file contains the following information:

- Entity ID
- Entity's certificate, for signature and encryption
- The URL address and protocol binding types supported by the SSO service, Single Logout (SLO) service, Assertion Consumer Service (ACS) and Artifact Resolution Service (ARS). Protocol binding defines the transport layer protocol used by SAML messages exchanged between SP and IdP. On the APV appliance, the SLO service supports HTTP Redirect and HTTP POST protocol binding types, and the ACS service supports the HTTP POST and HTTP Artifact protocol binding types.
  - HTTP Redirect: transports SAML messages via HTTP redirect (302 response) messages.
  - HTTP POST: transports SAML messages via Base64-encoded HTML content.
  - HTTP Artifact: transports the reference of SAML messages using HTML content or URL query strings.
- Other information like contact and attribute requirements

16.2.2.1.2 SSO Process

In the process of SAML-based SSO, the APV appliance acts as the SP server and authenticates end users based on the user identity information provided by the IdP server. The two parties exchange information based on the SAML standards. The following figure shows the basic process of SAML-based SSO:



1. The client accesses the SP server using a browser and requests resources on real services.

The SP creates a SAML authentication request and redirects the client to the IdP for authentication. The request carries the protocol binding type (configured by the "**aaa samlsp sp acs**" command) supported by the ACS module.

The client submits the SAML authentication request to the IdP.

The IdP returns an authentication page to the client, requesting the login credential.

The client enters the login credential.

The IdP validates the login credential and returns a SAML response to the client, which contains an assertion about the user identity. An example assertion is "This user is John Doe, he has an email address of john.doe@example.com, and he was authenticated into this system using a password mechanism."

The client browser sends the SAML response to the SP's ACS module.

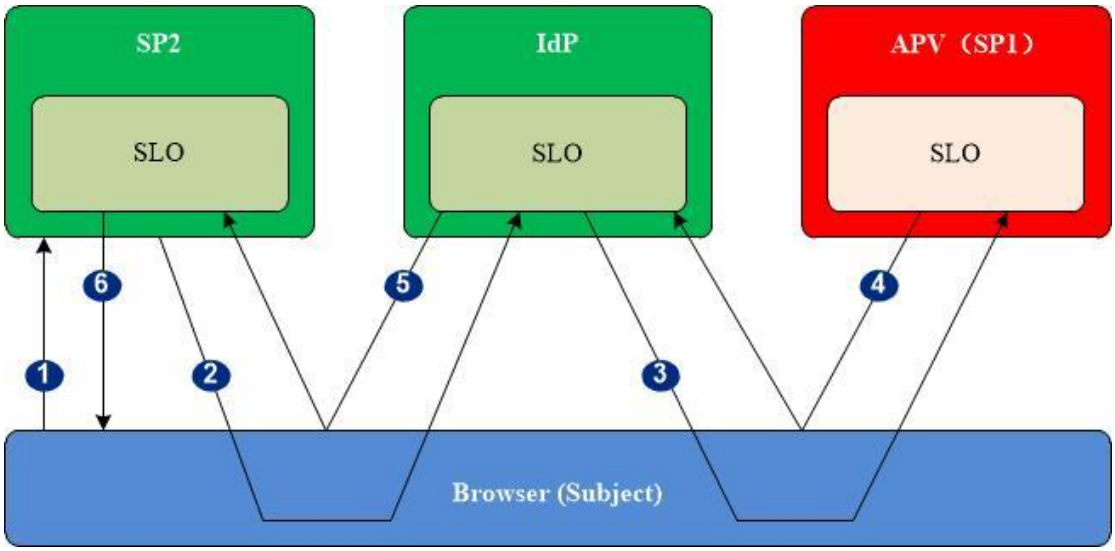
The SP extracts the username (configured by the "**aaa samlsp idp attribute**s" command) from the SAML response, confirming that the client has been authenticated and then it returns resources.

After this process is completed, if this client accesses other resources, the APV appliance will directly return the resources based on the existing authentication context. The client does not need to enter the login credential again.

16.2.2.1.3 SLO Process

Based on SAML SSO, an end user can access multiple sites using the same authentication context. Thus, it will have a login session on all sites. Comparably, the function of SLO is to roll back the SAML SSO processes of all sessions, that is, log out login sessions on all login sites at the same time. For example, when an end user logs into multiple SP sites via the same IdP server, if the user logs out from one of these SP sites, the login sessions on other sites will also be removed.

This following figure shows the basic process of SAML SLO:



1. The client requests to log out of the SP2 server.

SP2 creates a SAML logout request and redirects the client to the IdP for logout.

The IdP creates a SAML logout request and sends it to SP1's SLO module via the client's browser by using the protocol binding type defined in SP1's metadata file.

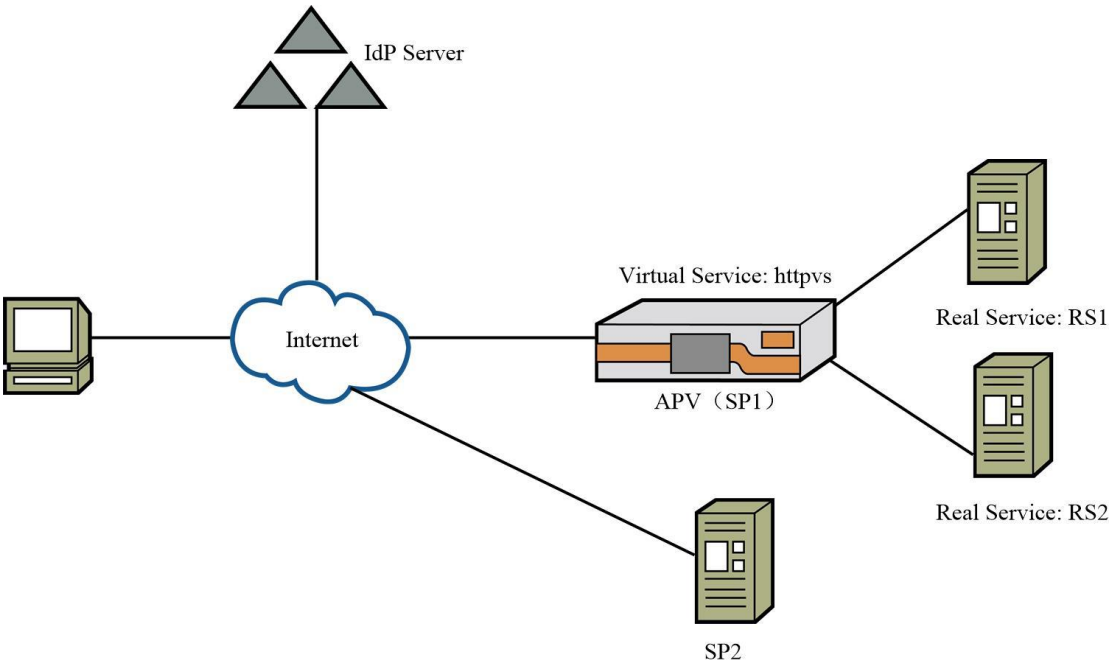
SP1 creates a SAML logout response to inform the IdP that the client's login session is removed.

The IdP returns a logout request via the client browser to inform SP2 that the client has logged out from other SP.

SP2 returns a logout response to the client, and the client is logged out.

16.2.2.1.4 Configuration Example

The following figure shows the configuration scenario:



The configuration requirement is that the APV appliance can perform SAML authentication for clients accessing real services “RS1” and “RS2” via the IdP server.

- **SLB and SSL Configurations**

SLB configurations refer to basic configurations such as virtual service, real service, policy and method configurations. Please refer to Chapter 7 Server Load Balancing (SLB) for the configuration methods.

SSL configurations refer to SSL host and certificate configurations. Please refer to 14 Secure Sockets Layer (SSL).

- **SAML Authentication Configuration**

1. Configure a SAML SP type AAA **server.**

```
AN(config)#aaa server samlsp saml_sp
```

Set the AAA server “saml_sp” as AAA method “samlsp_method”.

```
AN(config)#aaa method server samlsp_method saml_sp
```

Associate “samlsp_method” with virtual service “httpsvs”.

```
AN(config)#aaa method bind samlsp_method httpsvs
```

Import the IdP server’s metadata file into the SAML SP server.

```
AN(config)#aaa samlsp idp metadata saml_sp “http://idp.metadata/file”
```

Configure the attribute name based on which the SAML SP server obtains user identity information.

```
AN(config)#aaa samlsp idp attributes saml_sp “subject.nameid”
```

Configure the protocol binding type supported by the SAML SP server's ACS.

```
AN(config)#aaa samlsp sp acs saml_sp post
```

Configure the protocol binding type supported by the SAML SP server's SLO service.

```
AN(config)#aaa samlsp sp slo saml_sp redirect
```

Import the SAML SP server's metadata file into the IdP server.

You can use the "**show aaa samlsp metadata**" command to view the download address of the SAML SP server's metadata file, for example:

```
AN(config)#show aaa samlsp sp metadata saml_sp
https://10.8.6.140/prx/000/http/hostlocal/saml2/server1/module.php/saml/sp/metadata.php/server1
```

Add user role "staff".

```
AN(config)#aaa role name staff
```

Add a qualification rule "work" for "staff".

```
AN(config)#aaa role qualification staff work
```

Add a condition to the qualification rule of user role "staff". For example, all users that log in on the first day of each month will obtain the user role "staff".

```
AN(config)#aaa role condition staff work "LOGINDAY IS 1"
```

Associate the user role "staff" with virtual service "httpsvs".

```
AN(config)#aaa role bind staff httpsvs
```

Enable the AAA function.

```
AN(config)#aaa on httpsvs
```

- **(Optional) Session Management Configuration**

1. Set the total number of sessions allowed on virtual service "httpsvs".

```
AN(config)#aaa session virtual limit httpsvs 20000
```

Set the number of sessions allowed per user on virtual service "httpsvs".

```
AN(config)#aaa session virtual maxperuser httpsvs 200
```

Enable the session reuse function.

```
AN(config)#aaa session virtual reuse on
```

Set the timeout value of idle sessions.

```
AN(config)#aaa session virtual timeout idle httpsvs 3600
```

Set the session expiration time.

AN(config)#**aaa session virtual timeout lifetime httpsvs 3600**

16.2.2.2 LDAP

On the APV appliance, the LDAP authentication and authorization method supports all LDAPv3 protocols, including OpenLDAP and Active Directory. To prevent single of point failures, each LDAP server can be configured with three LDAP hosts. Administrators can configure authentication and authorization modes based on TLS for each LDAP host.

LDAP supports static and dynamic LDAP bind modes. The two modes cannot be configured simultaneously.

In dynamic LDAP bind mode, AAA will fetch the DN from the LDAP server first. After the dynamic LDAP bind mode is enabled, AAA sends a bind request containing the end user's username and password to the LDAP server and then a search request containing the search filter string configured by the command "**aaa ldap searchfilter**" to obtain the LDAP entry of the end user. Then AAA sends the DN obtained from the LDAP entry together with the password of the end user in another bind request to the LDAP server. After the end user passes the authentication, AAA reuses the obtained LDAP entry to authorize the end user.

In static LDAP bind mode, the system will construct the user's DN by concatenating the strings "<dn_prefix><USER><dn_suffix>". <USER> is the username used to log into the virtual site. "<dn_prefix>" and "<dn_suffix>"must be the same for all users using the same virtual site. After the static LDAP bind mode is enabled, AAA sends the DN (<dn_prefix><USER><dn_suffix>) together with the password of the end user in a bind request to the LDAP server. After the end user passes the authentication, AAA sends a search request containing the search filter string configured by the command "**aaa ldap searchfilter**" to obtain the LDAP entry of this end user. Then, it authorizes the end user based on the obtained LDAP entry.

16.2.2.3 RADIUS

The RADIUS authentication and authorization mode supports up to three hosts configured for each RADIUS server to achieve redundancy. RADIUS requests are non-blocking and can be controlled by the timeout mechanism to ensure the transfer efficiency.

16.2.2.4 OAuth

OAuth allows end users to grant authorization to third-party applications or websites to access their resources saved on other service providers. End users do not need to provide their user names and passwords to these applications or websites. During OAuth authentication process, there are mainly four roles:

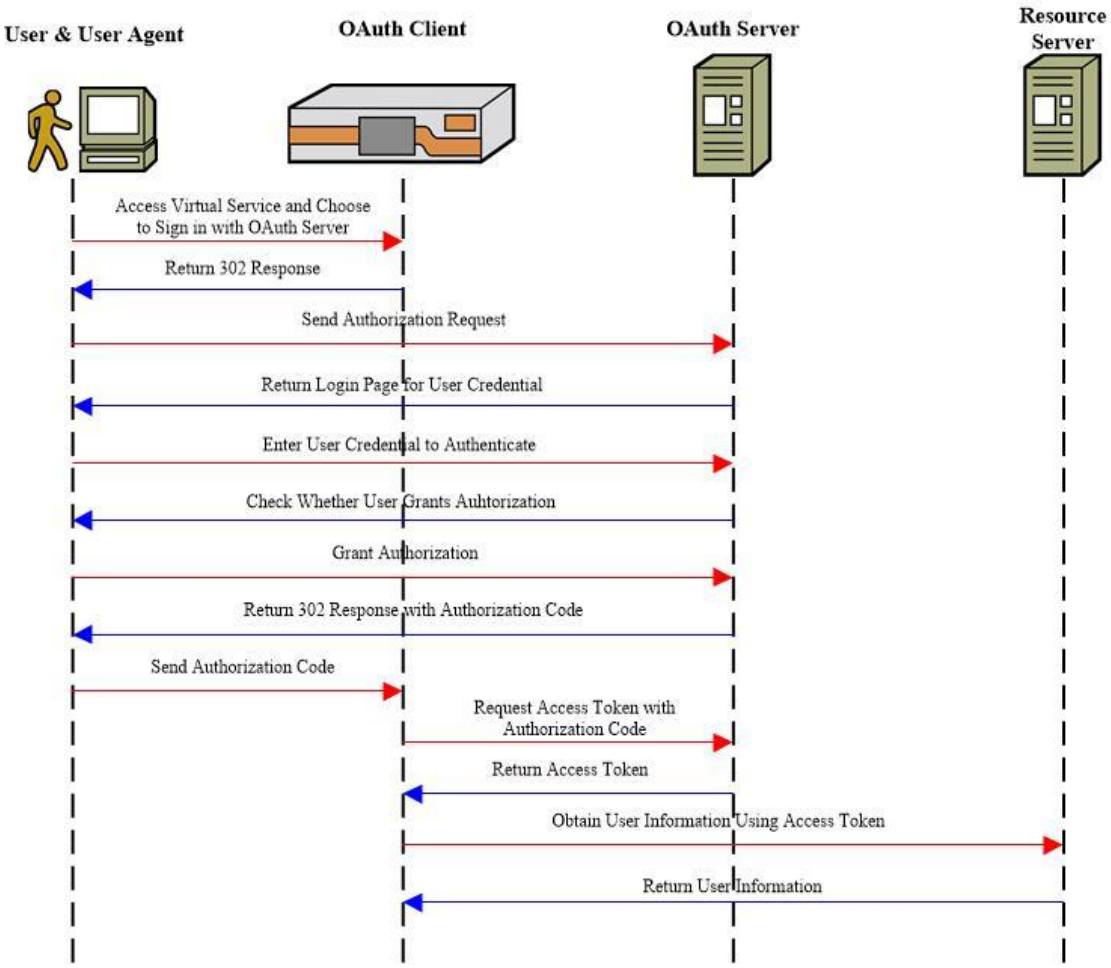
- Resource owner: indicates the end users.
- Resource server: indicates the service provider that stores the protected resources. To access these resources, the requester must obtain an access token.

- Client: represents the third-party applications that request for protected resources.
- Authorization server: after verifying the resource owner's identity and obtains an authorization grant, the authorization server will issue an access token to the client.

The APV appliance supports OAuth authentication via Google and WeChat login. When an end user accesses the virtual service, the APV appliance will redirect the user to the OAuth server of Google or WeChat for authentication and authorization. After the user passes authentication and grants authorization to the OAuth client, the OAuth client will request an access token from the OAuth server and use the access token to obtain user information and resources from the resource server.

16.2.2.4.1 Authentication Process

The following figure shows the work flow in which the APV appliance cooperates with the OAuth server and resource server to implement the OAuth authentication.



The detailed description of the workflow above is as follows:

1. The end user accesses the virtual service using the user agent (browser) and chooses to sign in with the third-party OAuth server.

The OAuth client (integrated on APV) returns a 302 response to redirect the user agent to the OAuth server.

The user agent sends an Authorization Request to the OAuth server.

The OAuth server validates the Authorization Request and returns the login page to the user for authentication.

The user enters the correct user credential.

The OAuth server checks whether the user grants the authorization to the OAuth client if the user passes the authentication.

The user grants the authorization to the OAuth client.

The OAuth server returns a 302 response with the authorization code.

The user agent sends the authorization code to the OAuth client.

The OAuth client sends the Access Token request to the OAuth server with the authorization code.

The OAuth server authenticates the OAuth client and returns the Access Token response carrying the issued access token.

The OAuth client requests the user information from the resource server with the access token.

The resource server returns the user information (such as user ID, email account, nickname and avatar picture) to the OAuth client.

As the OAuth 2.0 framework requires, the OAuth client should authenticate itself to the OAuth server. For this purpose, you need to register the OAuth client to obtain the Client ID and Secret and register the Redirection URL on the developer platform of the OAuth server's service provider. For the Google OAuth server, the Redirection URL must be in the format of "https://<virtual_service_domain_name>/prx/000/http/hostlocal/oauth_code". For the WeChat OAuth server, the Redirection URL must be the virtual service domain name. For information on how to register the OAuth client and the Redirection URL, please contact the service provider of the OAuth server.

Note:

3. OAuth authentication does not support multi-factor authentication or multi-step authentication.
4. The AAA method configured for OAuth authentication is not controlled by the AAA ranking function and will always be available on the login page for end users to choose.
5. To log into the virtual service using a third-party OAuth server, end users must access the virtual service using the domain name contained in the Redirection URL.

16.2.2.4.2 Advanced Settings

OAuth authentication supports the following advanced settings:

- **Post-OAuth User Registration**

With post-OAuth user registration, the OAuth authentication function allows you to bind the obtained OAuth usernames with existing company accounts saved on the AAA server.

When post-OAuth user registration is enabled, OAuth users are required to register to the system after passing the OAuth authentication. During the user registration, users need to authenticate themselves to the authentication server in the AAA method specified by the "**aaa method register**" command. After the user passes the authentication, the system will bind the obtained OAuth user IDs (UIDs) with the usernames used for registration. The usernames used for registration instead of the obtained OAuth usernames will be used for further authorization.

When post-OAuth user registration is disabled, the obtained OAuth usernames (email accounts for the Google OAuth server or nicknames for the WeChat OAuth server) will be used for authorization. Therefore, the authorization server in the same AAA method as the OAuth server should have accounts with the same usernames as the obtained OAuth usernames. Otherwise, the authorization will fail.

- **Using Email Account Prefix as Username**

When the Google OAuth server is configured and post-OAuth user registration is disabled, administrators can enable this option so that the email account will be used as OAuth username for authorization. For example, if the obtained email account is "test@gmail.com", only "test" will be used as the username for further authorization.

- **Post-OAuth Authorization Filter**

With the post-OAuth authorization filter function, the OAuth authentication function ensures that only valid company users can go through authorization after OAuth authentication. The system performs authorization for the user only when the OAuth username (email account for the Google OAuth server or nickname for the WeChat OAuth server) matches the post-OAuth authorization filter.

For example, when the post-OAuth authorization filter is configured as "@admin.net", the system performs authorization for the user if the obtained OAuth username is "test@admin.net".

16.2.3AAA Method

The APV appliance employs AAA method to define the AAA server used to authenticate users and the AAA server used to authorize users. For example, the administrator can define an AAA method in which LDAP will be used for authentication and RADIUS will be used for authorization.

16.2.3.1 Multi-factor Authentication

To enforce stricter security checks on users and ensure a higher level of security for the virtual service, APV allows the administrator to configure multiple authentication servers for a single AAA method to support multi-factor authentication (mutual username and multiple passwords).

The user can successfully logs into the virtual service only after passing authentication from all authentication servers.

A maximum of three authentication servers are allowed for one AAA method. These three authentication servers can be of the same type or different types. However SAML SP server cannot cooperate with other servers.

16.2.3.2 AAA Method Rank

When an application scenario has multiple AAA servers deployed and user credentials are stored on these AAA servers in a distributed manner, the administrator can define several AAA methods for a virtual service. Before a user logs into the virtual service, the user can select an AAA method for authentication as prompted. If the administrator does not want the users to know what authentication methods are used by the virtual service, the APV appliance supports the AAA rank function. With AAA rank, the APV appliance will hide AAA method options on the login page, after the user enters credentials, the APV appliance will select among the AAA methods based on the descending order of their priorities until the user passes authentication.

16.2.3.3 Authorization

During authorization, the APV appliance will obtain authorization data from the authorization server. Currently, the authorization data that can be obtained is the group to which a user belongs. The obtained authorization data will be further used for user role assignment.

16.2.4User Role

The APV appliance authorizes authenticated users with resources based on the qualifications of user roles. User roles allow more flexible, accurate and fine-grained user identification. Administrators can define one or multiple qualification rules for each user role, and each qualification rule can contain at most 32 conditions. Supported conditions include login time, user name, group name, source IP address and the used AAA methods etc.

Before accessing resources via a virtual service, the user must obtain at least one user role; otherwise, the APV appliance will log the user out and ask the user to log into the virtual service again. A user may match multiple qualification rules at the same time but can only obtain a user role when matching all conditions contained in the qualification rule. A user that does not match any qualification rule cannot obtain a user role, and therefore cannot obtain any resources via the APV appliance. Administrators can also define a user role for which the qualification rule does not contain any conditions. All authenticated users will be assigned this user role.

16.2.5Session Management

After an end user completes authentication, the system will create a login session for this user, which records information such as session ID, username and password. Administrators can use the

“**show aaa session active**” command to view existing authenticated sessions. Besides, it also supports the following configurations to manage authenticated sessions:

- Restriction on the number of sessions

The system supports the setting of limitations on the total number of sessions allowed on each virtual service and the number of sessions allowed per user on a virtual service. If the number of current active sessions reaches the specified limit, the system will not create new sessions any more.

- Session timeout

Idle sessions will occupy connection resources if they are kept open for a long time. Administrators can set the timeout value of idle sessions to release resources in time.

For active sessions, the system also supports the setting of expiration time to control the duration of sessions.

- Session reuse

The session reuse function allows users to share one session if they use the same username. If any of the users closes the connection, all other users must log in again to continue their access.

16.3 Web SSO

Besides SAML SSO, the APV appliance also supports Web SSO. In an application environment that bears multiple systems, Web SSO also allows end users to access all application systems that trust each other after one-time login authentication. The APV appliance relies on the SAML, LDAP, RADIUS or OAuth authentication function to support Web SSO. In application scenarios where Web SSO is needed, the SAML, LDAP, RADIUS or OAuth authentication function must be deployed in the first place.

Note:

6. When Web SSO is deployed and implemented based on SAML authentication, the “username” parameter in the “**aaa samlsp idp attributes**” command must be set to “uid”. In addition, administrators must make sure the APV appliance can obtain user credentials from the IdP server. The AG series can function as an IdP server to cooperate with the APV appliance to support the Web SSO function. For more information, please contact Customer Support.
7. When Web SSO is deployed and implemented based on OAuth authentication, the “save_password” parameter in the “**aaa oauth registration**” command must be set to “save”.

On the APV appliance, Web SSO can be implemented based on the following methods:

- NT LAN Manager (NTLM) authentication method
- HTTP basic authentication method

- HTTP POST SSO rule

16.3.2NTLM Authentication Method

NTLM is an authentication protocol that can provide protection for username and password exchanged during the authentication process. It employs the “challenge-response” scheme. Without submitting the actual passwords, clients can prove their knowledge of the passwords in an indirect way, thus this authentication mode can prevent the passwords from being intercepted by others.

Note: In the NTLM authentication mode, the function of keeping persistent connections to real services must be enabled. Administrators can use the “**http serverpersist on**” command to enable it.

The process of NTLM-based SSO between the APV appliance and real services is as follows:

1. The client requests resources on a real service via the APV appliance.

The real service returns a 401 Unauthorized response, requesting the client to do NTLM authentication.

The APV appliance submits an NTLM negotiation message (Type-1 message) on behalf of the client to negotiate on information such as the subject to be authenticated and the required security services. Besides, the client will also inform the real service of the protocols and encryption strength it supports.

The real service returns a challenge message (Type-2 message), which contains the selected encryption strength, security service and a challenge.

The APV appliance uses the client’s password and the received challenge to generate a challenge and sends it to the real service via an authentication message (Type-3 message).

The real service calculates an authentication message in almost the same way and compares it to the received one. If they match, the real service will confirm that the client owns the correct password. The authentication is completed.

16.3.3HTTP Basic Authentication Method

In HTTP basic authentication, the server simply validates the login credential (username and password) for authentication. The SSO process between the APV appliance and real services via HTTP basic is as follows:

1. The client requests resources on a real service via the APV appliance.

The real service returns a 401 Unauthorized response, requesting the client to do HTTP basic authentication.

The APV appliance submits an authentication request on behalf of the client, which contains the client’s username and password.

The real service validates the client's login credential and returns the requested resources.

16.3.4HTTP POST Rule

HTTP POST SSO rules define where and how the system submits the login credentials. If the URL address requested by an end user matches the HTTP POST SSO rule, the APV appliance will start the SSO POST process.

16.3.5Configuration Example

This example illustrates how to configure Web SSO based on the SAML authentication function for the APV appliance.

- **Configuration Requirement**

Configure the APV appliance to support Web SSO based on HTTP POST SSO rules.

- **Configuration Steps**

1. Configure SAML SSO.

Except the configuration of the "**aaa samlsp idp attributes**" command, the steps for configuring SAML SSO are the same as section "19.1.1.1.4 Configuration Example". When configuring SAML SSO for Web SSO support, the administrator must set the "username" parameter in this command to "uid". Besides, the administrator can optionally set the "password" parameter. For example:

```
AN(config)#aaa samlsp idp attributes saml_sp uid password
```

Enable the Web SSO function.

```
AN(config)#aaa sso on
```

Configure an HTTP POST SSO rule.

```
AN(config)#aaa sso post httpvs abc.com.cn "http://abc.com.cn/picture" username password
```

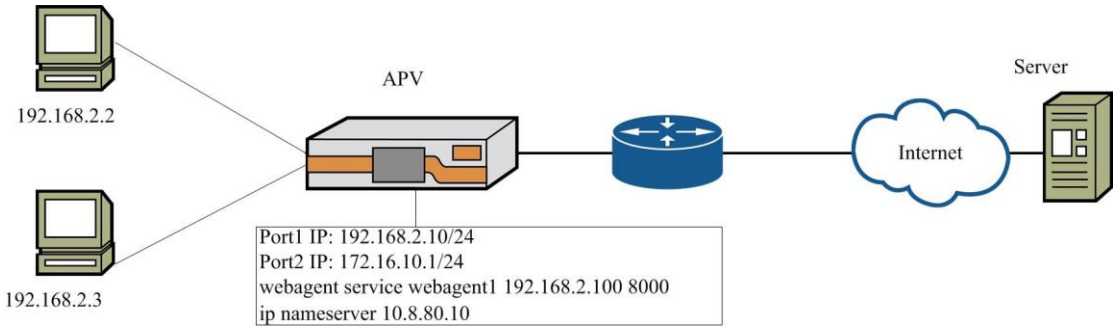
Based on the configurations above, if an end user accesses "http://abc.com.cn/picture", the APV appliance will submit the login credential on behalf of the client.

17Webagent

Webagent plays the role of a web proxy server. It is responsible for establishing connections to servers on the behalf of clients. On the APV appliance, the Webagent function proxies HTTP and HTTP requests from clients via the Webagent service. The Webagent service applies in both IPv4 and IPv6 environments. In addition, it supports the access control and DNS cache function.

17.2 Webagent Service

Administrators can configure a Webagent service and the DNS service on the APV appliance to implement its function as a proxy server. The following figure shows a basic configuration scenario for the Webagent service.



17.2.2.1.1.1.1 APV Functioning as a Webagent

17.3 Webagent Access Control

The APV appliance supports the following types of Webagent access control rules:

- Permits requests accessing specific host names
- Denies requests accessing specific host names

Webagent access controls can be applied to Webagent services only after being added to an access group. These types of access control rules can all be configured. The system supports a maximum of 1024 Webagent access control rules. By default, if a request hits both an access-permitted control rule and an access-denied control rule, the former takes effect.

Example:

1. Configure a Webagent service.

```
AN(config)#webagent service webagent1 192.168.2.100 8000
```

Configure an access-permitted control rule to allow the requests accessing “xyz*” and “abc*” to pass through.

```
AN(config)#webagent acl permit http host item_1 “xyz*”
AN(config)#web agent acl permit http host item_2 “abc*”
```

Configure an access group.

```
AN(config)#webagent acl group name group_1
```

Add “item_1” and “item_2” to “group_1”.

```
AN(config)#webagent acl group member group_1 item_1
AN(config)#webagent acl group member group_1 item_2
```

Apply the access group to the Webagent service.

```
AN(config)#webagent acl group apply webagent_1 group_1
```

17.4 DNS Cache

The system supports the cache of A and AAAA resource records for the Webagent service.

By default, the system provides dynamical cache of resource records returned by the DNS server. The cache time defaults to 60 minutes. Within the cache time, if a client queries this domain name again, the system will directly return the cached resource record without sending the query to the DNS server for resolution, and the cache time will also be renewed. The system supports a maximum of 256 dynamic domain names, each of which can have at most 8 A and 8 AAAA resource records. When the number of dynamically cached domain names reaches 256, if a new domain name needs to be cached, the system will replace the firstly cached domain name with the new one.

For frequently queried domain names, administrators can add them as static resource records by using the “**webagent dns host**” command. A static resource will never time out and can only be manually deleted after they are added to the cache. The system supports a maximum of 256 static DNS domain names, each of which can have at most 8 A and 8 AAAA resource records.

18Quality of Service (QoS)

18.2 Overview

This chapter introduces how to setup the QoS (Quality of Service) function on the APV 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.

18.3 Understanding QoS

QoS for networks is an industry-wide set of standards and mechanisms for ensuring high-quality performance for critical applications. By using QoS mechanisms, network administrators can use existing resources efficiently and ensure the required service level without reactively expanding or over-provisioning their networks.

QoS provides network administrators with the capacity of TCP, UDP and ICMP flow management by using queuing mechanism and packet filtering policies. By using queuing mechanism and filter rules, QoS supports both bandwidth management and priority control.

18.3.2Queuing Mechanism

The APV appliance has developed a queue-based QoS. Queue means a queue of network packet buffers. After the packet at the beginning of the queue has been processed, a new packet to be processed will be put at the end of the queue.

Each queue is bound with a particular network interface and controls either incoming or outgoing network traffic of that interface. QoS queues are organized in tree-like structures. On the top of a tree, a root queue is defined for either incoming or outgoing traffic of a network interface. Under the root queue, there can be multiple sub-queues. Sub-queues can also have their sub-queues. For each interface, at most two queue trees can be configured: one for the incoming traffic, and the other for the outgoing data.

Each queue is configured with bandwidth limit and priority for packet processing.

18.3.3Packet Filter Rule

A QoS filter is a rule which associates particular network traffic with a QoS queue.

In filter rule, the network traffic is specified by five parameters: source IP subnet, source port, destination IP subnet, destination port and protocol. By this association, administrators can deploy either application-oriented or link-oriented QoS control. Normally, application-oriented filter rules have TCP or UDP ports defined while link-oriented filter rules focus on source or destination IP addresses.

18.3.4Bandwidth 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 APV 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.

18.3.5Priority 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.

18.4 QoS Configuration

18.4.2Configuration Guidelines

18.4.2.1.1.1 General Settings of QoS

Operation	Command
Configure QoS interface	**qos interface** <*interface_name*> *[direction] [bandwidth]*
Define QoS queue	**qos queue root** <*queue_name*> <*interface_name*> *[direction] [bandwidth] [priority] [borrow] [default]* **qos queue sub** <*queue_name*> <*parent_queue*> *[bandwidth] [priority] [borrow] [default]*
Define QoS filter rules	**qos filter** <*filter_name*> <*queue_name*> < *src_addr*> <*smask*> <*sport*> <*dst_addr*> <*dmask*> <*dport*> <*proto*> *[priority]*
Enable QoS	**qos enable** <*interface_name*> *[direction]*

18.4.3Configuration Example via CLI

1. Define QoS interfaces.

```
AN(config)#qos interface port1 OUT 5Mb
AN(config)#qos interface port1 IN 5Mb
```

Define outgoing QoS queues.

```
AN(config)#qos queue root qr_oall port1 OUT 5Mb 3
AN(config)#qos queue sub qs_ossh qr_oall 2Mb 3 UNBORROW NONDEFAULT
AN(config)#qos queue sub qs_oftp qr_oall 512kb 2 UNBORROW NONDEFAULT
AN(config)#qos queue sub qs_odeflt qr_oall 8kb 3 UNBORROW DEFAULT
```

Default queue is for all the other packets which cannot hit any defined queues.

Define incoming QoS queues.

```
AN(config)#qos queue root qr_iall port1 IN 5Mb 3
AN(config)#qos queue sub qs_issh qr_iall 2Mb 3 BORROW NONDEFAULT
AN(config)#qos queue sub qs_iftp qr_iall 2Mb 2 BORROW NONDEFAULT
AN(config)#qos queue sub qs_ideflt qr_iall 8kb 3 BORROW DEFAULT
```

Define QoS filter rules.

```
AN(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
AN(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
AN(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
AN(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
```

Enable QoS.

```
AN(config)#qos enable port1 OUT
AN(config)#qos enable port1 IN
```

19Link Load Balancing (LLB)

19.2 Overview

This chapter details the configuration of the following Inbound and Outbound Link Load Balancing implementations:

- Single APV appliance and two ISPs
- Dual APV appliances and two ISPs

19.3 Understanding LLB

LLB (Link Load Balancing) allows TCP/IP network traffic to be load balanced through up to 128 upstream Internet Service Providers (ISPs). Load balancing can be performed on egress to the Internet (outbound LLB) or on ingress from the Internet (inbound LLB). LLB methods include rr (Round Robin), wrr (Weighted Round Robin), sr (Shortest Response), dd (Dynamic Detecting) , hi (Hash IP), hip (Hash IP+Port), lb (Least Bandwidth) and lc (Least Connections). LLB also includes ISP/link failure detection through default gateway and link health checking.

The APV appliance identifies links based on the logical port and peer MAC address. The statistics of LLB links are also collected based on the logical port and peer MAC address.

19.3.2Outbound LLB

Outbound LLB provides optimized outbound link utilization for environments that have more than one default gateway. In essence, it allows outbound traffic to be distributed among multiple upstream/ISP routers.

For example, let’s say you have Internet connectivity provided by two ISPs: ISP1 and ISP2. ISP1 assigns address range 100.1.1.0/24 so that you may use them on your network devices. ISP2 assigns address range 200.1.1.0/24 so that you may use them on your network devices. Outbound LLB allows you to load balance outbound connections traffic through ISP1 and ISP2. Connections forwarded through ISP1 are NATTed to an address from the range assigned by ISP1. Connections forwarded through ISP2 are NATTed to an address from the range assigned by ISP2. Thus, inbound responses for those connections will return through the ISP that they were originally sent through. If Internet connectivity through one of the ISP links is lost or interrupted, the outbound traffic will no longer be sent through that ISP. All traffic will be distributed to the functional ISP.

APV outbound LLB methods can work well on the data traffic based on TCP and UDP protocols, as well as the packets based on IP, IPsec or GRE protocols.

19.3.3Inbound LLB

Inbound LLB provides service resiliency for inbound clients. Hosted services are visible to external clients via a separate IP address on the address space assigned by each ISP.

To illustrate, let's use the same example ISPs as mentioned previously. All external clients trying to connect to the addresses assigned by ISP1 will be routed through ISP1's backbone. All external clients trying to connect to addresses assigned by ISP2 will be routed through ISP2's backbone. Inbound LLB allows you to advertise a device or Virtual IP (VIP) using two IP addresses: one from ISP1 and the other from ISP2. A DNS server on the APV 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.

19.3.4LLB Health Check

LLB Health Check is used to check whether the link between the APV interface and the upstream device is available. This can be accomplished by broadcasting ARP requests at regular intervals and pinging a user-defined upstream IP address. Besides, TCP-based and DNS-based health checks are also supported.

Broadcasting ARP requests at regular intervals can check the availability of the link between APV interface and the upstream ISP router. Pinging a user-defined upstream IP address not only can verify if the link between APV interface and the upstream ISP router is available, but also verify the link between upstream ISP router and user-defined upstream IP address. Multiple upstream IP addresses can be defined for reliable checking. If any of check point is pingable, the related link is usable. This ensures that the WAN link is up before forwarding traffic across that link.

19.3.5LLB Remote Site Accessibility Check

Apart from the LLB health check on the LLB links, the LLB Remote Site Accessibility Check function is designed to check whether a remote site is accessible to the APV appliance. A remote site stands for a remote network, and the administrator can specify a destination IP address in the remote network to be checked for the accessibility of the remote site.

ICMP- and TCP-based remote site accessibility checks are supported by this function.

This function works only for outbound traffic with the "rr" or "wrr" as outbound LLB method.

19.3.6LLB Methods

Outbound LLB supports the following load balancing methods:

- rr (Round Robin)
- wrr (Weighted Round Robin)
- sr (Shortest Response Time)
- dd (Dynamic Detecting)
- hi (Hash IP)

- hip (Hash IP+Port)
- lb (Least Bandwidth)
- lc (Least Connections)

Inbound LLB supports three load balancing methods:

- rr (Round Robin)
- wrr (Weighted Round Robin)
- proximity

Round Robin distributes each new session to gateways in an alternating (round robin) way. This is the default load balancing method.

Weighted Round Robin is similar to Round Robin except that a bias (or weight) may be assigned to each gateway so that some gateways may receive more sessions than others. This allows more traffic to be directed through an ISP with higher bandwidth capacity.

Shortest Response Time: The link with the shortest response time will get the next request. Calculation of shortest response time of a link is based on the initiation process of each TCP connection (both inbound and outbound connections). For the most accurate result, there should be enough TCP traffic instead of a few long existing TCP connections or only UDP traffic.

Note:

- If neither SLB traffic nor NAT traffic goes through the system, the LLB SR method cannot work properly.
- The “sr” method cannot be used to load balance IP fragments, non-TCP/UDP packets, and reassembled UDP packets.

Dynamic Detecting performs proximity calculations through all available ISP links to the destinations. By using parallel probe arithmetic, a request from the client will be sent to a destination by different ISP links 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, APV appliance will choose the best ISP connection, according to the results derived from these proximity calculations.

Hash IP distributes the outbound traffic among links in the way that the link with higher weight is routed with higher probability, by performing Hash operation on the IP address. When the chosen link is down, the system will carry another Hash operation on the links available. When HI is deployed as the LLB method, the IPflow function can be disabled.

Hash IP+Port distributes the outbound traffic among links in the way that the link with higher weight is routed with higher probability, by performing Hash operation on the IP address and port. When the chosen link is down, the system will carry another Hash operation on the links available. When HIP is deployed as the LLB method, the IPflow function can be disabled.

Least Bandwidth selects the link with the smallest bandwidth usage to forward outbound traffic. To use this method, administrators need to specify the maximum bandwidth allowed for each link through the "**llb link route**" command. If a link has no bandwidth limit configured, the lb method will not select this link. If the bandwidth limit is configured in none of the links, the wrr method will be used instead.

Least Connections selects the outbound link with the least number of connections to forward outbound traffic. The lc method is only recommended when all LLB links are configured with NAT and all LLB outbound traffic can be NATted. If all LLB links are not configured with NAT, the lc method actually functions as the wrr method. It is not recommended to use this method in other situations.

Proximity: The IP address of the nearest DNS server will be sent to the client as the response. When a DNS request arrives, APV will first search in the Eroute table reversely to find a proximity route matching the source address of the DNS request, and then give response to the client with the corresponding DNS server's IP address (A record) according to the Eroute gateway.

19.3.7Policy-based Routing (Eroute)

LLB policies provide the methods necessary to allow administrators to direct outbound traffic to a preferred route based on the IP address (source and destination) and service type (mail, FTP, Web, etc.).Policy based routing, unlike regular routing, allows the inclusion of the source IP, source port and destination port as well as the protocol into the route selection. For example, using routing policy can ensure that all the traffic generated by AOL instant messenger always uses the same link. If instant messenger client uses different destination IP addresses in its requests and these requests are sent through the different routes, this might confuse the server and cause login failure. Configuring routing policy will prevent this problem. The CLI command for that would be:

```
AN(config)#ip eroute aol_route 1500 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 5190 tcp gateway_ip 1
```

The APV appliance supports at most 5000 eroutes.

IP region

Eroute supports IP region. Administrators are allowed to import pre-defined IP region table via HTTP, FTP or Local File method, and then execute the command "**ipregion route**" to apply the imported IP region table. This will generate a large number of Eroute configurations, without making complex configurations. Administrators are also allowed to export the IP region table via FTP URL or Local File method.

APV appliance will check the contents of the file instead of the file type when an IP region file is imported. To ensure that the IP region file can be imported successfully, please pre-define the file contents strictly with the following items included in each entry:

- IP subnet (in CIDR format)
- Country name (optional, up to 7 bytes)

- Brief description (optional, up to 63 bytes)

These items must be separated with a “Tab”. For example:

```
27.8.0.0/13     CN  China Unicom Chongqing Province network
27.36.0.0/14    CN  China Unicom Guangdong province network
```

Note:

- By default, there are three predefined IP region tables including “predefined_cernet”, “predefined_cnc” and “predefined_ct”. It is recommended not to use the same name with the default predefined IP region tables.
- The routes and proximity rules configured for IP region exist as a whole in the system. Administrators cannot change or remove a single route or a rule.

19.3.8LLB 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.

19.3.9Route 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. APV appliance supports variant types of routing rules in which eroute priority is higher than priority of the default and static routes. Default routes will have priority 1 and static routes 101-132 depending on the netmask; i.e. the static route with 24-bit netmask will have priority 124 and with 32-bit netmask will have priority 132. The routes that correspond to the interfaces will have priority 2000. The routes created based on the traffic that come from the local subnet are called droutes (Direct Route) and will have priority 2000.

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

19.3.9.1.1.1 Route Priority

Name of Route	Priority
EROUTE-P	2001-2999
IROUTE, DROUTE	2000
RTS	1999
EROUTE-N	1001-1999
IPFLOW	1000-1999 (defaults to 1000)
STATIC ROUTE	101-132 (IPv4) 101-228 (IPv6)
DYNAMIC ROUTE	101-132 (IPv4) 101-228 (IPv6)

Name of Route	Priority
LLB LINK ROUTE	2
DEFAULT ROUTE	1

19.3.10 Link Bandwidth Management

For better link bandwidth management, the APV appliance allows administrators to set a threshold value for the LLB link bandwidth.

When performing link selection for the outbound traffic, the system considers not only the routing policies configured for links but also the load status of each link. That is, when the current link has reached the configured bandwidth threshold, the APV appliance will search for available links from matched routes according to the descending sequence of priorities. The APV appliance first searches for available links from routes with the same priority as the current link. If all available links reach their bandwidth thresholds, the APV appliance will search for available links from routes with lower priorities. If the gateways of all matched routes are down or reach the configured bandwidth thresholds, the APV appliance will still choose the current link to transmit traffic.

In addition, the APV appliance allows administrators to configure a priority for the LLB link bandwidth. If the priority of a matched route is higher than the LLB link bandwidth priority, the traffic will be directly forwarded through this route.

With the LLB bandwidth management function, you do not need to configure Eroutes with the same priorities for multiple links. This improves the efficiency and flexibility of link bandwidth configuration and management.

Note:

- If the traffic hits an RTS or IPflow route, the traffic will be directly forwarded through the relevant LLB link no matter whether the LLB link reaches the bandwidth threshold.
- If an Eroute has been configured with the source IP address, source mask, source port number, destination IP address, destination mask, and destination port number and these IP addresses and masks are set to 0.0.0.0 and port numbers are set to 0, the APV appliance will not search for available links from the matched routes whose priorities are lower than 1000.

19.3.11 DNS Proxy

The DNS proxy function provides link load balancing for outbound DNS queries. With DNS proxy function, DNS queries can be distributed “evenly” among DNS servers of the ISP links.

The DNS proxy function works as follows:

- When receiving a DNS query from an intranet client, the APV appliance will select a DNS server from all the DNS servers of ISP links according to the load balancing method, and change the destination IP address of the DNS query to the IP address of the DNS server.
- The APV appliance forwards the DNS query to the DNS server of the ISP link.

The DNS proxy function supports the following methods:

- rr: Distributes new DNS queries to the DNS servers of the ISP links in a round robin way.
- wrr: Assigns every DNS server a weight. The DNS server with the higher weight will receive more DNS queries.

Note: If the same "weight" is specified for every DNS server or no "weight" is specified, the rr method will be used to select the DNS server; otherwise, the wrr method will be used.

Configuration Example:

```
AN(config)# llb link route link1 100.10.1.1 1 0.0.0.0 0Mbps
AN(config)# llb link route link2 200.20.1.1 1 0.0.0.0 0Mbps

AN(config)#llb dnsproxy server server1 100.10.1.3 1
AN(config)#llb dnsproxy server server2 200.20.1.3 1
```

Besides, the DNS proxy function supports selecting the DNS server using the domain customization policy and DNS network policy for the outbound DNS query. The domain customization policy has a higher priority than the DNS network policy, and the DNS network policy has a higher priority than the load balancing method. Please refer to section 19.3.11.1 DNS Network Policy and section 19.3.11.2 Domain Customization Policy for details.

19.3.11.1 DNS Network Policy

The DNS network policy is used to select a DNS server for the DNS query whose source IP address belongs to a specific network segment. When the source IP address of a DNS query belongs to the network segment specified by a DNS network policy, the system will select the DNS server specified by this policy for this DNS query. If a DNS query matches multiple DNS network policies, the system will select the DNS network policy that has the longest subnet mask.

Configuration example:

```
AN(config)#llb dnsproxy network 10.0.0.0 255.0.0.0 dns_server1
AN(config)#llb dnsproxy network 10.0.1.0 255.255.255.0 dns_server2
```

After the above configuration is executed, the system will select the DNS server named "dns_server2" for the DNS query from the 10.0.1.3 source IP address.

19.3.11.2 Domain Customization Policy

The system supports three domain customization policies:

- Static domain policy: The system always selects the specified DNS server regularly for the DNS query of the specific domain name.
- Bypass domain policy: The system does not use the DNS proxy function for the DNS query of the specified domain name.
- Persistent domain policy: The system persistently selects the specified DNS server for the DNS queries of the specified domain based on the source IP address. When the DNS query matches the persistent policy of the domain name for the first time, the system selects the DNS server based on the DNS network policy (if it is configured and matched) or the load balancing method for the DNS query, and records the corresponding relation between the source IP address and the selected DNS server. When subsequent DNS queries from the same IP address hit this policy, the system will select the specified same DNS server persistently.

Configuration Example:

```
AN(config)#llb dnsproxy domain "*.edu.cn" edu_dns
```

After the above configuration is executed, the system will always select the DNS server named “edu_dns” for the DNS queries matching the “*.edu.cn” domain.

```
AN(config)#llb dnsproxy domain www.abc.com bypass
```

After the above configuration is executed, the system will not use the DNS proxy function for the DNS queries matching the “www.abc.com” domain.

```
AN(config)#llb dnsproxy domain www.abc.com persistent
```

After the above configuration is executed, the system will select the DNS server based on the persistent policy for the DNS queries matching the “www.abc.com” domain.

19.3.11.3 Link Bandwidth Check

The DNS proxy function also supports link bandwidth check. When the traffic of an LLB link reaches the peak, the DNS servers of this LLB link will no longer be selected until this LLB link has spare bandwidth. The system will select a DNS server from other available DNS servers configured for the DNS proxy function.

Configuration Example:

```
AN(config)#llb dnsproxy link link1 server1
```

19.3.11.4 Link Health Check

The DNS proxy function supports the link health check. The link health check allows the APV appliance to send the DNS query to the destination host to detect the health status of the specified link.

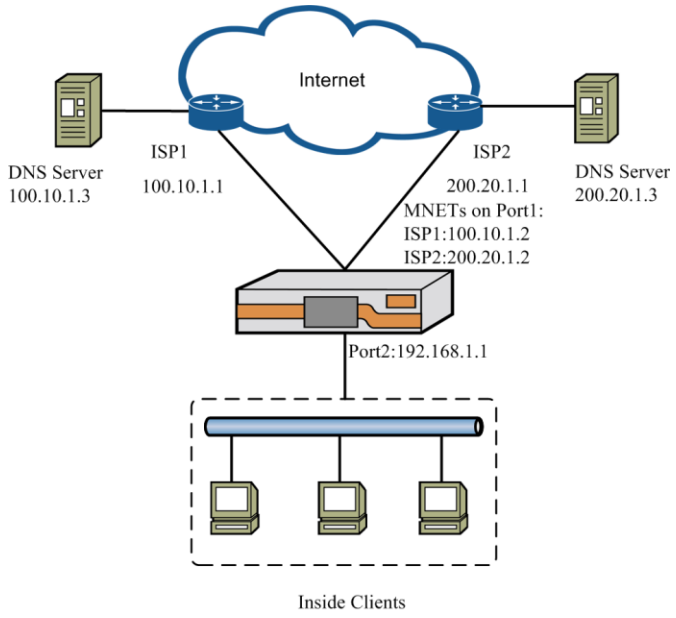
Configuration Example:

```
AN(config)#llb link route link1 100.10.1.1 1 0.0.0.0 0Mbps
AN(config)#llb dnsproxy server server1 100.10.1.3 1
AN(config)#llb dnsproxy link link1 server1
AN(config)#llb link health checker dns link1 100.10.1.3 www.test.com
AN(config)#llb link health on
```

19.3.11.5 Configuration Example

Configuration Objective:

The DNS queries from the intranet will be distributed “evenly” to link ISP1 and link ISP2.



19.3.11.5.1.1.1 DNS Proxy

- Configure the interface IP addresses for the LLB links ISP1 and ISP2.

```
AN(config)#mnet port1 ISP1
AN(config)#mnet port1 ISP2
AN(config)#ip address ISP1 100.10.1.2 255.255.255.0
AN(config)#ip address ISP2 200.20.1.2 255.255.255.0
```

- Configure the interface IP address for the interface connecting to the intranet.

```
AN(config)#ip address port2 192.168.1.1 255.255.255.0
```

- Configure the LLB link ISP1 and ISP2.

```
AN(config)#llb link route ISP1 100.10.1.1 1 0.0.0.0 0Mbps
AN(config)#llb link route ISP2 200.20.1.1 1 0.0.0.0 0Mbps
```

- Configure DNS servers for the DNS proxy function.

```
AN(config)#llb dnsproxy server server1 100.10.1.3 1
AN(config)#llb dnsproxy server server2 200.20.1.3 1
```

- Enable the DNS proxy function.

```
AN(config)#llb dnsproxy on
```

19.3.12 IPv6 Support for LLB

The APV appliance provides broad IPv6 support for the LLB module, of which the Eroute, inbound and outbound LLB, link health check and IP region can all work in the IPv6 network environment. For the Eroute, the source IP, destination IP, gateway IP and IP region can all be configured with the IPv6 addresses. However, please note that only IPv4 or only IPv6 addresses can be configured in one IP region table. For outbound LLB, only route-based LLB supports IPv6 configurations, while NAT-based LLB does not.

19.3.13 IP Address Group Support for LLB

The IP address group function allows administrators to add multiple address segments with the same outbound link to an IP address group. After a route is configured for the IP address group, the route takes effect for all IP address segments in the same IP address group.

When the IP address segment changes, the administrator only needs to manage the IP address group and does not need to care about the specific routing configuration, which simplifies the configuration and improves the usability.

The IP address group function allows administrators to associate IP address groups for Eroute. When an address segment in an IP address group is modified, the corresponding Eroute rule will take effect immediately. The system supports a maximum of 1024 IP address groups.

- For Eroutes associated with IP address groups, they are sorted by priority from high to low. If the priorities are the same, they are sorted according to the order of configuration. If the priorities and quintuple are the same, they are matched according to the LLB algorithm.
- If the Eroute of the associated IP address group and the Eroute of the unassociated address group have the same priority, the Eroute of the unassociated address group has the higher priority.
- The IP address group function allows administrators to associate IP address groups for source-based and destination-based NAT. When an IP address segment in an IP address group is modified, the corresponding NAT rule will take effect immediately.

- The matching order of NAT rules is: static NAT>destination-based NAT (unassociated IP address group)>source-based NAT (unassociated IP address group)>destination-based NAT (associated IP address group)>source-based NAT (associated with IP address group).
- For NAT rules associated with IP address groups, system sort them according to the order in which the IP address groups are associated.

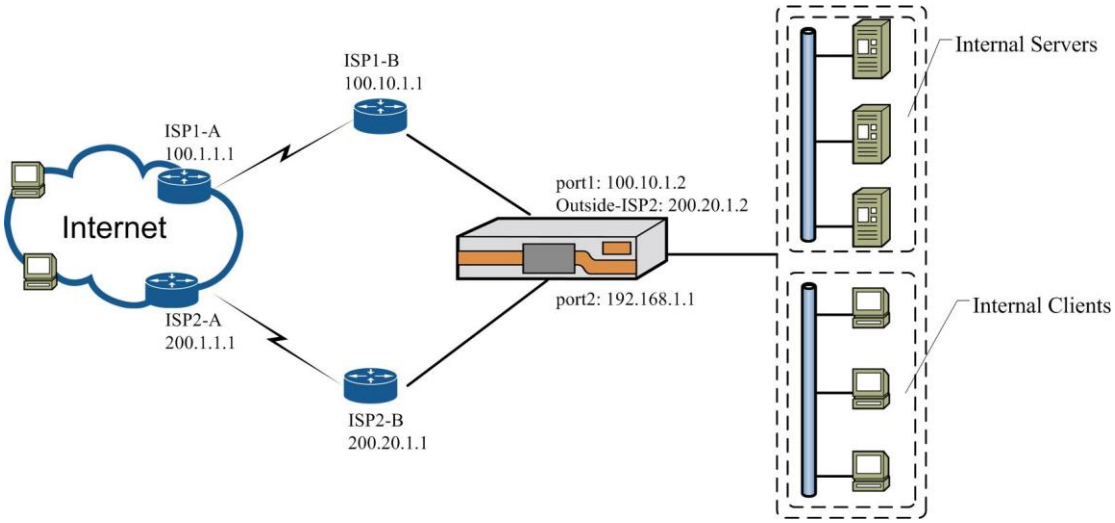
19.4 LLB Configuration

19.4.2Outbound LLB Configuration (One APV Appliance)

In this implementation example, one APV appliance will be configured to load balance outbound traffic through two ISPs.

If the single APV appliance stopped working, the network connectivity would be interrupted. Therefore, we recommend the implementation example with two APV appliances in section 19.4.3 Outbound LLB Configuration (Two APV Appliances).

19.4.2.1 Configuration Guidelines



19.4.2.1.1.1 Outbound LLB (One APV Appliance)

19.4.2.1.2 General Settings of Outbound LLB

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix} [overlap]*
Configure MNET	**mnet** *{system_ifname|bond_ifname|vlan_ifname} <user_interface_name>*

Configure LLB health check	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}** **llb link health checker icmp** *<link_name> <host> [hc_interval] [timeout] [hc_up] [hc_down]*
Configure LLB remote site accessibility check	**llb rsite checker** *<checker_name> {tcp|icmp} <port> [interval] [rsite_up] [rsite_down]* **llb rsite net** *<rsite_name> <network_ip> {netmask|prefix} <checker_name> [check_ip]* **llb rsite health {on|off}**
Configure outbound LLB method	**llb method outbound {rr|wrr|sr|hi|hip|lb|lc}** **llb method outbound dd** *[netmask] [prefix] [detecting_mode]*
Configure Eroutes and manage link bandwidth	**ip eroute** *<name> <priority> <srcip> {srcmask|prefix} <srcport> <dsthost> {dstmask|prefix} <dstport> <protocol> <gateway_ip> [weight]* **llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link bw_priority** *<priority>*
Configure NAT	**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*
Enable IPflow and RTS	**ip ipflow {on|off}** **ip rts off** **ip rts on** *[all|gateway]*

19.4.2.2 Configuration Example via the CLI

1. Configure interface IP addresses.

The Port1 interface IP will have an address from ISP1's address range. In order to assign an additional IP address on the Port1 interface, you must define and configure a multi-netted virtual interface (MNET). You will create an MNET named "outside_isp2" and assign it an IP address from ISP2's address range.

```
AN(config)#ip address port1 100.10.1.2 255.255.255.0
AN(config)#mnet port1 outside_isp2
AN(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

Then, configure the IP address of the Port2 interface.

```
AN(config)#ip address port2 192.168.1.1 255.255.255.0
```

Configure link.

Configure link ISP1:

```
AN(config)#llb link route ISP1 100.10.1.1
```

Configure link ISP2:

```
AN(config)#llb link route ISP2 200.20.1.1
```

Configure LLB health check (optional).

ISP link health checks are performed to ensure that the link between the local router (APV appliance) and the remote ISP router is up.

Configure link health check of ISP1.

```
AN(config)#llb link health checker icmp ISP1 “100.10.1.1” 10 5 3 3
```

Multiple health checks can be configured for the same link. Here, 100.1.1.2, 100.1.1.3 and 100.1.1.4 are another three remote routers of ISP1.

```
AN(config)#llb link health checker icmp ISP1 “100.1.1.2” 10 5 3 3
AN(config)#llb link health checker icmp ISP1 “100.1.1.3” 10 5 3 3
AN(config)#llb link health checker icmp ISP1 “100.1.1.4” 10 5 3 3
```

Only when all the health checks for ISP1 have failed, will the link ISP1 be deemed as down.

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*>”.

Configure link health check of ISP2.

```
AN(config)#llb link health checker icmp ISP2 “200.20.1.1” 10 5 3 3
```

Enter the following command to enable link health check:

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

Configure LLB remote site accessibility check (optional).

LLB remote site accessibility checks are performed to ensure that the remote site is accessible to the APV appliance.

Configure LLB remote site accessibility check.

```
AN(config)#llb rsite checker TCP1 tcp 80 15 3 3
```

The configured LLB remote site accessibility check can be associated with a remote site.

```
AN(config)#llb rsite net Beijing 10.10.1.0 255.255.255.0 TCP1 10.10.1.8
```

Enter the following command to enable LLB remote site accessibility check.

```
AN(config)#llb rsite health on
```

Configure outbound LLB method (optional).

The outbound LLB supports the following methods:

- Round Robin (rr)
- Weighted Round Robin (wrr)
- Shortest Response (sr)
- Dynamic Detecting (dd)
- Hash IP (hi)
- Hash IP+Port (hip)
- Least Bandwidth (lb)
- Least Connections (lc)

The default method is “rr”.

In this example, we use the “wrr” method.

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

Configure Eroutes and manage link bandwidth.

To make different traffic go through different links, configure the Eroutes for two LLB links.

```
AN(config)#ip eroute "er1" 1600 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
AN(config)#ip eroute "er2" 1400 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any 200.20.1.1 1
```

To make traffic that does not match the preceding Eroute configurations go through ISP1, configure the following Eroute:

```
AN(config)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
```

If necessary, update the LLB link bandwidth thresholds.

```
AN(config)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
AN(config)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
```

You can set a priority for the link bandwidth threshold to determine whether the configured link bandwidth threshold takes effect for the relevant LLB link.

```
AN(config)#llb link bw_priority 1500
```

Because the priority of Eroute “er1” is higher than the bandwidth priority, the gateway specified by the Eroute is not affected by the bandwidth threshold of ISP1. By comparison, the gateway specified by Eroute “er2” is affected by the bandwidth threshold of ISP2.

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:

```
AN(config)#nat port 100.10.1.10 192.168.1.0 255.255.255.0
```

NAT for ISP2:

```
AN(config)#nat port 200.20.1.10 192.168.1.0 255.255.255.0
```

Other required configuration.

Execute the following command to ensure that packets from the same connection will be directed to the same link by using the same NAT rule. By default, the IPflow function is disabled.

```
AN(config)#ip ipflow on
```

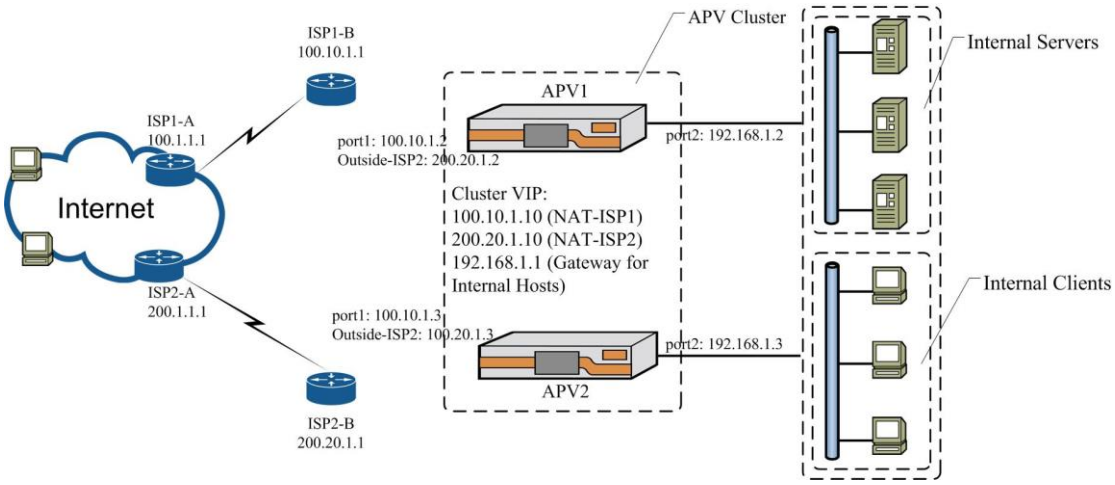
RTS (Return to Sender) should be turned on by executing the following command to ensure that a response packet (for example, ICMP response) will be directed to the link from which its corresponding request packet (for example, ICMP request) is sent. By default, the RTS function is disabled.

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

19.4.3Outbound LLB Configuration (Two APV Appliances)

In this implementation example, two APV appliances will be configured to load balance outbound traffic through two ISPs. This is the preferred implementation approach because the secondary APV appliance provides physical fault tolerance. If either APV appliance should fail, network connectivity will not be interrupted.

19.4.3.1 Configuration Guidelines



19.4.3.1.1.1.1 Outbound LLB (Two APV Appliances)

19.4.3.1.1.2 General Settings of Outbound LLB

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix} [overlap]*

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]*
Configure LLB health check	**llb link route** <*link_name*> <*route_ip*> *[weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}**
Configure LLB remote site accessibility check	**llb rsite checker** <*checker_name*> <*tcp|icmp*> <*port*> *[interval] [rsite_up] [rsite_down]* **llb rsite net** <*rsite_name*> <*network_ip*> *{netmask|prefix}* <*checker_name*> *[check_ip]* **llb rsite health {on|off}**
Configure cluster Virtual IPs for NATing traffic	**cluster virtual {on|off}** *[cluster_id|0] [interface_name]* **cluster virtual ifname** <*interface_name*> <*cluster_id*> **cluster virtual vip** <*interface_name*> <*cluster_id*> <*vip*> **cluster virtual priority** <*interface_name*> <*cluster_id*> <*priority*> *[synconfig_peer_name]*
Configure Eroutes and manage link bandwidth	**ip eroute** <*name*> <*priority*> <*srcip*> *{srcmask|prefix}* <*srcport*> <*dsthost*> *{dstmask|prefix}* <*dstport*> <*proto*> <*gatewayip*> *[weight]* **llb link route** <*link_name*> <*route_ip*> *[weight] [hc_srcip] [bandwidth_threshold]* **llb link bw_priority** <*priority*>
Configure NAT	**nat port** *{pool_name|vip}* <*source_ip*> *{netmask|prefix} [timeout] [gateway] [description]*
Enable IPflow and RTS	**ip ipflow {on|off}** **ip rts off** **ip rts on** *[all|gateway]*

19.4.3.2 Configuration Example via the CLI

Follow these steps to configure Outbound Link Load Balancing with clustered APV appliances. Due to the additional configuration required for a secondary APV 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.

1. Configure interface IP addresses.

You will need to define IP addresses on both APV appliances. The same MNET names may be used on both APV appliances.

(APV1) Port1 and Port2 IP address configuration:

```
APV1(config)#ip address port1 100.10.1.2 255.255.255.0
APV1(config)#mnet port1 outside_isp2
APV1(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
APV1(config)#ip address port2 192.168.1.2 255.255.255.0
```

(APV2) Port1 and Port2 IP address configuration:

```
APV2(config)#ip address port1 100.10.1.3 255.255.255.0
APV2(config)#mnet port1 outside_isp2
APV2(config)#ip address outside_isp2 200.20.1.3 255.255.255.0
APV2(config)#ip address port2 192.168.1.3 255.255.255.0
```

Configure a cluster virtual router for outbound traffic.

Outbound traffic (from behind the APV appliances) must be forwarded to an IP address on the APV appliances. To provide a fault tolerant gateway for inside devices, a virtual cluster VIP must be configured.

(APV1) Configure the first APV appliance as the master virtual router for all interfaces so it will process outbound traffic. Assign it a higher priority than the secondary APV appliance.

```
APV1(config)#cluster virtual ifname port2 1
APV1(config)#cluster virtual vip port2 1 192.168.1.1
APV1(config)#cluster virtual priority port2 1 200
APV1(config)#cluster virtual on 1 port2
```

(APV2) Configure the secondary APV appliance as a backup virtual router for all interfaces so it will not process outbound traffic unless the primary APV appliance fails. Assign it a lower priority than the primary APV appliance.

```
APV2(config)#cluster virtual ifname port2 1
APV2(config)#cluster virtual vip port2 1 192.168.1.1
APV2(config)#cluster virtual priority port2 1 100
APV2(config)#cluster virtual on 1 port2
```

Configure link.

Same settings should be configured in both APVs.

```
APV1(config)#llb link route ISP1 100.10.1.1
APV1(config)#llb link route ISP2 200.20.1.1

APV2(config)#llb link route ISP1 100.10.1.1 1
APV2(config)#llb link route ISP2 200.20.1.1 2
```

Health check can be configured to monitor the status of network, and turn on health check.

```
APV1(config)#llb link health checker icmp ISP1 “100.10.1.1” 10 5 3 3
APV1(config)#llb link health checker icmp ISP2 “200.20.1.1” 10 5 3 3
APV1(config)#llb link health on
```

```
APV2(config)#llb link health checker icmp ISP1 “100.10.1.1” 10 5 3 3
APV2(config)#llb link health checker icmp ISP2 “200.20.1.1” 10 5 3 3
APV2(config)#llb link health on
```

Remote site accessibility check can be configured to monitor the accessibility of remote sites.

```
APV1(config)#llb rsite checker TCP1 tcp 80 15 3 3
APV1(config)#llb rsite net Beijing 10.10.1.0 255.255.255.0 TCP1 10.10.1.8
APV1(config)#llb rsite health on

APV2(config)#llb rsite checker TCP1 tcp 80 15 3 3
APV2(config)#llb rsite net Beijing 10.10.1.0 255.255.255.0 TCP1 10.10.1.8
APV2(config)#llb rsite health on
```

Configure Eroutes and manage link bandwidth.

To make different traffic go through different links, configure the Eroutes for two LLB links.

```
AN(APV1)#ip eroute "er1" 1600 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
AN(APV1)#ip eroute "er2" 1400 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any 200.20.1.1 1
AN(APV2)#ip eroute "er1" 1600 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
AN(APV2)#ip eroute "er2" 1400 192.168.1.0 255.255.255.0 0 0.0.0.0 0.0.0.0 0 any 200.20.1.1 1
```

To make traffic that does not match the preceding Eroute configurations go through ISP1, configure the following Eroute:

```
AN(APV1)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
AN(APV2)#ip eroute "er3" 1001 0.0.0.0 0.0.0.0 0 0.0.0.0 0.0.0.0 0 any 100.10.1.1 1
```

If necessary, update the LLB link bandwidth thresholds.

```
AN(APV1)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
AN(APV1)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
AN(APV2)#llb link route ISP1 100.10.1.1 1 100.10.1.2 500Mbps
AN(APV2)#llb link route ISP2 200.20.1.1 2 200.20.1.2 300Mbps
```

You can set a priority for the link bandwidth threshold to determine whether the configured link bandwidth threshold takes effect for the relevant LLB link.

```
AN(APV1)#llb link bw_priority 1500
AN(APV2)#llb link bw_priority 1500
```

Because the priority of Eroute “er1” is higher than the bandwidth priority, the gateway specified by the Eroute is not affected by the bandwidth threshold of ISP1. By comparison, the gateway specified by Eroute “er2” is affected by the bandwidth threshold of ISP2.

Configure cluster Virtual IPs for NATing traffic.

(APV1) Cluster VIPs for NAT on each ISP. Assign a higher priority than the secondary APV appliance.

```
APV1(config)#cluster virtual ifname port1 1
APV1(config)#cluster virtual vip port1 1 100.10.1.10
APV1(config)#cluster virtual prio port1 1 200
APV1(config)#cluster virtual on 1 port1
APV1(config)#cluster virtual ifname outside-isp2 1
APV1(config)#cluster virtual vip outside-isp2 1 200.20.1.10
APV1(config)#cluster virtual prio outside-isp2 1 200
APV1(config)#cluster virtual on 1 outside-isp2
```

(APV2) Cluster VIPs for NAT on each ISP. Assign them a lower priority than the primary APV appliance.

```
APV2(config)#cluster virtual ifname port1 1
APV2(config)#cluster virtual vip port1 1 100.10.1.10
APV2(config)#cluster virtual prio port1 1 100
APV2(config)#cluster virtual on 1 port1
APV2(config)#cluster virtual ifname outside-isp2 1
APV2(config)#cluster virtual vip outside-isp2 1 200.20.1.10
APV2(config)#cluster virtual prio outside-isp2 1 100
APV2(config)#cluster virtual on 1 outside-isp2
```

Configure NAT for outbound LLB sessions.

(Both APVs) NAT rules for ISP1 and ISP2:

```
APV1(config)#nat port 100.10.1.10 192.168.1.0 255.255.255.0
APV1(config)#nat port 200.20.1.10 192.168.1.0 255.255.255.0
APV2(config)#nat port 100.10.1.10 192.168.1.0 255.255.255.0
APV2(config)#nat port 200.20.1.10 192.168.1.0 255.255.255.0
```

Other required configuration.

Execute the following command to ensure that packets from the same connection will be directed to the same link by using the same NAT rule. By default, the IPflow function is disabled.

```
AN(config)#ip ipflow on
```

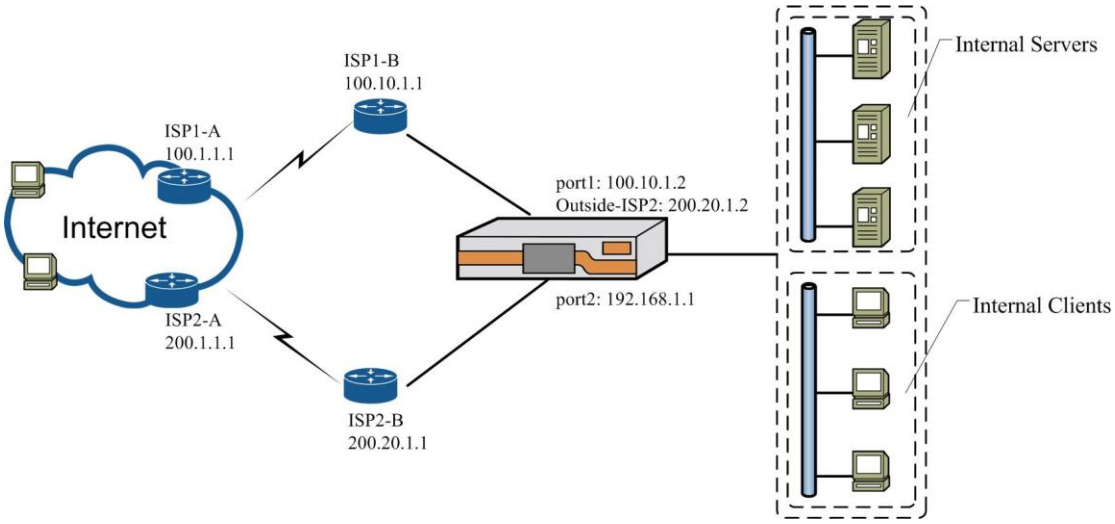
RTS (Return to Sender) should be turned on by executing the following command to ensure that a response packet (for example, ICMP response) will be directed to the link from which its corresponding request packet (for example, ICMP request) is sent. By default, the RTS function is disabled.

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

19.4.4Inbound LLB Configuration

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

19.4.4.1 Configuration Guidelines



19.4.4.1.1.1 Inbound LLB

19.4.4.1.2 General Settings of Inbound LLB

Operation	Command
Configure interface IP address	**ip address** *{system_ifname|mnet_ifname|vlan_ifname|bond_ifname} <ip_address> {netmask|prefix} [overlap]*
Configure MNET	**mnet** *{system_ifname|bond_ifname} <user_interface_name>*
Configure LLB health check	**llb link route** *<link_name> <route_ip> [weight] [hc_srcip] [bandwidth_threshold]* **llb link health {on|off}**
Configure SLB	**slb real http** *<real_service> <ip> [port] [max_connection] [hc_type] [hc_up] [hc_down]* **slb virtual http** *<virtual_service> <vip> [vport] [arp_support] [max_connection]* **slb policy static** *<virtual_service> <real_service>*
Configure LLB DNS host and TTL	**llb dns host** *<host_name> <ip> [weight] [port] [link_name]* **llb dns ttl** *<host_name> [seconds]*
Configure load balancing method	**llb method inbound {rr|wrr|proximity}**
Enable IPflow and RTS	**ip ipflow {on|off}** **ip rts off** **ip rts on** *[all|gateway]*

19.4.4.2 Configuration Example via the CLI

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

1. Configure interface IP addresses.

The Port1 interface IP will have an address from ISP1's address range. In order to assign an additional IP address on the Port1 interface, you must define and configure a multi-netted virtual interface (MNET). You will create an MNET named "outside_isp2" and assign it an IP address from ISP2's address range.

```
AN(config)#ip address port1 100.10.1.2 255.255.255.0
AN(config)#mnet port1 outside_isp2
AN(config)#ip address outside_isp2 200.20.1.2 255.255.255.0
```

Then, configure the IP address of the Port2 interface.

```
AN(config)#ip address port2 192.168.1.1 255.255.255.0
```

Configure link.

Configure link ISP1:

```
AN(config)#llb link route ISP1 100.10.1.1
```

Configure link ISP2:

```
AN(config)#llb link route ISP2 200.20.1.1
```

Health check can also be configured to monitor the status of network. ISP link health checks are performed to ensure that the link between the local router (APV appliance) and the remote ISP router is up.

Configure the health check of link ISP1.

```
AN(config)#llb link health checker icmp ISP1 "100.10.1.1" 10 5 3 3
```

Configure the health check of link ISP2.

```
AN(config)#llb link health checker icmp ISP2 "200.20.1.1" 10 5 3 3
```

Enter the following command to enable link health check:

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

Configure Server Load Balance.

```
AN(config)#slb virtual http vip1 100.10.1.10
AN(config)#slb virtual http vip2 200.20.1.10
AN(config)#slb real http server1 192.168.1.100
AN(config)#slb policy static vip1 server1
AN(config)#slb policy static vip2 server1
```

Configure LLB DNS host and TTL for inbound.

```
AN(config)#llb dns host llb.abc.com 100.10.1.10 2
AN(config)#llb dns host llb.abc.com 200.20.1.10 1
```

```
AN(config)#llb dns ttl llb.abc.com 60
```

Configure inbound load balancing method.

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

Note: To use the “proximity” method for inbound load balancing, please first make configurations about “ip eroute”.

Other required configuration.

Execute the following command to ensure that packets from the same connection will be directed to the same link by using the same NAT rule. By default, the IPflow function is disabled.

```
AN(config)#ip ipflow on
```

RTS should be turned on by executing the following command to ensure that a response packet (for example, ICMP response) will be directed to the link from which its corresponding request packet (for example, ICMP request) is sent. By default, the RTS function is disabled.

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

20Global Server Load Balancing (GSLB)

20.2 Overview

Global Server Load Balancing (GSLB) distributes network traffic among servers providing the same service and deployed in different locations by returning different DNS resolution results. The APV appliance adopts the Smart DNS (SDNS) technology to implement GSLB. SDNS makes smart decisions at DNS resolution to achieve the following goals:

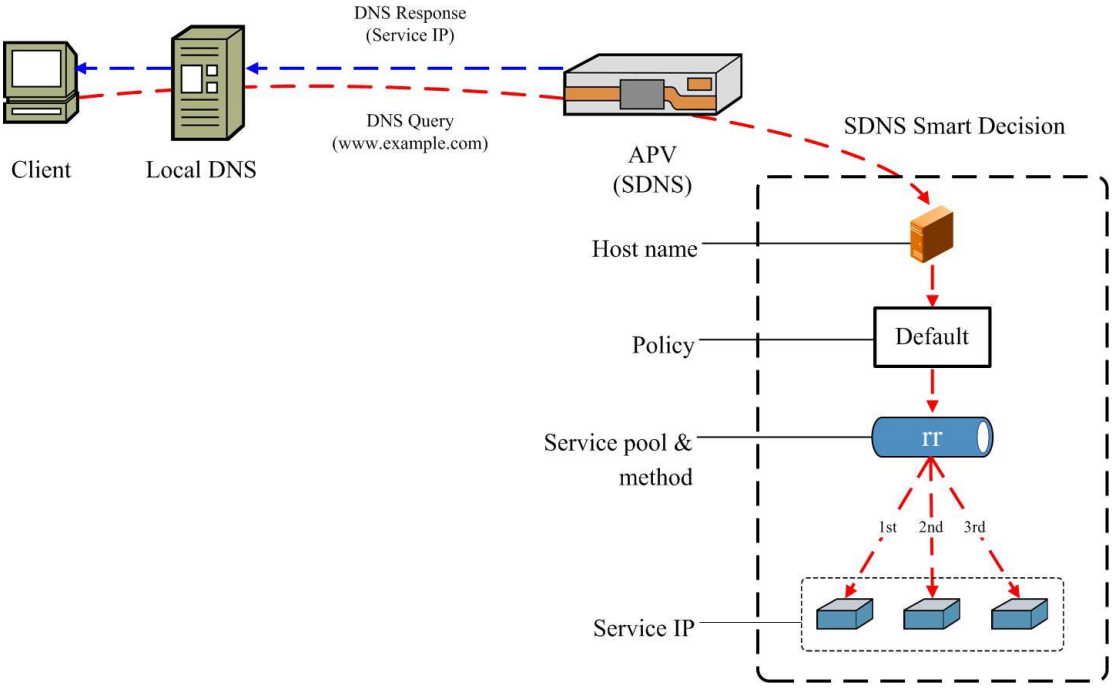
- High access efficiency: Based on the geographical location and network proximity of the user's local DNS, SDNS returns the IP address of the server that is "logically" nearest to the user or has the shortest response time, which improves the user's service access efficiency.
- Disaster tolerance: When detecting that a server is faulty, SDNS will pick up the IP addresses of other available servers to respond to DNS queries. This ensures that users can still access the service normally when a server is faulty and therefore enhances the service continuity and availability. When the faulty server restores, SDNS can also switch the traffic back to it.
- Load balancing: SDNS can distribute traffic among multiple available servers.

The following sections will describe the function principles and configuration examples of SDNS.

20.3 Function Principles

20.3.2SDNS Domain Name Resolution

SDNS provides smart and authoritative resolution for domain names. The following figure shows the process of domain name resolution provided by SDNS.



20.3.2.1.1.1.1 SDNS Domain Name Resolution

The process of domain name resolution provided by SDNS is detailed as follows:

1. The client sends the DNS query to the local DNS to resolve the domain name "www.example.com".
2. After a few rounds of lookups, the local DNS finds that SDNS is the authoritative DNS for the domain name and forwards the DNS query to SDNS.
3. SDNS makes smart decision to pick up healthy service IPs and return them to the local DNS.
4. The local DNS returns the service IPs to the client.

The process of SDNS smart decision is detailed as follows:

1. Based on the domain name in the DNS query, SDNS finds the hit SDNS policy.
2. Based on the SDNS policy, SDNS finds the hit SDNS service pool.
3. Based on the method of the hit SDNS service pool, SDNS picks up service IPs.
4. SDNS returns the picked service IPs to the local DNS.

20.3.3Basic Concepts

20.3.3.1 SDNS Host Name

The SDNS host name is the basic unit of the domain name resolution provided by SDNS. In SDNS, the domain names that can be resolved are defined as SDNS host names. The system supports a maximum of 16,000 SDNS host names.

For example:

```
AN(config)#sdns host name "www.xyz.com" 60
AN(config)#sdns host name "www.abc.com" 60
```

20.3.3.2 SDNS Listening IP Address

By default, the system provides the SDNS service on all IP addresses of the appliances, including the virtual IP address, interface IP address and HA floating IP address. It processes DNS queries received on these IP addresses with the destination port number 53.

Administrators can use the "**sdns listener**" command to configure SDNS listening IP addresses, so that the system will provide the SDNS service on only these listening IP addresses. After SDNS listening IP addresses are configured, a DNS query will be processed only when its destination IP address matches any of these listening IP addresses. Otherwise, it will be refused.

20.3.3.3 SDNS Service IP and Service Pool

In SDNS, one SDNS host name can be resolved into multiple IP addresses. SDNS allows the administrator to add these resolved IP addresses in to a pool, which is called a “service pool”. The IP address included in the service pool is called a “service IP”.

The service IP can be an IPv4 or IPv6 address and the service IPs in a service pool can be either all IPv4 addresses or IPv6 addresses. Note that a service pool cannot contain both IPv4 and IPv6 addresses.

For example:

```
AN(config)#sdns pool name "pool1" 1
AN(config)#sdns pool name "pool2" 1
AN(config)#sdns service ip "service1" 10.8.6.201 0
AN(config)#sdns service ip "service2" 10.8.6.202 0
AN(config)#sdns service ip "service3" 2012:1086::203 0
AN(config)#sdns service ip "service4" 2012:1086::204 0
AN(config)#sdns pool service "pool1" "service1"
AN(config)#sdns pool service "pool1" "service2"
AN(config)#sdns pool service "pool2" "service3"
AN(config)#sdns pool service "pool2" "service4"
```

20.3.3.3.1 SDNS Service Pool Method

The SDNS service pool method determines which service IPs will be picked up from the service pool to return to the local DNS. Service IPs picked up from the same service pool vary with service pool methods. SDNS supports the following service pool methods:

- Round Robin (rr)
- Weighted Round Robin (wrr)
- IP Overflow (ipo)
- Hash IP (hi)
- Simple Network Management Protocol (snmp)
- drop

The following table describes the meanings of SDNS service pool methods.

20.3.3.3.1.1 SDNS Service Pool Method

Method	Meaning
rr	If the service pool contains three service IPs, the maximum_rr_count is 1, and the rr method is used, the service IPs will be returned in the DNS responses in the sequence of “1, 2, 3, 1, 2, 3…”.
wrr	The wrr method is similar to the rr method. The difference is that every

Method	Meaning
	service IP in the service pool is assigned a weight. The second service IP will not be returned in the DNS responses until the number of times that the first service IP is returned reaches the weight value.
ipo	If the ipo method is used, the service IP with the highest priority will always be returned in the DNS responses.
hi	If the hi method is used, SDNS will compute the hash value based on the source IP address. For DNS queries with the same source IP address, SDNS returns the same service IP.
snmp	If the snmp method is used, SDNS will collect the data of specified OIDs from every service IP through the SNMP protocol, compute the metric of every service IP using the specified SNMP expression, and sort up service IPs in the specified sorting mode. The service IP in the in the first order will be returned in the DNS response. For details, refer to section 20.3.7.2 SNMP Data Collection.
drop	If the drop method is used, SDNS will directly discard the DNS query and return no DNS response.

Note: The SDNS service pool methods are targeted at single thread environment. Therefore, in a multi-threaded environment, the number of service IP hits of the rr method and the wrr method may deviate. For the rr method, in a single thread environment, the difference in the number of hits of any two available service IPs in the service pool is always not more than once, and in a multi-threaded environment, the number of hits of any two available service IPs in the service pool will not exceed the number of threads.

Besides, SDNS supports configuring two levels of service pool methods: primary and secondary. When failing to use the primary method to pick up any available service IP from the service pool, SDNS will use the secondary method. Therefore, the administrator must configure the primary method and can choose to configure the secondary method.

For example:

```
AN(config)#sdns pool method primary "pool1" "ipo"
AN(config)#sdns pool member priority "pool1" "service1" 1
AN(config)#sdns pool member priority "pool1" "service2" 2
AN(config)#sdns pool method secondary "pool1" "rr"

AN(config)#sdns pool method primary "pool2" "wrr"
AN(config)#sdns pool member weight "pool2" "service3" 3
AN(config)#sdns pool member weight "pool2" "service4" 2
AN(config)#sdns pool method secondary "pool2" "drop"
```

20.3.3.3.2 Auto Failover

SDNS provides the Auto Failover function for the SDNS service pool using the ipo method. When the service IP(s) with the highest priority become(s) unavailable, the Auto Failover function will

automatically switch the currently selected service IP(s) to the service IP(s) with the second highest priority in the same SDNS service pool. By default, this function is enabled.

For example:

```
AN(config)#sdns pool failover "pool1" on
```

20.3.3.3.3 Preemption

SDNS provides the Preemption function for the SDNS service pool using the ipo method. When the service IP(s) with the highest priority become(s) available again, the Preemption function will automatically switch the currently selected service IP(s) in the same SDNS service pool to the service IP(s) with the highest priority. By default, this function is enabled.

For example:

```
AN(config)#sdns pool preempt "pool1" on
```

20.3.3.3.4 Manual Switchover

SDNS provides the Manual Switchover function for the SDNS service pool using the ipo method. This function allows the administrator to manually switch the currently selected service IP(s) to the service IP(s) with the specified priority in the same SDNS service pool.

For example:

```
AN(config)#sdns pool ipo reset "pool1" 2
```

Note:

- If both the Auto Failover and Preemption functions are enabled, the manual switchover operation will not work.
- The manual switchover operation cannot be saved into memory. Therefore, the manual switchover operation will stop working after system reboot.
- When all the service IPs with the specified priority become down, DNS resolution will fail.

20.3.3.4 SDNS Policy

The SDNS policy determines the SDNS service pool for the host name. SDNS supports three types of policies:

- Region policy: determines the SDNS service pool based on the SDNS host name and the SDNS region into which the DNS query is divided. SDNS allows configuring multiple region policies for the same host name.
- Default policy: determines the SDNS service pool based on the SDNS host name. SDNS will use the default policy only when failing to pick up any available service IP from the service pool by using the hit region policy.

- Emergency policy: determines the SDNS service pool based on the host name. SDNS will use the emergency policy only when failing to pick up any available service IP from the service pool by using the region policy and the default policy. When the emergency policy is used, IPs from the associated service pool will be returned regardless of their health status.

The process of hitting SDNS policies is as follows:

a. When the host name in the DNS query matches multiple region policies, SDNS will first pick up available service IP(s) from the service pool associated with the region policy with the highest priority.

b. If SDNS fails to pick up any available service IP from this service pool, it will pick up available service IP(s) from the service pool associated with the default policy.

c. If SDNS fails to pick up any available service IP from this service pool, it will pick up available service IP(s) from the service pool associated with the emergency policy.

d. If SDNS still fails to pick up any available service IP, it will return an empty DNS response to the local DNS.

For details on SDNS region policy, please refer to section 20.3.6 SDNS Region Policy.

20.3.4SDNS CNAME Pool

In SDNS, the host name can also resolved into the Canonical Name (CNAME) record. An SDNS CNAME pool contains only one CNAME record. When SDNS receives a DNS CNAME query and the CNAME pool is hit, SDNS will return the CNAME record to the local DNS. When SDNS receives a DNS A/AAAA query and the CNAME pool is hit, SDNS first finds the CNAME record, then performs resolution on the CNAME, and finally returns the resolved IP addresses to the local DNS.

For example:

```
AN(config)#sdns service ip "web1" 10.8.6.10
AN(config)#sdns service ip "web2" 10.8.6.20
AN(config)#sdns pool name "pool_web" 1
AN(config)#sdns pool method primary "pool_web" rr
AN(config)#sdns pool service "pool_web" "web1"
AN(config)#sdns pool service "pool_web" "web2"
AN(config)#sdns pool cname "pool_cname" "www.web.example.com"
AN(config)#sdns host name "www.example.com" 60
AN(config)#sdns host name "www.web.example.com" 60
AN(config)#sdns policy default "www.example.com" "pool_cname"
AN(config)#sdns policy default "www.web.example.com" "pool_web"
```

After the preceding configurations are completed:

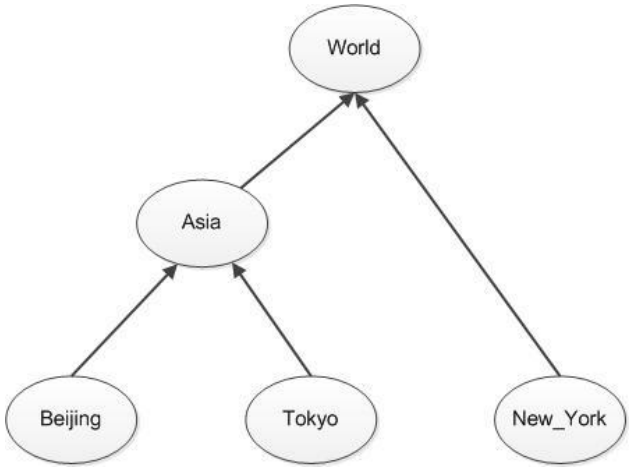
- When receiving the DNS CNAME query for the domain name “www.example.com”, SDNS returns the CNAME “www.web.example.com” to the local DNS.

- When receiving the DNS A/AAAA query for the domain name “www.example.com”, SDNS first finds the CNAME record of “www.example.com”, then performs resolution on the CNAME “www.web.example.com”, and finally returns the service IP “web1” or “web2” to the local DNS.

20.3.5SDNS Service Pool Fallback

If no available service IP can be picked up from an SDNS service pool based on the pool method, SDNS can switch the DNS queries of this service pool to another SDNS pool, which provides a failover mechanism for the SDNS service pool. The second SDNS pool is called a fallback pool.

One SDNS service pool has only one fallback pool but can be the fallback pools of multiple other service pools. Multiple SDNS service pools can form a tree structure of fallback, as shown in the following figure.



20.3.5.1.1.1.1 Service Pool Fallback (Tree Structure)

For example:

```
AN(config)#sdns pool name "Beijing" 1
AN(config)#sdns pool name "Tokyo" 1
AN(config)#sdns pool name "New_York" 1
AN(config)#sdns pool name "Asia" 1
AN(config)#sdns pool name "World" 1
AN(config)#sdns pool fallback "Beijing" "Asia"
AN(config)#sdns pool fallback "Tokyo" "Asia"
AN(config)#sdns pool fallback "New_York" "World"
AN(config)#sdns pool fallback "Asia" "World"
```

After the preceding configurations are completed:

- The service pools “Beijing” and “Tokyo” can fall back to the service pool “Asia” when they are unavailable.

- The service pool “New_York” can fall back to the service pool “World” when the service pool “New_York” is unavailable.
- The service pool “Asia” can fall back to the service pool “World” when the service pool “Asia” is unavailable.

Note:

- SDNS disallows multiple SDNS service pools to form a loop structure of fallback.
- An SDNS service pool can fall back to an SDNS CNAME pool, but an SDNS CNAME pool cannot fall back to an SDNS service pool.

20.3.6SDNS Region Policy

SDNS logically divides DNS queries coming from various locations into different regions, which are called SDNS regions. The administrator can divide SDNS regions based on geographical localities, such as Unites States, Japan and China. The administrator can also divide SDNS regions based on actual requirements, such as based on the Network Service Provider (ISP).

For an SDNS host name, the administrator can configure only one default policy and multiple region policies. These region policies associate DNS queries of different SDNS regions with different SDNS service pools.

The region policy determines the SDNS service pool based on the SDNS host name and the SDNS region into which the DNS query is divided. In multiple region policies configured for the same SDNS host name, the SDNS service pools vary with the SDNS regions. SDNS uses the SDNS proximity rules to determine the SDNS regions into which DNS queries coming from a specified subnet are divided. When receiving DNS queries, SDNS uses the longest prefix matching rule to find the hit proximity rules and therefore determines the SDNS regions into which the DNS queries are divided. When a DNS query matched multiple region policies, SDNS determines the hit region policy based on the priority. The DNS query is processed by the SDNS service pool associated with the region policy with the highest priority.

SDNS allows the administrator to configure static proximity rules to associate specified subnets with SDNS regions, and to associate the ipregion table with the SDNS region to generate proximity rules in batch. The proximity rules generated using the ipregion table are also called ipregion proximity rules.

In addition, SDNS supports using the Dynamic Proximity System (DPS) to generate dynamic proximity rules. For details on DPS, please refer to section 20.3.8 SDNS Dynamic Proximity System (DPS).

For example:

```
AN(config)#sdns host name "www.xyz.com" 1
AN(config)#sdns pool name "pool_beijing" 1
AN(config)#sdns pool name "pool_tokyo" 1
AN(config)#sdns pool name "pool_newyork" 1
```

```
AN(config)#sdns region name "Beijing"
AN(config)#sdns region name "Tokyo"
AN(config)#sdns region name "New_York"
AN(config)#sdns proximity 211.100.0.0 255.255.0.0 "Beijing"
AN(config)#sdns proximity 210.10.0.0 255.255.0.0 "Tokyo"
AN(config)#sdns proximity 216.100.0.0 255.255.0.0 "New_York"
AN(config)#sdns policy region "policy_beijing" "www.xyz.com" "pool_beijing" "Beijing" 0
AN(config)#sdns policy region "policy_tokyo" "www.xyz.com" "pool_tokyo" "Tokyo" 0
AN(config)#sdns policy region "policy_newyork" "www.xyz.com" "pool_newyork"
"New_York" 0
```

After the preceding configurations are completed:

- The DNS queries for the host name “www.xyz.com” coming from the subnet 211.100.0.0/16 will hit the region policy “policy_beijing”.
- The DNS queries for the host name “www.xyz.com” coming from the subnet 210.10.0.0/16 will hit the region policy “policy_tokyo”.
- The DNS queries for the host name “www.xyz.com” coming from the subnet 216.100.0.0/16 will hit the region policy “policy_newyork”.

Note: The system now support the edns-client-subnet (ECS) option, which carries information about the network that originated a DNS query and the network for which the subsequent response can be cached. With ECS support, the system can get the originator’s real address and assign their requests to the optimal servers. Combined with SDNS region policies, ECS helps provide better resolution services and improve user experience.

20.3.7SDNS Monitor

SDNS monitor can be used to perform health check on SDNS services and collect SNMP data from SDNS services. To use SDNS monitor, the administrator must first define the health check template or data collection template and then apply the template to an SDNS service or SDNS service pool to generate the monitor instance(s). The system will execute health check or data collection based on the monitor instances.

SDNS monitor supports the ICMP-type, TCP-type HTTP-type and HTTPS-type health check templates and the SNMP-type data collection template. A maximum of 200 health check and data collection templates are supported.

When a template is applied to an SDNS service, the system will generate a monitor instance for the SDNS service; while the template is applied to an SDNS service pool, the system will generate a monitor instance for every SDNS service in the service pool. The monitor instance generated based on the health check template is called the health check instance, and the monitor instance generated based on the SNMP data collection template is called the SNMP data collection instance. A maximum of 5000 monitor instances are supported and a maximum of 1024 SNMP data collection instances are supported.

20.3.7.1 SDNS Health Check

SDNS provides the health check function to check the health status of the service IPs in the SDNS service pool. When a service IP becomes unavailable, SDNS will pick up other available service IPs to respond to DNS queries. This ensures that service IPs returned to the local DNS is healthy and available and therefore enhances the service continuity and availability.

20.3.7.1.1 Health Check Type

SDNS supports the following types of health checks:

- **ICMP-type health check**

The ICMP-type health check determines the availability of a service IP by sending the ICMP Echo Request (ping) to the service IP. If receiving the ICMP Echo Response from the service IP, SDNS regards this service IP as available (Up); otherwise, SDNS regards this this service IP as unavailable (Down).

- **TCP-type health check**

The TCP-type health check determines the availability of a service IP by establishing the TCP connection with the service IP. If the TCP connection is established successfully, SDNS regards this service IP as available (Up); otherwise, SDNS regards this this service IP as unavailable (Down).

- **HTTP-type health check**

The HTTP-type health check determines the availability of a service IP by sending the HTTP request to the service IP. If the service IP returns the expected response, SDNS determines the health result based on the result flag. If the result flag is “up”, SDNS regards this service IP as available (Up); if the result flag is “down”, SDNS regards this service IP as unavailable (Down).

- **HTTPS-type health check**

The HTTPS-type health check determines the availability of a service IP by implementing SSL handshake with the service IP, and then sending an HTTP request to this service IP afterwards. If the SSL handshake is successful and the service IP returns the expected response, SDNS determines the health check result based on the result flag. If the result flag is “up”, SDNS regards this service IP as available (Up); if the result flag is “down”, SDNS regards this service IP as unavailable (Down).

20.3.7.1.2 Health Check Template and Instance

To use the SDNS health check function, the administrator needs to first define health check templates and then apply health check templates to service IPs or service pools to generate health check instances.

The health check template is comprised of the following elements:

- **Health check type**: can be ICMP, TCP, HTTP, or HTTPS.

- **Request line**: specifies the predefined HTTP request content, which includes HTTP method, URL path and HTTP version. This parameter is available only for the HTTP-type and HTTPS-type health check templates. The default value is “HEAD / HTTP/1.0\r\n\r\n”.
- **Expected response**: specifies the expected HTTP response(s). This parameter is available only for the HTTP-type and HTTPS-type health check templates. The default value is “200 OK”.
- **Result flag**: specifies the determined health check result when the expected HTTP response is received. This parameter is available only for the HTTP-type and HTTPS-type health check templates. The default value is “up”.
- **Result flag**: specifies the determined health check result when the expected HTTP response is received. This parameter is available only for the HTTP-type health check template. The default value is “up”.
- **Health check interval**: specifies the interval of sending health check packets.
- **Health check timeout**: specifies the timeout value of the health check packet. The value of the health check timeout should not be greater than that of the health check interval.
- **Maximum retry times**: specifies the maximum times of health checks to determine the health status of the destination IP address. For example, if it is set to 3, the health status of the destination IP address will be regarded as Up when the health check result is Up for three consecutive times; the health status of the destination IP address will be regarded as Down when the health check result is Down for three consecutive times.
- **Destination IP address**: specifies the IP address of the health check object. Usually this parameter is left empty. When the health check template is applied to the service IP or service pool, the destination IP address will be replaced by the corresponding service IP in the generated health check instances.
- **Destination port**: specifies the port of the health check object. This parameter is available only for the TCP-type, HTTP-type and HTTPS-type health check templates. Usually, this parameter is set to 0. When the health check template is applied to the service IP or service pool, the destination IP port will be replaced by the health check port of the corresponding service IP in the generated health check instances. Note that the destination port of the TCP-type, HTTP-type and HTTPS-type health check template and the health check port of the associated service IP cannot be both 0s.
- **Source IP address**: specifies the source IP address of the health check packets.
- **Gateway IP address**: specifies the IP address of the gateway forwarding health check packets in a multi-link network environment.

After the health check template is applied to the specified service IP, SDNS will generate the health check instance where the destination IP address is replaced by the service IP, and send the health check packet to the service IP. If the health check template is applied to the specified service pool, SDNS will generate the health check instance for each member (service IP) of the

service pool. The administrator can choose to apply the health check templates to the service IP or service pool based on actual requirements.

If multiple health check templates are applied to a service IP, the service IP has multiple health check instances; if multiple health check templates are applied to a service pool, these health check templates will be applied to each service IP in the service pool. Therefore, each service IP in the service pool has multiple health check instances. SDNS allows the administrator to specify the relationship among health check instances of a service IP as an individual service IP or as a member of a service pool.

- "AND": When the results of all health check instances of the service IP are Up, the status of the service IP is set as Up.
- "OR": When the result of any health check instance for the service IP is Up, the status of the service IP is set as Up.
- "Mixed Relationship": When the results of all health check instance for the service IP meet the requirement of the mixed relationship expression defined in "**sdns pool health mixrelation**" command, the status of the service IP is set as Up.

The default relationship among health check instances of a service IP is "AND".

For example:

```
AN(config)#sdns monitor icmp "hc_icmp" 5 5 3 0.0.0.0 0.0.0.0 0.0.0.0
AN(config)#sdns monitor tcp "hc_tcp" 5 5 3 0.0.0.0 1000 0.0.0.0 0.0.0.0
AN(config)#sdns monitor http "hc_http" "HEAD / HTTP/1.0\r\n\r\n" "200 OK" "up" 5 5 3
0.0.0.0 0 0.0.0.0 0.0.0.0
AN(config)#sdns service ip "sv1" 10.8.6.201 80
AN(config)#sdns service ip "sv2" 10.8.6.202 80
AN(config)#sdns service ip "sv3" 10.8.6.203 80
AN(config)#sdns service ip "sv4" 10.8.6.204 80
AN(config)#sdns pool name "pool1" 1
AN(config)#sdns pool service "pool1" "sv1"
AN(config)#sdns pool service "pool1" "sv2"
AN(config)#sdns pool service "pool1" "sv3"
AN(config)#sdns pool service "pool1" "sv4"
AN(config)#sdns pool monitor apply "pool1" "hc_icmp"
AN(config)#sdns pool monitor apply "pool1" "hc_tcp"
AN(config)#sdns pool monitor apply "pool1" "hc_http"
```

20.3.7.2 SNMP Data Collection

SDNS provides the SNMP data collection function to collect the data of specified SNMP OIDs from SDNS services. The snmp method of the SDNS service pool needs to depend on the SNMP data collection function. SDNS computes the metric of every SDNS service based on the SNMP expression associated with the service pool. The SNMP expression determines the data of what

SNMP OIDs will be used to compute the metric. To collect the data of such SNMP OIDs from every SDNS service, the administrator needs to configure SNMP data collection.

For the snmp method to work normally, the administrator needs to:

Define the SNMP OIDs whose data needs to be collected.

Define the SNMP expression used to compute the metric of the SDNS service.

Define the SDNS service pool using the snmp method and the associated SNMP expression.

Define SNMP-type data collection templates for the SNMP OIDs.

Apply the SNMP-type data collection templates to the SDNS service pool to generate SNMP data collection instances.

20.3.7.2.1 SNMP OID and SNMP Expression

Every SNMP expression uses the data of one SNMP OID at minimum and three SNMP OIDs at maximum. The format of the SNMP expression is as follows:

Data of OID1 × Weight of OID1 + Data of OID2 × Weight of OID2 + Data of OID3 × Weight of OID3

Before defining the SNMP expression, the administrator needs to define the SNMP OIDs to be used. A maximum of eight SNMP OIDs can be defined.

After the system receives the data of the SNMP OIDs, SDNS will compute the metric of every SDNS service based on the SNMP expression, and sort up SDNS services according to the sorting mode configured for the SDNS service pool. The SDNS service pool supports two sorting modes:

- Ascending: SDNS service IPs are sorted up in the ascending order of the metrics.
- Descending: SDNS service IPs are sorted up in the descending order of the metrics.

20.3.7.2.2 SNMP Data Collection Template and Instance

To collect the data of SNMP OIDs from the SDNS services of an SDNS service pool, the administrator needs to create SNMP data collection templates for these SNMP OIDs and apply these templates to the SDNS service pool. In this way, the system generates SNMP data collection instances for every SDNS service in the SDNS service pool.

The SNMP data collection template is comprised of the following elements:

- **OID name**: specifies name of the SNMP OID whose data needs to be collected.
- **Community string**: specifies the community string of the SNMP server. The default value is “public”.
- **SNMP version**: specifies SNMP protocol version used for communication.
- **Interval**: specifies the interval of data collection.
- **Destination IP address**: specifies the IP address of the data collection object. Usually this parameter is left empty. When the data collection template is applied to the service pool, the

destination IP address will be replaced by the corresponding service IP in the generated data collection instances.

- **Destination port**: specifies the SNMP port of the data collection object. The default value is 161.
- **Source IP address**: specifies the source IP address of the SNMP requests.
- **Gateway IP address**: specifies the IP address of the gateway forwarding SNMP requests in a multi-link network environment.

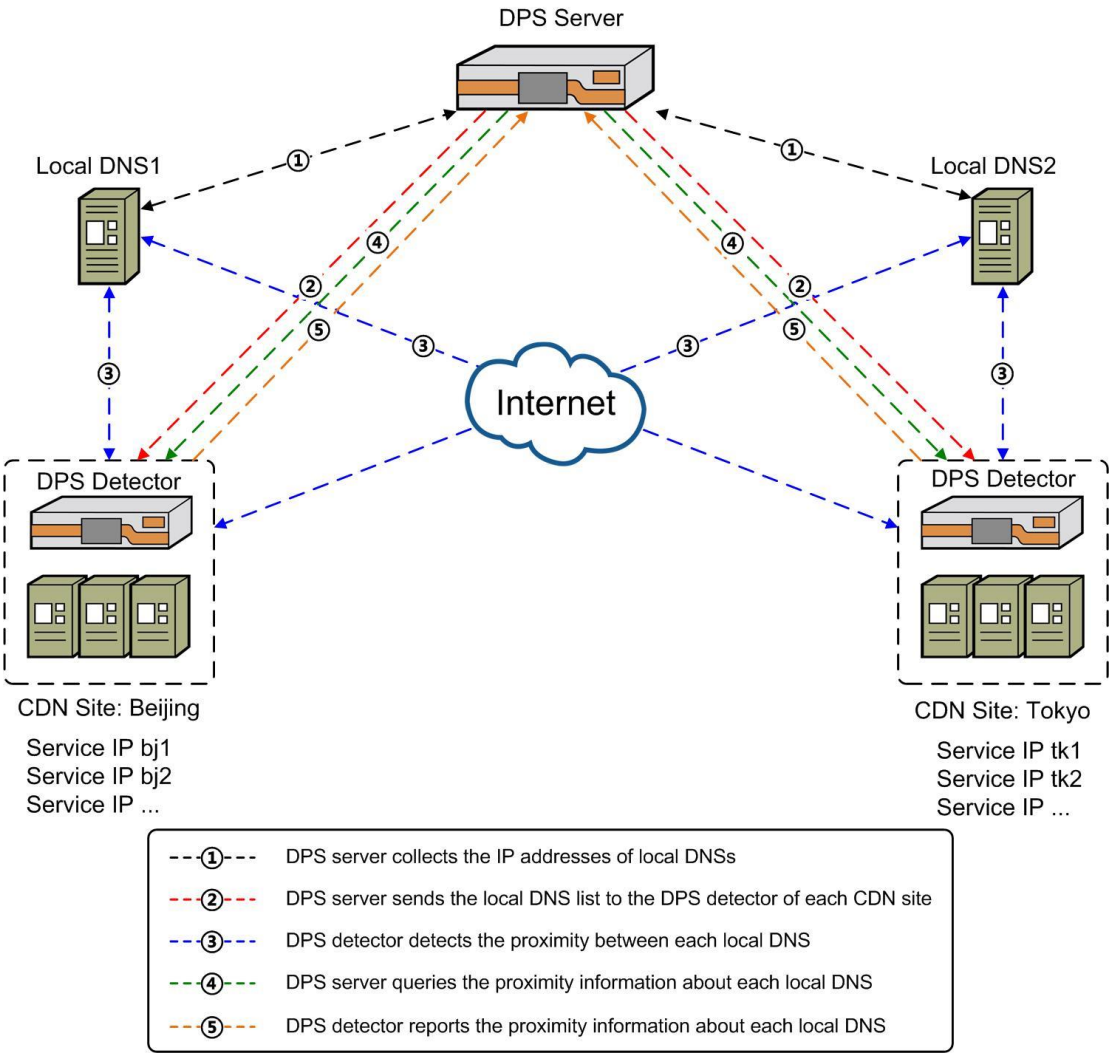
Note: The SNMP-type data collection template can be applied only to the SDNS service pool.

Configuration example:

```
AN(config)#sdns snmp oid "id1" ".1.3.6.1.4.1.7564.30.1.0"
AN(config)#sdns snmp oid "id2" ".1.3.6.1.4.1.7564.4.2.0"
AN(config)#sdns snmp oid "id3" ".1.3.6.1.4.1.7564.30.2.0"
AN(config)#sdns snmp exp weight "expression1" "id1" 1 "id2" 2 "id3" 3
AN(config)#sdns service ip "sv1" 10.8.6.201 0
AN(config)#sdns service ip "sv2" 10.8.6.202 0
AN(config)#sdns service ip "sv3" 10.8.6.203 0
AN(config)#sdns service ip "sv4" 10.8.6.204 0
AN(config)#sdns pool name "pool1" 1
AN(config)#sdns pool method primary "pool1" "snmp" "expression1" "asc"
AN(config)#sdns pool service "pool1" "sv1"
AN(config)#sdns pool service "pool1" "sv2"
AN(config)#sdns pool service "pool1" "sv3"
AN(config)#sdns pool service "pool1" "sv4"
AN(config)#sdns monitor snmp collector "dc1" "id1" "public" "v2c" 60 0.0.0.0 161 0.0.0.0
0.0.0.0
AN(config)#sdns monitor snmp collector "dc2" "id2" "public" "v2c" 60 0.0.0.0 161 0.0.0.0
0.0.0.0
AN(config)#sdns monitor snmp collector "dc3" "id3" "public" "v2c" 60 0.0.0.0 161 0.0.0.0
0.0.0.0
AN(config)#sdns pool monitor apply "pool1" "dc1"
AN(config)#sdns pool monitor apply "pool1" "dc2"
AN(config)#sdns pool monitor apply "pool1" "dc3"
```

20.3.8SDNS Dynamic Proximity System (DPS)

SDNS supports using the Dynamic Proximity System (DPS) to generate dynamic proximity rules.



20.3.8.1.1.1.1 SDNS DPS

DPS is comprised of DPS servers and DPS detectors.

➢ DPS Server

The DPS master (the APV appliance) mainly provides the following function:

- Collects the IP addresses of local DNSs and sends the address list to DPS detectors for detection.
- Queries the proximity detection results from DPS detectors.
- Generates dynamic proximity rules based on the proximity detection results.

➢ DPS Detector

The DPS detector mainly provides the following functions:

- Receives the IP address list of local DNSs from DPS servers.
- Sends detection packets to local DNSs to obtain three types of proximity information: Round Trip Time (RTT), Packet Loss Rate (PLR) and hops.

- Reports the proximity detection results to the DPS servers.

Note: The DPS detector can be deployed on the APV appliance or on other servers running the Linux or FreeBSD operating system.

➢ **How to Generate Dynamic Proximity Rules?**

When receiving the proximity detection results from the specified SDNS region's DPS detector, the DPS server will calculate the metrics of the paths between the DPS detector (region) and all local DNSs. The DPS server will then generate a dynamic proximity rule between the local DNS and SDNS region whose path has the smallest metric.

SDNS supports four DPS methods:

- rtt: Metric = RTT
- plr: Metric = PLR
- hops: Metric = hops
- mix: Metric = RTT x Weight + PLR x Weight + Hops x Weight

When the mix DPS method is used, the administrator needs to specify the weights for RTT, PLR and hops.

➢ **Working Mechanism of DPS**

- The local DNS sends DNS queries to the DPS server.
- After a while, the DPS server collects a list of local DNSs and their IP addresses.
- The DPS server sends the IP addresses of the collected local DNSs to the DPS detector of each region (CDN site) for detection.
- The DPS detector starts to detect the proximity between its region and all local DNSs.
- When the DPS server queries the proximity detection results, the DPS detector reports proximity detection results to the DPS server.
- The DPS server generates the dynamic proximity rule based on received proximity detection results.

For example:

```
AN(config)#sdns region name "Beijing"
AN(config)#sdns region name "Tokyo"
AN(config)#sdns on
AN(config)#sdns dps interval query 1200
AN(config)#sdns dps method rtt
AN(config)#sdns dps expire 300
AN(config)#sdns dps interval send 120
AN(config)#sdns dps detector "Beijing" 211.100.1.100 44544 900 172800
```

```
AN(config)#sdns dps detector "Tokyo" 210.10.1.100 44544 900 172800
AN(config)#sdns dps on
```

20.3.9Full-DNS Resolution

In the APV appliance, SDNS processes DNS queries of the A, AAAA, PTR, MX, NS and CNAME types, and the Full-DNS function processes DNS queries of all the other types, such as MX and PTR.

When receiving DNS queries of the A, AAAA, PTR, MX, NS and CNAME types, the APV appliance dispatches them to SDNS for processing. When receiving DNS queries of other types, the APV appliance dispatches them to Full-DNS for processing. If the domain names in the DNS queries of the A, AAAA, PTR, MX, NS and CNAME types are not configured on SDNS, the APV appliance will return empty DNS responses. When Full-DNS fails to process DNS queries of other types, the APV appliance will also return empty DNS responses.

When the Recursive Query function is enabled:

- The APV appliances will forward the DNS queries of the A, AAAA, PTR, MX, NS and CNAME types to Full-DNS for processing if the domain names in the DNS queries have not been configured on SDNS.
- The APV appliances will forward the DNS queries to other external DNSs to perform recursive query if Full-DNS cannot process them and finally returns the resolution results to the local DNS.

By default, the Recursive Query function is disabled. To enable the Recursive Query function, execute the following command:

```
AN(config)#sdns recursion on
```

Note: Full-DNS needs to be configured via WebUI.

Assume that only resource records of the A, AAAA, or CNAME type are available on SDNS for the domain name “image.example.com”. To ensure that the DNS queries of the other record types can be correctly processed, you are advised to configure a TXT record in the following format for the domain name “example.com” in the Full-DNS zone file:

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

Note: The “text string” can be any description about this domain name.

20.3.10 IPv6 Support

SDNS provides comprehensive IPv6 support.

- AAAA DNS resolution: SDNS can process the DNS query of the AAAA type.

- SDNS service IP: The service IP can be an IPv4 or IPv6 addresses.
- SDNS service pool: The service IPs in a service pool can be either all IPv4 addresses or IPv6 addresses. Note that a service pool cannot contain both IPv4 and IPv6 addresses.
- SDNS proximity rule: Proximity rules support IPv6.
- SDNS DPS: SDNS supports IPv6 DPS detectors and supporting detecting IPv6 local DNSs.
- SDNS health check: SDNS supports performing health checks on IPv4 and IPv6 service IPs.

20.3.11 Data Center Synchronization

The SDNS data center synchronization function aims at placing the SDNS devices located at the same or in different places in a synchronization group logically. Inside the group, every SDNS device functions as a data center. Each synchronization group supports up to sixty four data centers (defined by the "**sdns dc**" command). The data center will use the IP address defined in this command to communicate with other data centers.

Each data center identifies its local service IP by its association with an SDNS service (by the "**sdns service ip**" command). After the association (between the service IP and a health check template), this service IP and the generated health check instance will be considered belonging to this data center. The service IP that is not associated with any data center and the generated health check instance do not belong to any data center. Each data center detects only the status of health check instances belonging to it and those that do not belong to any data centers, and at the same time it will exchange the status of its local health check instances with other data centers. In this way, all data centers will share each other's service IP status, which helps guarantee the consistency of DNS resolution at the global level.

20.3.11.1 Configuration and Status Synchronization

Configuration synchronization between data centers can be implemented by the "**sdns sync config from**" and "**sdns sync config to**" commands. Note that these commands do not support synchronization of the "**sdns dc**" and "**sdns ipregion proximity**" configurations. Therefore, every data center must have the local and remote data centers defined before a configuration synchronization operation. In addition, if the IP region table function is to be used, the IP region proximity rules must be manually configured on each data center.

The service IP status synchronization mechanism between data centers is composed of two parts: proactive request and proactive push. Following are the specific synchronization principles:

- Each data center proactively detects the status of health check instances belonging to the local data center and also detects the status of local health check instances that do not belong to any data center. And each data center will generate a list of monitor items, including local and all other data centers' monitor instance IDs and status.

- When a local service IP alternates between up and down, the data center will proactively send the status change to other data centers via the PUSH message. Any data center receiving a PUSH message will update the corresponding instance status.
- Each data center sends GET requests to other data centers at a specific interval (3s by default) to query instance status at other data centers. When receiving a GET request, the data center will check the instance status according to the queried instance list in the GET requests and reply the status to the requester in GET ACK messages. When receiving a GET ACK message, the data center will update the corresponding monitor instance status. If it cannot find the status of a specific instance in the message, it will record a health check failure for this instance. If it does not receive the expected GET ACK message within the specified period of time (2s by default), it will record a health check failure for all instances that have been queried.

The sending interval of GET requests and timeout value in waiting for a response during status synchronization between data centers can be defined using the "**sdns sync status**" command. The timeout for waiting for a response cannot be greater than the interval for sending a request.

Except for the normal mode, the system adds a failover probe mode. When the synchronization of service IP health status between data centers failed, the local data center will actively send health check status detection packets to other data centers.

- In the normal mode, the system adopts the original behavior. When the local data center did not receive the status of the health check instance returned by other data centers, the system will not perform active probe, and it will directly set the health check instance status of the data center to "Down".
- In the failover probe mode, the system actively probes when the local data center did not receive the health check instance status returned by other data centers. Specifically, it is divided into the following two situations:
  - If the destination IP address is set in the health check template, and the destination IP address is different from the SDNS service IP address, after applying the health check template to the SDNS service, the system will create two health check instances, the primary instance and the standby instance (viewed by the command "**show statistics sdns dc instance**").
    - When the local data center detects the health check status of other data centers, the health check status of the primary instance will be marked as the detected status, and the health check status of the standby instance is the same as that of the primary instance.
    - If the local data center cannot detect the health check status of other data centers, the health check status of the primary instance will be marked as "Down", and the health check status of the standby instance is obtained by detecting the SDNS service IP address.
  - If the destination IP address of the health check template is the same as the IP address of the SDNS service, after associating the SDNS service with the

health check template, only one health check instance will be generated, and the local data center will not perform active detection if the status of the health check instance fails

- If the destination IP address is not set for the health check template, the system generates only one health check instance after applying the health check template to the SDNS service. There are two specific situations:
  - When the local data center receives the status of a health check instance from other data centers, it will directly use this status to identify the status of the health check instance.
  - When the local data center did not receive the status of the health check instance from other data centers, the system will actively detect the health check instance status.

A configuration example is as follows:

```
Demo(config)#sdns service ip abc 192.168.83.11 80 dc1 1
Demo(config)#sdns monitor tcp h1 5 5 3 192.168.88.10 80 0.0.0.0 0.0.0.0
Demo(config)#sdns service monitor apply abc h1
Demo(config)#show statistics sdns dc instance
DC Name:              dc1
Ip Address:           192.168.93.100
Port:                 60000
ID:                   0
Type:                 local|master
Status:               LISTEN
Instance Count:       2
    Instance ID:      0
    References:       1
    HC Mode:          0
    Status:           DOWN
    Monitor:          h1
    Type:             TCP
    Interval:         5
    Timeout:          5
    Max retries:      3
    Address:          192.168.88.10
    Port:             80
    Source:           0.0.0.0
    Gateway:          0.0.0.0

    Instance ID:      1
    References:       1
    HC Mode:          1
    Primary ID:       0
    Status:           DOWN
```

```
    Monitor:          h1
    Type:             TCP
    Interval:         5
    Timeout:          5
    Max retries:      3
    Address:          192.168.83.11
    Port:             80
    Source:           0.0.0.0
    Gateway:          0.0.0.0

DC Name:              dc2
Ip Address:           192.168.93.200
Port:                 60000
ID:                   1
Type:                 remote
Status:               ESTABLISHED
Instance Count:       0
```

20.3.11.2 SDNS Runtime Configuration Synchronization

After enabling the SDNS runtime configuration synchronization function, when the configuration of a data center changes, the data center synchronizes these changes with other data centers in real time through the CPUSH method. The system only synchronizes configuration changes made after this function is enabled. By default, this function is disabled.

SDNS runtime configuration synchronization between data centers is only possible when this function is enabled on both the local and remote data centers. SDNS configuration items that do not need to be synchronized are listed in a blacklist. After the SDNS runtime configuration synchronization function is enabled, all configurations except those in the blacklist will be synchronized.

SDNS runtime configuration synchronization supports logging function, which is used to display the information during the SDNS runtime configuration synchronization process. The SDNS runtime configuration synchronization function and the logging function will be enabled or disabled at the same time.

- **SDNS Runtime Configuration Synchronization**

```
Demo(config)#sdns sync config runtime on
Demo(config)#show sdns sync config
sdns sync config runtime on
Demo(config)#show sdns sync status
sdns sync status 3 2
```

- **SDNS Runtime Configuration Synchronization Log Analysis**

```
Demo(config)#show sdns sync log
```

```
dc name: dc2(192.168.93.200)
Total failed :  2
Total loss   :  0
Total success:  65
Failed cli command:
no sdns service ip “resorts1”
Service name “resorts1” not found.
Failed tp execute “no sdns service ip” “resorts1”
no sdns service ip “resorts2”
Service name “resorts2” not found.
Failed tp execute “no sdns service ip” “resorts2”
dc name: dc22(182.16.8.222)
Total failed :  0
Total loss   :  2
Total success:  65
Loss cli command:
no sdns service ip “resorts1”
no sdns service ip “resorts2”
Demo(config)#show sdns sync log pool
192.168.93.200:3:clear sdns pool cname
192.168.93.200:4:clear sdns pool name
192.168.93.200:33:sdns pool name "p1" 1
192.168.93.200:34:sdns pool name "return" 1
......
```

20.3.11.3 HA Support Within A Data Center

SDNS data center supports the HA active/standby mode. Within each data center, it is allowed to deploy one active device and one or more standby devices for backup. In addition, the data center’s IP address (defined by the “**sdns dc**” command) must be set to the HA floating IP address. The device on which the floating IP address resides functions as the active device in the data center and joins in the synchronization group. The standby device does not belong to any data center and will not join the synchronization group either. They will detect the status of health check instances based on the following mechanisms. When the active device experiences a failure, the two devices can quickly perform a switchover, and the standby device will join in the synchronization group as the active device.

Active device:

- Proactively detects the status of health check instances belonging to the local data center. When the local service IP status changes, it will PUSH the change to other data centers. When it receives a PUSH message, it will update the status of corresponding instances.
- Proactively detects the status of local health check instances that do not belong to any data centers.

- Sends GET requests to other data centers at a specific interval (3s by default) to query instance status at other data centers. When receiving a GET request, it will reply the status of corresponding health check instances of the local data center.

Standby device:

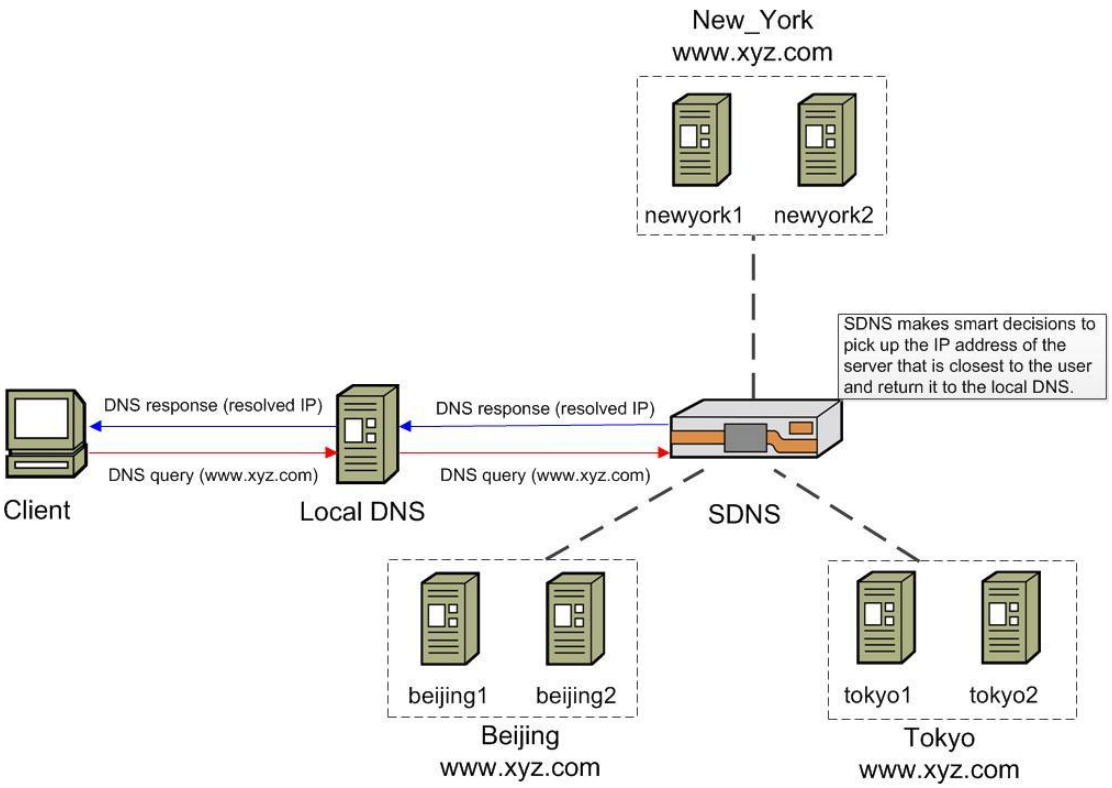
- Proactively detects the status of local health check instances that do not belong to any data centers.
- Sends GET requests to other data centers (including the local data center) at a specific interval (3s by default) to query instance status at other data centers.

In addition, when SDNS Runtime Configuration Synchronization is performed between SDNS data center hosts, the system supports synchronization between SDNS data center hosts and standby devices through HA.

20.4 Configuration Example

20.4.2Configuration Objectives

Assume that “Beijing”, “Tokyo” and “New_York” sites in the CDN network provides the same service identified by the domain name “www.xyz.com” and each CDN site has two servers to provide the service simultaneously, as shown in the following figure.



20.4.2.1.1.1.1 Application Example of SDNS Region Policies

The configuration objectives are as follows:

- For the DNS query sent from the local DNSs near to the “Beijing” site, SDNS will return the IP address of either server on the “Beijing” site. When both servers on the “Beijing” site become unavailable, SDNS will return the address of either server on the “Tokyo” site. When both servers on the “Tokyo” site also become unavailable, SDNS will return the address of either server on the “New_York” site.
- For the DNS query sent from the local DNSs near to the “Tokyo” site, SDNS will return the IP address of either server on the “Tokyo” site. When both servers on the “Tokyo” site become unavailable, SDNS will return the address of either server on the “Beijing” site. When both servers on the “Beijing” site also become unavailable, SDNS will return the address of either server on the “New_York” site.
- For the DNS query sent from the local DNSs near to the “New_York” site, SDNS will return the IP address of either server on the “New_York” site. When both servers on the “New_York” site become unavailable, SDNS will return the address of any server on the “Beijing” or “Tokyo” site.

To achieve the preceding objectives, the administrator can:

- Define three CDN sites as regions “Beijing”, “Tokyo”, and “New_York” respectively.
- Define region policies to associate DNS queries coming from the regions “Beijing”, “Tokyo”, and “New_York” with the service pools “pool_beijing”, “pool_tokyo” and “pool_newyork” respectively. The service pool “pool_beijing” includes the service IPs “beijing1” and “beijing2”; the service pool “pool_newyork” includes the service IPs “tokyo1” and “tokyo2”; the service pool “pool_newyork” includes the service IPs “newyork1” and “newyork2”.
- Configure fallback pools for the service pools “pool_beijing”, “pool_tokyo” and “pool_newyork”. The service pools “pool_beijing” and “pool_tokyo” can fall back to the service pool “pool_asia” that includes all service IPs of the service pools “pool_beijing” and “pool_tokyo”. The service pools “pool_newyork” and “pool_asia” can fall back to the service pool “pool_world” that includes all service IPs of the service pools “pool_beijing”, “pool_tokyo”, and “pool_newyork”.

20.4.3Configuration Examples

20.4.3.1 Basic Configurations

- **Add the SDNS Host Name**

For example:

```
AN(config)#sdns host name "www.xyz.com" 60
```

- **Add SDNS Service IPs**

For example:

```
AN(config)#sdns service ip "beijing1" 211.100.20.101 80
```

```
AN(config)#sdns service ip "beijing2" 211.100.20.102 80
AN(config)#sdns service ip "tokyo1" 210.10.50.101 80
AN(config)#sdns service ip "tokyo2" 210.10.50.102 80
AN(config)#sdns service ip "newyork1" 216.100.10.101 80
AN(config)#sdns service ip "newyork2" 216.100.10.102 80
```

- **Configure SDNS Service Pools and Pool Methods**

For example:

```
AN(config)#sdns pool name "pool_beijing" 1
AN(config)#sdns pool name "pool_tokyo" 1
AN(config)#sdns pool name "pool_newyork" 1
AN(config)#sdns pool method primary "pool_beijing" "ipo"
AN(config)#sdns pool failover "pool_beijing" on
AN(config)#sdns pool preempt "pool_beijing" on
AN(config)#sdns pool method primary "pool_tokyo" "rr"
AN(config)#sdns pool method primary "pool_newyork" "rr"
AN(config)#sdns pool service "pool_beijing" "beijing1"
AN(config)#sdns pool service "pool_beijing" "beijing2"
AN(config)#sdns pool member priority "pool_beijing" "beijing1" 1
AN(config)#sdns pool member priority "pool_beijing" "beijing2" 2
AN(config)#sdns pool service "pool_tokyo" "tokyo1"
AN(config)#sdns pool service "pool_tokyo" "tokyo2"
AN(config)#sdns pool service "pool_newyork" "newyork1"
AN(config)#sdns pool service "pool_newyork" "newyork2"
```

- **Add SDNS Region Policies**

For example:

```
AN(config)#sdns region name "Beijing"
AN(config)#sdns region name "Tokyo"
AN(config)#sdns region name "New_York"
AN(config)#sdns proximity 211.100.0.0 255.255.0.0 "Beijing"
AN(config)#sdns proximity 210.10.0.0 255.255.0.0 "Tokyo"
AN(config)#sdns proximity 216.100.0.0 255.255.0.0 "New_York"
AN(config)#sdns policy region "policy_beijing" "www.xyz.com" "pool_beijing" "Beijing" 0
AN(config)#sdns policy region "policy_tokyo" "www.xyz.com" "pool_tokyo" "Tokyo" 0
AN(config)#sdns policy region "policy_newyork" "www.xyz.com" "pool_newyork"
"New_York" 0
```

- **Configure SDNS Health Check**

For example:

```
AN(config)#sdns monitor icmp "hc1" 5 5 3 0.0.0.0 0.0.0.0 0.0.0.0
AN(config)#sdns monitor tcp "hc2" 5 5 3 0.0.0.0 1000 0.0.0.0 0.0.0.0
```

```
AN(config)#sdns service monitor apply "beijing1" "hc1"
AN(config)#sdns service monitor apply "beijing1" "hc2"
AN(config)#sdns service monitor apply "beijing2" "hc1"
AN(config)#sdns service monitor apply "beijing2" "hc2"
AN(config)#sdns service monitor apply "tokyo1" "hc1"
AN(config)#sdns service monitor apply "tokyo1" "hc2"
AN(config)#sdns service monitor apply "tokyo2" "hc1"
AN(config)#sdns service monitor apply "tokyo2" "hc2"
AN(config)#sdns service monitor apply "newyork1" "hc1"
AN(config)#sdns service monitor apply "newyork1" "hc2"
AN(config)#sdns service monitor apply "newyork2" "hc1"
AN(config)#sdns service monitor apply "newyork2" "hc2"
AN(config)#sdns service health relation "beijing1" "and"
AN(config)#sdns service health relation "beijing2" "and"
AN(config)#sdns service health relation "tokyo1" "and"
AN(config)#sdns service health relation "tokyo2" "and"
AN(config)#sdns service health relation "newyork1" "and"
AN(config)#sdns service health relation "newyork2" "and"
```

- **Configure Fallback Pools for SDNS Service Pools**

For example:

```
AN(config)#sdns pool name "pool_asia" 1
AN(config)#sdns pool method primary "pool_asia" "rr"
AN(config)#sdns pool service "pool_asia" "beijing1"
AN(config)#sdns pool service "pool_asia" "beijing2"
AN(config)#sdns pool service "pool_asia" "tokyo1"
AN(config)#sdns pool service "pool_asia" "tokyo2"
AN(config)#sdns pool name "pool_world" 1
AN(config)#sdns pool method primary "pool_world" "rr"
AN(config)#sdns pool service "pool_world" "beijing1"
AN(config)#sdns pool service "pool_world" "beijing2"
AN(config)#sdns pool service "pool_world" "tokyo1"
AN(config)#sdns pool service "pool_world" "tokyo2"
AN(config)#sdns pool service "pool_world" "newyork1"
AN(config)#sdns pool service "pool_world" "newyork2"
AN(config)#sdns pool fallback "pool_beijing" "pool_asia"
AN(config)#sdns pool fallback "pool_tokyo" "pool_asia"
AN(config)#sdns pool fallback "pool_newyork" "pool_world"
AN(config)#sdns pool fallback "pool_asia" "pool_world"
```

- **Enable SDNS**

For example:

```
AN(config)#sdns on
```

20.4.3.2 Advanced Configurations

With basic configurations completed, a dedicated line can be laid out to connect the sites and the SDNS data center function can be configured to implement service IP status synchronization among the three sites.

1. Define data centers at each site.

Beijing:

```
AN(config)#sdns dc beijing_dc 192.168.0.136 10008 local
AN(config)#sdns dc newyork_dc 10.0.9.12 10009 remote
AN(config)#sdns dc tokyo_dc 172.16.9.8 12001 remote
```

New York:

```
AN(config)#sdns dc newyork_dc 10.0.9.12 10009 local
AN(config)#sdns dc tokyo_dc 172.16.9.8 12001 remote
AN(config)#sdns dc beijing_dc 192.168.0.136 10008 remote
```

Tokyo：

```
AN(config)#sdns dc tokyo_dc 172.16.9.8 12001 local
AN(config)#sdns dc beijing_dc 192.168.0.136 10008 remote
AN(config)#sdns dc newyork_dc 10.0.9.12 10009 remote
```

Associate service IPs with data centers at any of the Beijing, New York and Tokyo sites.

```
AN(config)#sdns service ip "beijing1" 211.100.20.101 0 beijing_dc
AN(config)#sdns service ip "beijing2" 211.100.20.102 0 beijing_dc
AN(config)#sdns service ip "tokyo1" 210.10.50.101 0 tokyo_dc
AN(config)#sdns service ip "tokyo2" 210.10.50.102 0 tokyo_dc
AN(config)#sdns service ip "newyork1" 216.100.10.101 0 newyork_dc
AN(config)#sdns service ip "newyork2" 216.100.10.102 0 newyork_dc
```

Set the interval at which GET requests are sent and the response timeout value. Note that the interval value must be equal to or larger than the timeout value.

```
AN(config)#sdns sync status 5 2
```

Synchronize configurations to other data centers.

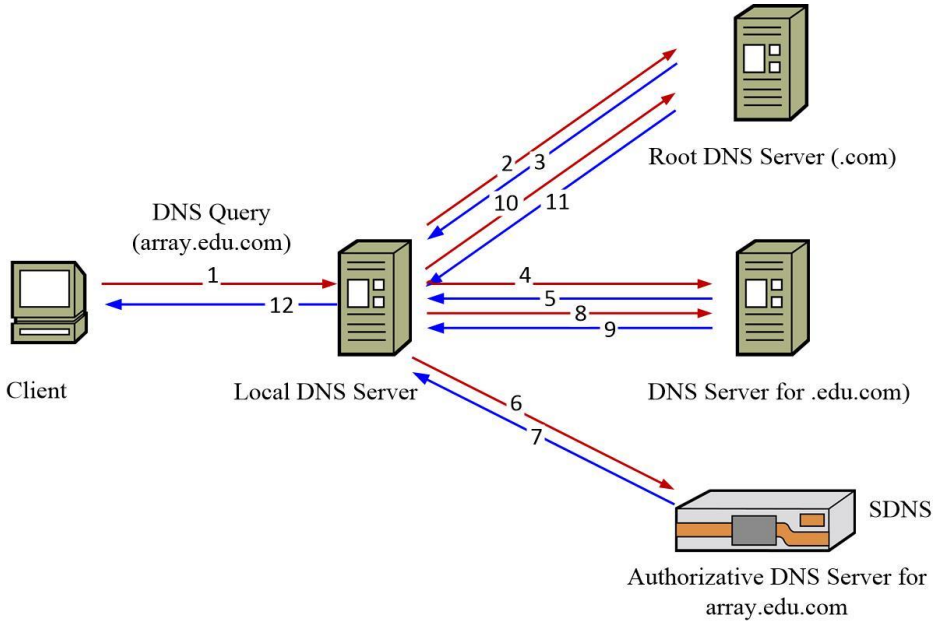
```
AN(config)#sdns sync config to all
```

20.5 SDNS DNSSEC

SDNS supports the DNSSEC function, which is designed to defend against man-in-the-middle attack during DNS recursive queries, and is used to prevent against DNS spoofing attacks and DNS cache poisoning.

DNSSEC helps ensure the integrity and authenticity of DNS responses by utilizing a cryptographic signature mechanism. The DNSSEC records signing is achieved by the combined use of ZSK (Zone Signing Key) and KSK (Key Signing Key). In the DNSSEC mechanism, ZSK is used to sign all the Resource Records in the zone, while KSK is used to sign only the ZSK (DNSKEY Record). The signing Keys and RRSIG records provided by the DNS server help ensure that the Resource Records have not been tampered in any way. To further check the legitimacy of KSK, the local DNS server queries the upper-level DNS server for the DS (Delegation Signer) record, which contains the hash of the DNSKEY record, to create a chain of trust for the DNS resolution process.

The DNS resolution process of SDNS DNSSEC is as shown in the following figure:



20.5.2.1.1.1.1 DNS Resolution Process of SDNS DNSSEC

As shown in the preceding figure, the DNS resolution process of SDNS DNSSEC is as follows:

1. The client sends a DNS query to the local DNS server for resolving the domain name “array.edu.com”.

The local DNS server requests the root DNS server for the responding Resource Record (such as A record) of this domain name and the public key of the root DNS server.

The DNS server sends the IP address of upper-level DNS server (for the.edu.com domain) and its own public key to the local DNS server.

The local DNS server requests the upper-level DNS server for the A record.

The upper-level DNS server sends the IP address of the authoritative DNS server (for the array.edu.com domain name) to the local DNS server.

The local DNS server requests the authoritative DNS server for the A record and the DNSKEY record of this domain.

The authoritative DNS server (SDNS) sends the A record, RRSIG record (singed A record) and its own public key to the local DNS server.

The local DNS server requests the upper-level DNS server for the DS record of the authoritative DNS server for array.edu.com domain name. The DS record contains the message digest of the KSK public key, and helps verify its authenticity.

The upper-level DNS server returns the required DS record to the local DNS server.

The local DNS server requests the root DNS server for the DS record of the upper-level DNS server to verify the authenticity of the KSK public key in the upper-level DNS server.

The root DNS server returns the required DS record to the local DNS server.

The local DNS server returns the A record of the domain name to the client.

After the SDNS configuration for the specified SDNS host name (domain name) is completed, the appliance will function as the authoritative DNS server for this host name. To use the DNSSEC function, administers need to perform the following configurations:

- Create ZSK and KSK signing keys
- Publish the signing keys onto the specified SDNS host name
- Sign the Resource Records for the specified SDNS host name to produce RRSIG records and DS file.
- Upload the DS file of the signing key to the upper-level DNS server
- Enable the DNSSEC function for SDNS

Example:

1. Configure the SDNS function for the SDNS host name “array.edu.com”.

```
AN(config)#sdns host name "array.edu.com" 60
AN(config)#sdns service ip "service1" 211.100.20.101 80
AN(config)#sdns service ip " service2" 211.100.20.102 80
AN(config)#sdns pool name "pool1" 1
AN(config)#sdns pool method primary "pool1" "rr"
AN(config)#sdns pool service "pool1" " service1"
AN(config)#sdns pool service "pool1" " service2"
AN(config)#sdns policy default "array.edu.com" "pool1"
AN(config)#sdns on
```

Note: The “max_rr_count” parameter in the “**sdns pool name**” command can only be set to 1.

Generate the ZSK and KSK for array.edu.com.

```
AN(config)#sdns dnssec keygen array.edu.com ksk
AN(config)#sdns dnssec keygen array.edu.com zsk
```

```
AN(config)#show sdns dnssec keygen
Karray.edu.com.+005+00942+zsk+RSASHA1+1024
Karray.edu.com.+005+28783+ksk+RSASHA1+1024
```

Publish the ZSK and KSK signing keys.

```
AN(config)#sdns dnssec keypub array.edu.com ksk
"Karray.edu.com.+005+28783+ksk+RSASHA1+1024"
AN(config)#sdns dnssec keypub array.edu.com zsk
"Karray.edu.com.+005+00942+zsk+RSASHA1+1024"
AN(config)#show sdns dnssec keypub
array.edu.com Karray.edu.com.+005+00942+zsk+RSASHA1+1024
array.edu.com Karray.edu.com.+005+28783+ksk+RSASHA1+1024
```

Sign the Resource Records for array.edu.com.

```
AN(config)#sdns dnssec zonesig array.edu.com
AN(config)#show sdns dnssec zonesig
signed host: array.edu.com
```

Display the DNSSEC status information.

```
AN(config)#show sdns dnssec status array.edu.com
host status flags: HOST_PUBLISH|HOST_KEY|HOST_SIGN
ttl: 60(s)
zsk key counter: 1
ksk key counter: 1
ZSK key(0):
        keyfile: Karray.edu.com.+005+43368+zsk+RSASHA1+1024
        keyid: 43368
        type: ZSK
        algorithm: RSASHA1
        status_flags: KEY_PUBLISH|KEY_SIGN|KEY_TIMER
        ttl: 1(hour)
        ex_time: 30(days)
        ro_time: 25(days)
KSK key(0):
        keyfile: Karray.edu.com.+005+07177+ksk+RSASHA1+1024
        keyid: 7177
        type: KSK
        algorithm: RSASHA1
        status_flags: KEY_PUBLISH|KEY_SIGN|NO
        ttl: 1(hour)
        ex_time: 30(days)
        ro_time: 25(days)
```

Export the DS file of the array.edu.com host name.

```
AN(config)#sdns dnssec export ds array.edu.com "ftp://test:password@10.8.5.55/
dsset-edu.com."
```

Upload the DS file of the array.edu.com host name to an upper-level DNS server.

Note:

- To make the DNSSEC function to take effect, please ensure that the upper-level and root DNS servers support and have already enabled the DNSSEC function.
- The "path" of the FTP URL (ftp://user:password@host:port/path) in the "sdns dnssec import ds" command should be a relative path.

20.6 SDNS Configuration Backup

The SDNS configuration backup function supports backing up the currently running SDNS configuration on a local disk and backing up the SDNS configuration to an SCP server.

➢ **Back up the SDNS configuration to the local disk**

```
Demo(config)#sdns config write file abc
Demo(config)#show sdns config file
Running configuration backup files:
length        date/time                  name                        hmac-sm3
2336           Feb 03 2023 10:23:17 abc.cfg                  4089bc58207d8945ab84a77c886
495f7c1d0f214968f79699ccf11b9f061078c
Demo(config)#show sdns config file abc
Current configuration on disk
#Last configuration change at Fri Feb  3 10:23:17 2023 by array
#version Beta.APV.10.4.3.145 build on Mon Jan 30 16:15:11 2023
#smart DNS configuration
sdns on
sdns recursion off
sdns fulldns on
.....
Demo(config)#no sdns config file abc.cfg
Type "YES" to delete this configuration file: YES
Demo(config)#clear sdns config file
Type "YES" to delete all the configuration files: YES
10 file(s) removed
```

➢ **Back up the SDNS configuration to the SCP server**

```
Demo(config)#sdns config write scp "remote_server_name" "remote_user_name" abc
Password for remote_user_name@remote_server_name:
Warning: Permanently added 'remote_server_name' (ED25519) to the list of known hosts.
abc                                   100% 4919KB   1.6MB/s   00:03
```

20.7 SDNS Configuration Loading

The SDNS configuration loading function supports loading backup SDNS configurations from local disk and SCP servers. Administrators can use the command "**sdns config file**" to load the backup local SDNS configuration, and use the command "**sdns config scp**" to load the backup SDNS configuration on the SCP server.

- **Load the SDNS configuration from the local disk**

```
Demo(config)#sdns config file abc
```

- **Load the SDNS configuration from the SCP server**

```
Demo(config)#sdns config scp "remote_server_name" "remote_user_name" abc
Password for remote_user_name@remote_server_name:
Warning: Permanently added 'remote_server_name' (ED25519) to the list of known hosts.
abc                                   100% 4919KB    1.6MB/s    00:03
Type "Yes" to display this config before load:NO
Type "YES" to load this config:YES
```

21Deep Packet Inspection (DPI)

Deep Packet Inspection (DPI) is in comparison of the traditional analysis of only contents encapsulated in the fourth protocol layer and beneath of IP packets, including the source IP, destination IP, source port, destination port and the protocol type. In addition, an increasing number of network protocols do not use fixed ports for communications anymore, so protocol identification based on the port numbers of packets is no longer accurate. DPI can not only analyze the contents in the fourth protocol layer and beneath, but also achieves application layer analysis to identify the application protocol and contents of application data. Using the DPI function of the APV appliance, administrator can configure a dedicated LLB link for each type of application protocol. This will help prevent the quality deterioration of critical services caused by heavy load of specific applications.

21.2 Application Protocol Type

With DPI configured, the APV appliance is capable of identifying more than 200 types of predefined application protocols. Administrators can execute the "**show dpi protocol name predefine**" command to view all the predefined application protocol types supported by the DPI function.

Besides, the DPI function also supports the identification of custom application protocols that are self-defined by the IP address, host name and port number. Administrators can execute the "**show dpi protocol name custom**" command to view all the self-defined application protocols configured for the DPI function.

- Base on IP address: Application protocols are defined based on the source IP or destination IP segment. For example, traffic coming from 10.8.6.0/24 is defined as "application1"; traffic destined for 192.168.0.0/16 is defined as "application2".

```
AN(config)#dpi protocol rule ip application1 10.8.6.0 24
AN(config)#dpi protocol rule ip application2 192.168.0.0 16
```

- Based on hostname: Application protocols are defined based on the domain name of destination host. For example, traffic destined for the host "xyz.com" is defined as "application3".

```
AN(config)#dpi protocol rule host application3 xyz.com
```

- Based on port number: Application protocols are defined based on the port number of an application, for which the Layer 4 protocol can be UDP or TCP. For example, UDP traffic on port 8008 is defined as "application4"; TCP traffic on port 9009 is defined as "application5".

```
AN(config)#dpi protocol rule port application4 8008 udp
AN(config)#dpi protocol rule port application5 9009 tcp
```

21.3 Profile

After an application protocol is defined for DPI to identify, the administrator needs to manually import its profile (obtained from the service provider) and associate it with the application protocol.

Execute the following command to import the profile:

dpi import profile <profile_name> <url>

Execute the following command to associate it with the application protocol:

dpi attach profile <*application_name*> <*profile_name*>

For the Office 365 application, besides the manual import of the profile, the system also supports online download and updates of its profile from "https://support.content.office.net/en-us/static/O365IPAddresses.xml".

21.4 LLB Link

DPI can recognize the application protocol of received traffic and its content by application layer analysis, which lays a foundation for further routing based on application protocol types. The APV appliance supports configuration of independent LLB links for different application protocols recognized by DPI. This will help prevent the existing resources from being overused by application protocols running large-volume traffic, which will cause quality deterioration of existing services. Meanwhile, independent LLB links for such applications will help guarantee their service stability. Administrators can use the "**dpi apply link route**" command to associate the specified application protocol with a dedicated LLB link, then all traffic of this application will be forwarded through this LLB link.

For example, traffic of Office 365 will be forwarded only by LLB link "link1":

AN(config)#**dpi apply link route office365 link1**

22 Application Security

22.2 WebWall

22.2.2 Overview

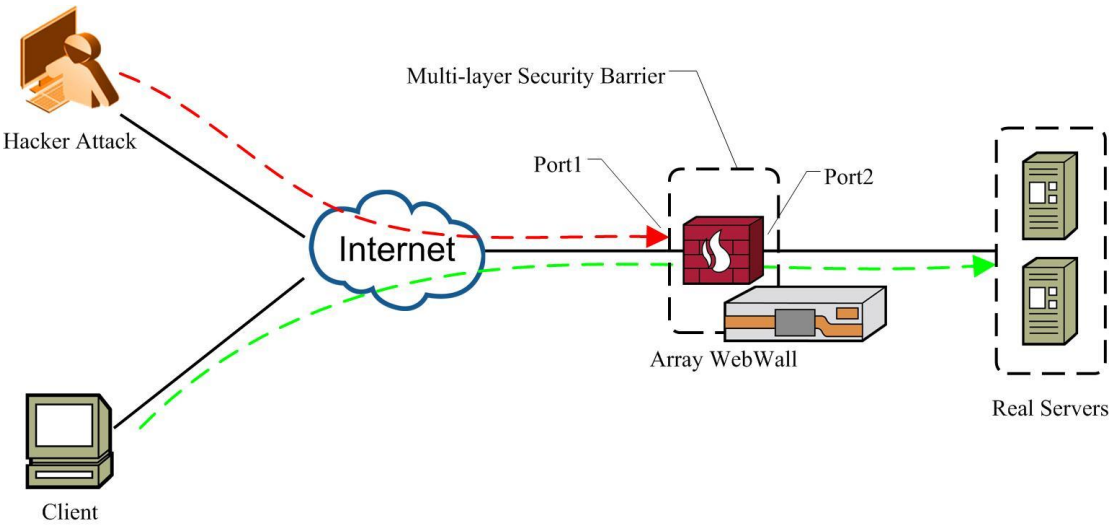
The WebWall functionality of the APV appliance allows you to create permit/deny rules to filter packets passing through your network infrastructure. The WebWall supports the filtering of TCP, UDP and ICMP packets that are using the IPv4 or IPv6 address. To use access lists you will define these “permit” and “deny” rules and apply them to access groups. Once the access lists are configured, you may apply or bind the group to an interface within the network.

The steps for basic WebWall configurations are explained in this section, along with some advanced features and general knowledge of how WebWall works. For ArrayOS, the WebWall feature can independently control each interface.

By default, WebWall permits SLB health check. For details, please refer to the “SLB Health Check” section in “Chapter 7 Server Load Balancing (SLB)”.

22.2.3 Understanding WebWall

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



22.2.3.1.1.1.1 WebWall

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

- ACL (Access Control List) filtering

- Protection against Syn-Flood, Fragmentation and DoS (Denial Of Service) attacks
- Stateful packet inspection
- Single packet attack prevention

ACL Filtering provides tight control over who may and may not enter the network by utilizing APV'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 ArrayOS capability.

In addition to ACL filtering, the WebWall provides stateful packet inspection and protects against Syn-Flood, fragmentation, DoS and single packet attacks.

The WebWall is a default-deny firewall. Default-Deny refers to the notion that if you do not have any permit rules in your access control lists, no packets will be allowed to pass through the appliance. During the initial installation of the box it might be helpful to leave the WebWall in the off or disengaged state until your total configuration is complete.

Note: By default the WebWall is turned off. The WebWall function will remain disabled until it is activated via the "**webwall on**" command.

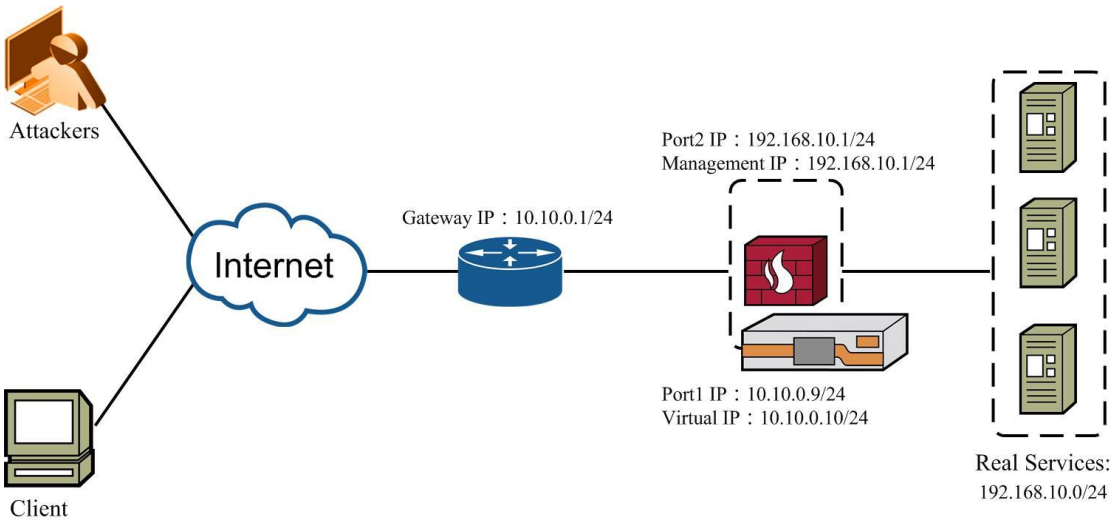
22.2.4WebWall Configuration

22.2.4.1 Configuration Scenario

Configure WebWall on the APV appliance to make sure it processes management traffic and service traffic based on the following rules:

- Permit the IP address 10.10.10.30 to configure and manage the APV appliance through port 22 (for SSH access).
- Permit the IP address 10.10.10.30 to configure and manage the APV appliance through port 8888 (for New WebUI access).
- Permit all IP addresses except 10.10.10.30 to access the virtual IP address 10.10.0.10.
- Permit all clients on the internal network to ping the IP address of port2 interface (192.168.10.1).

This configuration example is based on the network topology in the following figure.



22.2.4.1.1.1 WebWall Configuration

The following table lists the commands for configuring the WebWall. For detailed information of each command, please see the ArrayOS APV CLI Handbook.

22.2.4.1.1.2 General Settings of WebWall

Operation	Command
Configure access group	**accessgroup** *<accesslist_id> <interface>*
Configure ACL rules	**accesslist permit icmp echoreply** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist permit icmp echorequest** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist permit tcp** *<source_ip> {source_mask|source_prefix} <source_port> <destination_ip> {destination_mask|destination_prefix} <destination_port> <accesslist_id>* **accesslist permit udp** *<source_ip> {source_mask|source_prefix} <source_port> <destination_ip> {destination_mask|destination_prefix} <destination_port> <accesslist_id>* **accesslist permit esp** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist permit ah** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny icmp echoreply** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny icmp echorequest** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny tcp** *<source_ip> {source_mask|source_prefix}*

Operation	Command
	<source_port> <destination_ip> {destination_mask|destination_prefix} <destination_port> <accesslist_id> **accesslist deny udp** *<source_ip> {source_mask|source_prefix} <source_port> <destination_ip> {destination_mask|destination_prefix} <destination_port> <accesslist_id>* **accesslist deny esp** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>* **accesslist deny ah** *<source_ip> {source_mask|source_prefix} <destination_ip> {destination_mask|destination_prefix} <accesslist_id>*
Enable/Disable WebWall	**webwall** *<interface>* **on** *[mode]* **webwall** *<interface>* **off**
View WebWall configurations	**show interface** **show accesslist** **show accessgroup**

22.2.4.2 Configuration Steps

- **Configuring access groups**

1. Configure access group 50, which is used to associate ACL rules of miscellaneous types.

```
AN(config)#accessgroup 50 port2
```

Configure access group 100, which is used to associate ACL rules for management IP access.

```
AN(config)#accessgroup 100 port2
```

Configure access group 150, which is used to associate ACL rules for virtual IP access.

```
AN(config)#accessgroup 150 port1
```

- **Configuring ACL rules**

1. Permit the IP address 10.10.10.30 to configure and manage the APV appliance by using SSH at port 22

```
AN(config)#accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1 255.255.255.255 22 100
```

Permit the IP address 10.10.10.30 to configure and manage the APV appliance by using the new WebUI at port 8888.

```
AN(config)#accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1 255.255.255.255 8888 100
```

Permit all IP addresses except 10.10.10.30 to access the virtual IP address 10.10.0.10 through port 80.

```
AN(config)#accesslist permit tcp 0.0.0.0 0.0.0.0 0 10.10. 0.10 255.255.255.255 80 150
```

```
AN(config)#accesslist deny tcp 10.10.10.30 255.255.255.255 0 10.10. 0.10 255.255.255.255 0
150
```

Permit all IP address on the internal network to ping 192.168.10.1, IP address of the port2 interface.

```
AN(config)#accesslist permit icmp echorequest 192.168.10.0 255.255.255.255 192.168.10.1
255.255.255.255 50
AN(config)#accesslist permit icmp echoreply 192.168.10.1 255.255.255.255 192.168.10.0
255.255.255.0 50
```

- **Enabling the WebWall**

Execute the following command to enable the WebWall.

```
AN(config)#webwall port2 on
AN(config)#webwall port1 on
```

Before enabling the WebWall, please note that:

- If you have configured a DNS server and need to enable WebWall on the interface through which DNS traffic will pass, you need to add corresponding access rules to permit traffic on port 53.
- If you need to enable WebWall on the interface where the "**synconfig to**" or "**synconfig from**" command is applied, you need to manually add access rules to permit traffic on port 65,519; otherwise, configuration synchronization will fail. In an HA scenario, you can either configure the "**ha synconfig bootup on**" command on the local and peer units or manually add access rules to permit the synchronization data.

After WebWall is enabled, if you need to adjust configurations, please note that the SSH or WebUI session in use might be terminated due to misconfigurations.

- **Verification and Troubleshooting**

After finishing configuration of access rules and groups, you can use the "**show accesslist**" and "**show accessgroup**" commands to check the configurations.

```
AN(config)#show accesslist
accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1 255.255.255.255 22 100
accesslist permit tcp 10.10.10.30 255.255.255.255 0 192.168.10.1 255.255.255.255 8888 100
accesslist permit tcp 0.0.0.0 0.0.0.0 0 10.10. 0.10 255.255.255.255 80 150
accesslist deny tcp 10.10.10.30 255.255.255.255 0 10.10. 0.10 255.255.255.255 0 150
accesslist permit icmp echorequest 192.168.10.0 255.255.255.255 192.168.10.1 255.255.255.255
50
accesslist permit icmp echoreply 192.168.10.1 255.255.255.255 192.168.10.0 255.255.255.0 50
AN(config)#show accessgroup
accessgroup 50 port2
accessgroup 100 port2
accessgroup 150 port1
```

If you run into problems with access rules, keep your configurations simple. With multiple access groups, you can apply them once at a time to check which access rule is causing problems. You can turn the WebWall off to determine if the WebWall itself is indeed causing the problem.

To view the interfaces on which WebWall is enabled, run the "**show interface**" command. For example:

```
AN(config)#show webwall
webwall "port2" on 0
webwall "port3" on 0
```

To view the packet drop statistics on the webwall, run the "**show interface**" command. See italics in the following example.

```
AN(config)#show interface
port2(port2): flags=2008842<BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:25:90:39:97:f1
        media: autoselect
        status: no carrier
        webwall status: ON
        Hardware is i82574l
        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（Number of packets dropped by default denial rules）
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        packet drop (deny): 0（Number of packets dropped by configured denial rules）
                tcp 0           udp 0           icmp 0          ah 0            esp 0
        5 minute input rate 0 bits/sec, 0 packets/sec
        5 minute output rate 0 bits/sec, 0 packets/sec
```

22.3 Web Application Firewall (WAF)

22.3.2Overview

Web Application Firewall (WAF) can provide real-time security protection for the Web sites and Web application systems. The WAF function can defend against the common threats and malicious attacks, such as Structured Query Language (SQL) injection.

The WAF function of APV appliance provides the following features:

- HTTP/HTTPS traffic detection: The WAF function is capable of parsing the HTTP protocol completely, including the headers, parameters and payload. Besides, the WAF function can cooperate with the SSL acceleration function to parse the HTTPS protocol.
- Rule engine: The WAF function provides a set of rules that is easy-to-use and flexible. The administrator can use the predefined ruleset to recognize the common Web attack, or use the custom ruleset to recognize and prevent other attacks. Furthermore, the WAF function is compatible with the open source intrusion detection and protection engine “ModSecurity”. The administrator can use the ModSecurity Core Rule Set (CRS) in the WAF function.
- Virtual patching: The virtual patching function of the WAF function can protect the Web application against attacks by fixing the flaws of Web application on the APV appliance, which is more convenient and faster compared with fixing such flows on the Web application flaws directly.
- Traffic logging: The WAF function can monitor the HTTP and HTTPS traffic in real time, and record the entire event transaction log. Furthermore, the WAF function can cooperate with the third-party remote audit log server and provide the statistical analysis and security event report using the third- party remote audit log server.
- Cooperative with other functions: The WAF function can work together with functions such as SLB, Cache, Compression and SSL acceleration.
- Flexible deployment: The WAF function can be deployed in transparent mode and reverse proxy mode.

22.3.3Basic Concepts

In the WAF function, virtual services can be associated with **identical** or different security profiles. A security profile includes one or more rulesets and a ruleset includes one or more rules.

22.3.3.1 Rule

A rule is the basic directive of the WAF function. The WAF function analyzes the rules and takes corresponding actions based on the matching result of the HTTP and HTTPS traffic and rules, hence protecting the system against possible Web attacks.

22.3.3.1.1 Rule Syntax

The syntax of a rule is: SecRule Variable Operator Action. The following figure provides an example of the rule:

Figure 22-1 Rule Example

The meanings of elements of a rule are as follows:

- SecRule: Creates a rule that uses a specified operator to check the variable and performs the action when the variable matches the operator.
- Variable: Specifies the variable name to be checked. Each rule can contain one or more variables. The variable value can be any HTTP parameter. For example, in the figure above, "REQUEST_URI" is a variable name and specifies the request body of the variable to be checked.
- Operator: Describes how to check the variable. The operator uses regular expressions. The WAF function also provides multiple available operators. "@" is used to specify which operator is used. For example, in the figure above, "@beginsWith abc" is an operator and specifies the URL address that begins with "abc".
- Action: Indicates which action will be performed when the variable matches the operator. For example, "phase:2,auditlog,deny,msg:'Failed to parse request body.'" is an action of the rule. The action specifies that the system will perform the rule in the "phase:2", block the request, display the error message "Failed to parse request body." on the browser of the client, and record the contents of the request in the audit log.

The WAF function extracts the related information from the Request Headers, Request Body, Response Headers or Response Body of HTTP and HTTPS packets and takes actions defined in the rule. The corresponding phases are phase:1, phase:2, phase:3 and phase:4.

For details of the rules of WAF function and the related phases of the rules, please contact Customer Support.

22.3.3.1.2 Action

The action is the behavior that the rule will take on the HTTP and HTTPS traffic matched with the rule.

The actions of the rules include:

- deny: When the attack is detected, the system will send the error page to the client to indicate that the request is blocked and record the log.
- pass: When the attack is detected, the system will only record the log.

- block: When the attack is detected, the system will take the default action of the rule, which is the action of ruleset where the rule belongs. For details about default action of the rule, please refer to the section “Ruleset”.

22.3.3.2 Ruleset

The ruleset includes one or more rules. The WAF function matches the HTTP and HTTPS traffic with every rule in the ruleset and takes corresponding action on the matched traffic. The type of a ruleset can be either predefined or custom. The predefined ruleset includes SQL injection and Cross Site Scripting (XSS). The administrator can use the custom ruleset to define custom rules to prevent the system against attacks. In addition, the administrator can import the custom ruleset via the CLI interface or the WebUI interface, or export the ruleset via WebUI interface flexibly.

Rulesets are implemented in the descending order of priorities. For predefined rulesets and custom rulesets with an identical priority, the execution of the rulesets will be at random.

A ruleset defines the default action for the rules. The actions of the rulesets include “deny” and “pass”. All the actions of rules are defined as “block” in the predefined ruleset. And the system will take the action of the predefined ruleset. For the custom ruleset, the action of a rule can be “deny”, “pass”, “block” and so on. For the rule whose action is “deny” or “pass”, the system will take the action of the rule; for the rule whose action is “block”, the system will take the action of the custom ruleset.

Note:

If the directive <Location> or <LocationMatch> is configured and the action of the custom rule corresponding to the directive is configured as “block”, the “pass” action will be executed regardless of the action of the custom ruleset.

22.3.3.3 Security Profile

A security profile includes one or more rulesets. By adding multiple rulesets to the security profile and associating the security profile with the virtual service, the administrator can configure security profile for the virtual service without associating every ruleset with the virtual service which provides simpler and easier configuration methods. In addition, the administrator can flexibly control the processing mode on detected system attacks by changing the working mode of the security profile without changing the action of each ruleset. For the details of working modes, please refer to the section “Working Mode”.

22.3.3.4 Working Mode

The working modes of the security profile include:

- Passive mode: When the attack is detected, the system will record the log, but will not block the traffic or send the error page to the client.

- Active mode: When the attack is detected, the system will record the log, block the traffic and send the error page to the client.

22.3.3.5 Logging

The existing logging function of the APV appliance can record all information about client access. The security level of the log is “warning”. Besides, the WAF function provides the audit log function.

Audit logs will record the HTTP and HTTPS traffic. Audit logs can be stored on the local disk or sent to the third-party remote audit log server. For how to install the third-party remote audit log server, please contact Customer Support.

Note:

Audit logs cannot be stored on both the local disk and the third-party remote audit log server.

22.3.4Configuration Example

22.3.4.1 Configuration Purpose

Add the custom ruleset to protect the system against the CSRF attack, add the ruleset to the security profile and associate the security profile with the virtual service, so that the APV appliance can protect the virtual service against CSRF attack.

Prerequisites:

The Layer 7 virtual service vs1 is already defined. For details of the configuration of virtual service, please refer to Chapter 7 Server Load Balancing (SLB).

The custom ruleset s1000 is already defined.

22.3.4.2 Configuration Example

1. Add a security profile and specify the working mode.

For example:

```
AN(config)#waf profile name p1
AN(config)#waf profile mode active p1
```

2. Import the custom ruleset for defending against the CSRF attack.

For example:

```
AN(config)#waf import " s1000" http://192.168.10.20/c1.txt ruleset
```

3. Add the custom ruleset for defending against the CSRF attack to the security profile.

For example:

```
AN(config)#waf profile custom "p1" " s1000" deny 1000
```

4. Associate virtual service vs1 with security profile p1.

For example:

```
AN(config)#waf profile assign p1 vs1
```

5. Enable the audit log.

For example:

```
AN(config)#waf log audit on
```

22.4 Advanced ACL

22.4.2Overview

The system supports the configuration of advanced Access Control List (ACL) to limit network traffic and control clients' access behavior. This helps protect against attackers that maliciously launch large volume of access traffic so as to improve the availability of internal resources. This function is implemented by access control rules that impose checks on the source IP address, protocol and host name of packets. It determines whether to permit a packet or take other actions when the packet matches the conditions speculated in an access control rule.

The advanced ACL function of the APV appliance supports three types of access control rules, ACL rules, HTTP ACL rules and DNS ACL rules, all of which are applicable in both IPv4 and IPv6 environments.

- The ACL rule restricts the number of connections per second (CPS) and concurrent connections (CC). It applies to virtual services that are running over TCP.
- The HTTP ACL rule restricts the number of requests per second (RPS) and the download rate. It applies to only HTTP and HTTPS virtual services.
- The DNS ACL rule restricts the number of requests per second (RPS). It applies to only DNS virtual services and SDNS.

Note:

- ACL rules, HTTP ACL rules and DNS ACL rules may impact the system performance.
- ACL rules, HTTP ACL rules and DNS ACL rules take effect only for connections established after the rules are configured.
- When accessing HTTP or HTTPS virtual services that use the insert cookie policy, the client requests that carry cookies inserted by the appliance are not restricted by

ACL rules and HTTP ACL rules.

22.4.3ACL Rule

The ACL rule supports two control modes and two control types.

➢ **Control modes**

- "total" mode indicates that all the clients on a subnet will be restricted by the ACL rule as a whole.
- "per-ip" mode indicates that every client on a subnet will be restricted by the ACL rule individually.

➢ **Control types**

- "cps" indicates restriction on the CPS.
- "concurrent" indicates restriction on the CC.

The administrator needs to combine the control modes and control types as needed for configuring ACL rules for a subnet. There are four combinations of ACL rules for the same subnet. See the following table.

22.4.3.1.1.1 Four Rules for the Same Subnet

Mode \ Type	cps	concurrent
total	The total CPS of all the clients on the subnet cannot exceed the specified maximum value.	The total CC of all the clients on the subnet cannot exceed the specified maximum value.
per-ip	The CPS of every client on the subnet cannot exceed the specified maximum value.	The CC of every client on the subnet cannot exceed the specified maximum value.

One or more rules of the preceding combinations are allowed for a subnet on the APV appliance. If multiple rules have been configured for a subnet, these rules will work at the same time.

➢ **Subnet Nesting**

The ACL rule supports subnet nesting. If the IP address of a client hits multiple subnets, the client is subject to the restriction of all the ACL rules of the smallest subnet hit, and of the "total" rules of all the upper subnets of the smallest subnet. The system supports 3-layer subnet nesting at most.

For example:

- 10.8.0.0/16: Both "total" and "per-ip" rules are configured.
- 10.8.0.0/24: Both "total" and "per-ip" rules are configured.

  Then, clients on the subnet of 10.8.1.0/24 will be subject to the restriction of:

- 10.8.0.0/24: “total” and “per-ip”
- 10.8.1.0/16: “total” rules

➢ **ACL Whitelist**

The administrator can configure ACL whitelists to make the clients on the whitelists free from restriction of ACL rules. When both ACL rules and whitelists are applied to the same virtual service, the whitelists have higher processing priorities than the ACL rules.

If the IP address of a client belongs to the subnet defined in an ACL whitelist, the client will not be subject to the restriction of any ACL rule. However, the CPS and CC of the client will be counted in the total CPS and CC of the subnet it belongs to respectively.

For example:

- Rule: 10.8.1.0/24; total concurrent≤10,000
- Whitelist: 10.8.1.10/32

  If the CC on the client whose IP address is 10.8.1.10 is 1000,

  Then, the maximum total CC of the subnet 10.8.1.0/24 (excluding 10.8.1.10) is 9000.

➢ **Application Scope**

ACL rules and ACL whitelists can be applied not only to an individual virtual service, but also to virtual services globally. The advanced ACL function supports all TCP-based virtual services.

When accessing a specified virtual service, a client is subject to the restriction of not only the ACL rules applied to this virtual service individually but also the ACL rules applied globally.

For example:

- rule1: 0.0.0.0/0, per-ip concurrent≤1000
- rule2: 0.0.0.0/0, per-ip concurrent≤900
- rule1 is applied to vs1; rule2 is applied globally.

  Then, when the client whose IP address is 10.8.1.10 accesses the virtual service “vs1”, the allowed maximum CC is 900.

➢ **Application of ACL Rules to HTTP-based Virtual Services**

For Layer 7 HTTP and HTTPS virtual services with the insert cookie policy used, the APV appliance will insert cookies in the responses sent to the clients. The APV appliance can identify these clients with the inserted cookies. When a client accesses the virtual service again and carries the cookie inserted by the APV appliance in the requests, the client will not be subject to the restriction of any ACL rule. However, the CPS and CC of the client will be counted in the total CPS and CC of the subnet it belongs to respectively.

22.4.4HTTP ACL Rule

HTTP ACL rules impose restrictions on the RPS and download rate of clients accessing HTTP and HTTPS virtual services. When an HTTTP ACL rule restricting RPS is triggered, the system will exercise control over clients' access or take specific approaches. The HTTP ACL rule that restricts the download rate will control the download rate of clients within a specific range.

22.4.4.1 Rule Type

The system supports static and dynamic HTTP ACL rules. The static HTTP ACL rules are manually configured via CLI or WebUI, and dynamic HTTP ACL rules are automatically generated by the system's security module based on judgment of whether the system is under an attack.

- **Static HTTP ACL Rule**

This type of HTTP ACL rules allows the administrator to customize control rules on HTTP traffic. Administrators can use the "acl http rule" command to configure static HTTP ACL rules, and then the "**acl http apply rule virtual**" command to apply it to an HTTP or HTTPS virtual service.

acl http rule *<rule_name> <client_ip> {netmask|prefix} <acl_type> <max_limit> <condition>*
acl http apply rule virtual *<rule_name> <virtual_service>*

- **Dynamic HTTP ACL Rule**

The system will generate dynamic HTTP ACL rules after the system DDoS attack defense function is enabled (by the "**ddos on**" command). Dynamic HTTP ACL rules are an enhancement to the traditional HTTP ACL rules (static HTTP ACL rule). Besides static HTTP ACL rules, when the system detect an HTTP DDoS attack, if HTTP verification is not enabled, the system will generate a dynamic HTTP ACL rule for a suspicious client to restrict its access to a virtual service. This rule specifies the RPS threshold as 100 and the download rate threshold as 1024 KB/s. If the RPS of the client that matches this rule reaches 100, subsequent requests from this client will be dropped, until its RPS decreases to a value smaller than 100 again. At the same time, the download rate of the client that matches this rule will be restricted within 1024 KB/s. If HTTP verification is enabled when the system detects an HTTP DDoS attack, the IP address of the client that has been confirmed as illegitimate will be recorded in the DDoS blacklist.

Administrators can use the "**show acl http rule**" command to view both static HTTP ACL rule configurations and dynamic HTTP ACL rules that the system generates.

Note:

- Static HTTP ACL rules take precedence over dynamic HTTP ACL rules. When both types are triggered, the system preferentially executes the static one.
- HTTP ACL rules take effect only after being applied to a virtual service.

22.4.4.2 Control Type and Processing Mode

HTTP ACL rules support two types of control modes: RPS and download rate.

1. RPS control

The RPS-based HTTP ACL rule calculates the number of HTTP requests initiated by a client or all clients on a subnet per second. When the number reaches the specified threshold, the system will take actions on subsequent requests as speculated in the HTTP ACL rule. In this way, it prevents servers from being overloaded by large number of HTTP requests and improves the availability of servers.

Matching Condition

The system calculates the RPS based on the matching conditions speculated in the HTTP ACL rule. When an HTTP request meets all of the following conditions, it is included in the RPS count of an HTTP ACL rule:

- The client' IP address belongs to the subnet specified by the HTTP ACL rule.
- The requested URL matches the URL regular expression in the HTTP ACL rule.
- The HTTP access method matches the HTTP method specified in the HTTP ACL rule. The system supports the following HTTP methods for filtering HTTP requests:
  - All: indicates all HTTP methods, that is, all HTTP requests are included in the RPS count.
  - GET, POST, HEAD, DELETE, or PUT: indicates that only HTTP requests using the specified HTTP method will be included in the RPS count.

Processing Mode

For clients whose RPS exceeds the specified limit of an HTTP ACL rule, the system will take actions on subsequent requests that also match the rule. It supports two processing modes:

- Returning an error page: By default, the appliance returns a built-in 480 error page. Administrators can also customize the format and content of the returned error page by importing or redirecting the error page.
  - Error page import: Configure a customized 480 error page using the "**http import error**" and "**http error**" commands. The system will return the customized 480 error page that is imported.
  - Error page redirect: Configure a response redirect rule using the "**http redirect error**" command. The system will return a redirect response, and the client will be redirected to the URL address in the response.
  - If both a customized 480 error page and a response redirect rule exist, the response redirect rule takes effect.
- Resetting the connection by sending an RST packet

Example:

AN(config)#**acl http rule rule1 10.8.6.0 24 rps 1000 "url=<regex>abc* method=GET action=errorpage"**

In this example, when a client in subnet 10.8.6.0/24 accesses a URL address that matches "abc*", the number of HTTP GET requests initiated by the client per second is restricted to be 1000 at most. Upon receiving a subsequent request, the system will return an in-built 480 error page.

Download rate control

The download rate-based HTTP ACL rule restricts the download rate of a client or every client on a subnet below the threshold specified by the rule.

Matching Condition

The system detects the download rate of clients based on the matching conditions speculated in the HTTP ACL rule. When an HTTP request meets all of the following conditions, the initiating client's download rate will be restricted by the HTTP ACL rule:

- The client's IP address belongs to the subnet specified by the HTTP ACL rule.
- The requested URL matches the URL regular expression specified in the HTTP ACL rule.

Example:

AN(config)#**acl http rule rule1 10.8.6.0 24 throughput 1000 "url=<regex>abc*"**

In this example, when a client in subnet 10.8.6.0/24 accesses a URL address that matches "abc*", the client's download speed cannot exceed 1000 KB/s.

22.4.4.3 Subnet Nesting

HTTP ACL rules allow at most 3-layer subnet nesting.

If a client's IP address hits multiple subnets, and the request matches multiple HTTP ACL rules for different subnets, this client is subject to the restriction of the HTTP ACL rule for the smallest subnet.

For example, a client's IP address is 10.8.6.11. If the client request matches all of the following rules, this client is subject only to the restriction of the HTTP ACL rule for subnet 10.8.6.0/24:

- HTTP ACL rule for subnet 10.0.0.0/8
- HTTP ACL rule for subnet 10.8.0.0/16
- HTTP ACL rule for subnet 10.8.6.0/24

If the client request does not match any HTTP ACL rule for subnet 10.8.6.0/24, the client is subject only to the restriction of the HTTP ACL rule for subnet 10.8.0.0/16.

22.4.5DNS ACL Rule

DNS ACL rules are used to control the number DNS queries per second. They can be used in both the DNS SLB and SDNS application scenarios. A DNS ACL rule will specify an RPS threshold. When the number DNS queries per second reaches this threshold, subsequent DNS queries will be dropped.

In the DNS SLB application scenario, the DNS ACL rule imposes control over the number of DNS queries received on a DNS virtual service.

In the SDNS application scenario, the DNS ACL rule imposes control over the number of DNS queries received on the BIND9 DNS server.

22.4.5.1 Rule Type

The system supports static and dynamic DNS ACL rules. The static DNS ACL rules are manually configured via CLI or WebUI, and dynamic DNS ACL rules are generated by the system.

- **Static DNS ACL Rule**

This type of DNS ACL rules allows the administrator to customize control rules on DNS traffic. Administrators can use the "**acl dns rule**" command to configure static DNS ACL rules, and then the "**acl dns apply rule virtual**" or "**acl dns apply rule sdns**" command to apply it to a DNS virtual service or SDNS.

acl dns rule <*rule_name*> <*client_ip*> {*netmask*|*prefix*} <*acl_mode*> <*dns_type*> <*max_limit*>
acl dns apply rule virtual <rule_name> <virtual_service>
acl dns apply rule sdns <rule_name>

- **Dynamic DNS ACL Rule**

In initial setup, the sysem will generate dynamic ACL rules after the DNS DDoS attack defense function is enabled (by the "**ddos dns on**" command). Dynamic DNS ACL rules are an enhancement to the traditional DNS ACL rules (static DNS ACL rule). During the runtime of a network, traffic load fluctuates with clients' access behavior. However, the traditional DNS ACL rules cannot dynamically control network traffic as the volume of network traffic changes.

To provide a type of DNS ACL rules that impose dynamic control over the fluctuating network traffic and as an effort to simplify administrators' configuration load, the system supports dynamic DNS ACL rules. This type of rules is dynamically generated by the system based on the actual traffic load on the network. The system will continuously learn and calculate the DNS traffic load on the network during runtime to generate dynamic DNS ACL rules on each virtual service, regardless of whether static DNS ACL rules are configured.

Administrators can use the "**show acl dns rule**" command to view both static DNS ACL rule configurations and dynamic DNS ACL rules that the system generates.

Note:

- Static DNS ACL rules take precedence over dynamic DNS ACL rules. When both types are triggered, the system preferentially executes the static one.
- DNS ACL rules take effect only after being applied to a virtual service or SDNS.

22.4.5.2 Control Mode

The control mode specifies the range that a DNS ACL rule takes effect on a specific subnet. It supports two modes:

- "total": indicates that all the clients on a subnet will be restricted by the DNS ACL rule as a whole.
- "per-ip": indicates that every client on a subnet will be restricted by the DNS ACL rule individually.

Matching Condition

The system calculates the RPS based on the matching conditions speculated in the DNS ACL rule. When a DNS query meets all of the following conditions, it is included in the RPS count of a DNS ACL rule:

- The client' IP address belongs to the subnet specified by the DNS ACL rule.
- The DNS query matches the DNS resource record type specified in the DNS ACL rule. The system supports the following DNS resource record types for filtering DNS queries:
  - All: indicates all types of DNS queries are included in the RPS count.
  - A, NS, MD, MF, CNAME, SOA, MB, MG, MR, NULL, WKS, PTR, HINFO, MIFO, MX, TXT, AAAA, OTHERS or ANY: indicates that only the specified type of DNS queries are included in the RPS count.

Example:

```
acl dns rule dnsrule1 192.168.0.0 16 per-ip AAAA 100
```

In this example, when the number of AAAA DNS queries initiated by a client in subnet 192.168.0.0/24 per second reaches 100, the system will drop subsequent AAA DNS queries coming from this client.

22.4.5.3 Subnet Nesting

DNS ACL rules allow at most 3-layer subnet nesting.

If a client's IP address hits multiple subnets, and the request matches multiple DNS ACL rules for different subnets, this client is subject to the restriction of the DNS ACL rule for the smallest subnet and of the "total" rules of all the upper subnets of the smallest subnet.

For example, a client's IP address is 10.8.6.11. If the client request matches all of the following rules (all are per-ip based), this client is subject only to the restriction of the DNS ACL rule for subnet 10.8.6.0/24:

- DNS ACL rule for subnet 10.0.0.0/8
- DNS ACL rule for subnet 10.8.0.0/16
- DNS ACL rule for subnet 10.8.6.0/24

If the client request does not match any DNS ACL rule for subnet 10.8.6.0/24, the client is subject only to the restriction of the DNS ACL rule for subnet 10.8.0.0/16.

22.4.6Advanced ACL Configuration

22.4.6.1 Configuring ACL Rules

➢ **Configuration Steps**

To configure ACL rules, you need to first add ACL rules and then apply these rules to a specified virtual service or apply them globally.

1. Execute the following command to add ACL rules:

acl rule <*rule_name*> <*client_ip*> {*netmask*|*prefix*} <*acl_mode*> <*acl_type*> <*max_limit*>

For example:

AN(config)#**acl rule rule1 61.130.10.0 255.255.255.0 total cps 1000000** AN(config)#**acl rule rule2 61.130.10.0 255.255.255.0 total concurrent 50000** AN(config)#**acl rule rule3 61.130.10.0 255.255.255.0 per-ip cps 10000** AN(config)#**acl rule rule4 61.130.10.0 255.255.255.0 per-ip concurrent 500**

Execute the following command to apply ACL rules to virtual services:

acl apply rule virtual <*rule_name*> <*virtual_service*>

For example:

AN(config)#**acl apply rule virtual rule1 tcp_vs1** AN(config)#**acl apply rule virtual rule3 tcp_vs1** AN(config)#**acl apply rule virtual rule2 http_vs1** AN(config)#**acl apply rule virtual rule4 http_vs1**

➢ **Configuring ACL Whitelists**

To configure ACL whitelists, you need to first add ACL whitelists and then apply these whitelists to a specified virtual service or apply them globally. ACL whitelist configuration is optional.

1. Execute the following command to add ACL whitelists:

acl whitelist <*whitelist_name*> <*client_ip*> {*netmask*|*prefix*}

For example:

```
AN(config)#acl whitelist whitelist1 61.130.10.10 255.255.255.255
```

Execute the following command to apply ACL whitelists to virtual services:

acl apply whitelist virtual <*whitelist_name*> <*virtual_service*>

For example:

```
AN(config)#acl apply whitelist virtual whitelist1 tcp_vs1
AN(config)#acl apply whitelist virtual whitelist1 http_vs1
```

➢ **Configuration Results**

After the preceding configurations are completed, the APV appliance will:

When clients are accessing the virtual service "tcp_vs1":

- Restrict the maximum total CPS of all clients on the subnet 61.130.10.0/24 to 1,000,000.
- Restrict the maximum CPS of every client on the subnet 61.130.10.0/24 to 10,000.
- Not restrict the maximum CPS of the client whose IP address is 61.130.10.10.

When clients are accessing the virtual service "http_vs1":

- Restrict the maximum total CC of all clients on the subnet 61.130.10.0/24 to 50,000.
- Restrict the maximum CC of every client on the subnet 61.130.10.0/24 to 500.
- Not restrict the maximum CC of the client whose IP address is 61.130.10.10.

22.4.6.2 Configuring HTTP ACL Rules

➢ **Configuration Steps**

1. Execute the following command to add HTTP ACL rules:

```
acl http rule <rule_name> <client_ip> {netmask|prefix} <acl_type> <max_limit>
<condition>
```

For example:

```
AN(config)#acl http rule rule1 10.8.6.0 24 rps 1000 "url=<regex>abc* method=ALL
action=reset"
AN(config)#acl http rule rule2 192.168.0.0 16 throughput 1000 "url=<regex>xyz*"
AN(config)#acl http rule rule3 192.168.1.0 24 throughput 100 "url=<regex>xyz*"
```

Execute the following command to apply HTTP ACL rules to virtual services:

```
acl http apply rule virtual <rule_name> <virtual_service>
```

For example:

```
AN(config)#acl http apply rule virtual rule1 httpvs1
AN(config)#acl http apply rule virtual rule2 httpvs2
AN(config)#acl http apply rule virtual rule3 httpvs2
```

➢ **Configuration Results**

- When clients on subnet 10.8.6.0/24 access the URL address matching "abc*" via virtual service "httpvs1", if the RPS of any client reaches 1000, the system will reset the connection when receiving subsequent requests from the client.
- When clients on subnet 192.168.0.0/16 access the URL address matching "xyz*" via virtual service "httpvs2", the download rate of each client cannot exceed 1000 KB/s.
- When clients on subnet 192.168.1.0/24 access the URL address matching "xyz*" via virtual service "httpvs2", they are subject only to the restriction of "rule3" though this subnet hits both rule2 and rule3. Therefore, the download rate of each client cannot exceed 100 KB/s.

22.4.6.3 Configuring DNS ACL Rules

➢ **Configuration Steps**

1. Execute the following command to add DNS ACL rules:

acl dns rule <*rule_name*> <*client_ip*> {*netmask*|*prefix*} <*acl_mode*> <*dns_type*> <*max_limit*>

For example:

```
AN(config)#acl dns rule rule1 10.8.6.0 24 total ALL 12500
AN(config)#acl dns rule rule2 192.168.0.0 16 per-ip A 200
```

Execute the following command to apply DNS ACL rules to virtual services:

acl dns apply rule virtual <*rule_name*> <*virtual_service*>

For example:

```
AN(config)#acl http apply rule virtual rule1 dnsvs1
AN(config)#acl http apply rule virtual rule2 dnsvs2
```

Execute the following command to apply DNS ACL rules to SDNS:

acl dns apply rule sdns <*rule_name*>

For example:

```
AN(config)#acl dns apply rule sdns ru1e1
AN(config)#acl dns apply rule sdns ru1e2
```

➢ **Configuration Results**

In the DNS SLB application scenario:

- When clients on subnet 10.8.6.0/24 requests DNS resource records via virtual service “dnsvs1”, if the total number of DNS queries initiated by all clients in this subnet reaches 12,500, the system will drop subsequent DNS queries from this subnet.
- When clients on subnet 192.168.0.0/16 requests DNS resource records via virtual service “dnsvs2”, if the number of A-type DNS queries initiated by any client reaches 200, the system will drop subsequent A-type DNS queries from this client.

In the SDNS application scenario:

- If the total number of DNS queries initiated by all clients in this subnet reaches 12,500, the system will drop subsequent DNS queries from this subnet.
- If the number of A-type DNS queries initiated by any client on subnet 192.168.0.0/16 reaches 200, the system will drop subsequent A-type DNS queries from this client.

22.5 DDoS Attack Defense

The Distributed Denial of Service (DDoS) attack has become an increasingly frequent attack mode on the Internet. It is characteristics of multiple sources launching attacks on the same target at the same time, resulting in service unavailability. There are mainly the following root causes that lead to the service unavailability:

- Exhaust of system resources (bandwidth, memory and computing) to cause a device failure, such as the UDP Flood and SYN Flood attacks
- Disruption of network protocol or status to cause device malfunction, such as IP fragment attack and Ping of Death attack
- Exploitation of application errors to cause system abnormality or disclosure of critical information, such as DNS poisoning

Based on the network layer where DDoS attacks occur, the DDoS attacks can be classified as network layer attack, session layer attack and application layer attack. Two tendencies have emerged in DDoS attacks. On the one hand, the attack traffic is dramatically increasing. On the other hand, application-layer attacks burst and are increasingly complicated, which is hard to detect and mitigate.

The DDoS attack defense function of the APV appliance provides attack detection and defense mechanisms against DDoS attacks on the three network layers above. It implements an extra layer of protection for SLB services. In the initial setup, DDoS attack defense is disabled. Administrators can use the “**ddos {on|off}**” command to globally enable or disable the system DDoS attack defense function.

22.5.2Network Layer

This section lists DDoS attacks that frequently occur on the network layer and the defense mechanisms provided by the APV appliance.

22.5.2.1 IP DDoS

- **IP option attack**

Symptom: IP options are used for debugging or for some special applications. Attackers usually use IP options for source routing, which employs the trusted client's IP address to attack the server. An IP header can contain multiple IP options. These options can have a flexible combination in the type, length and order. Some new systems and protocols may encounter fatal errors when processing these options.

Defense: The system does not process IP options by default. It will strip all IP options from received packets.

22.5.2.2 ICMP DDoS

- **ICMP Flood**

Symptom: This type of attack overloads the target host with too much ICMP traffic. The host will be unable to serve normal requests. Finally, the network is crashed.

Defense: The system filters out ICMP traffic to invalidate the attack via the underlayer operating system.

- **Ping of Death**

Symptom: In Ping of Death attacks, the server will receive IP packets whose length is longer than 65,535 bytes. Some systems cannot process such long packets, and may malfunction or even reboot.

Defense: The system will drop all overlong IP packets.

- **Smurf**

Symptom: Smurf attackers use the target server's IP address to send ICMP requests to a broadcast address. When all hosts in the broadcast subnet reply ICMP responses to the target server, the network is crashed.

Defense: The system will drop ICMP requests that are sent to a broadcast address.

22.5.2.3 TCP DDoS

In initial setup, TCP DDoS attack defense takes effect when the system DDoS attack defense is globally enabled. Administrators can use the "**ddos tcp {on|off}**" command to enable or disable TCP DDoS attack defense.

After TCP DDoS attack defense is enabled, the system will detect and defend against the following types of TCP DDoS attacks.

- **SYN Flood**

Symptom: By using faked IP addresses, attackers repeatedly send SYN packets to each port of the target server. The server repeatedly replies SYN-ACK packets on each port, and finally encounters failures due to exhausted memory or CPU resources.

Defense: The system will check the validity of SYN packets, drop the attack packets and file attack records.

- **PUSH/ACK Flood**

Symptom: ACK and PUSH-ACK packets are used to confirm the receipt of packets coming from the other communication party. In an ACK Flood, the target server will receive a large number of faked ACK packets. Its resources will be exhausted because of processing these packets. Finally, the service quality decreases and even is interrupted.

Defense: The system will detect the suspicious source and file attack records.

- **FIN/RST Flood**

Symptom: The attackers send faked RST or FIN packets at an extremely high rate to the target server. These packets do not belong to any TCP session, but the server commit a lot of resources in looking up the matched TCP sessions. If the RST or FIN packet accidentally matches an existing TCP session, the session will be disconnected.

Defense: The system will detect the suspicious source and file attack records.

- **Connection Flood**

Symptom: This type of attack features the long-time opening of massive connections to the target server by using real IP addresses. These idle connections exhaust the server's connection resources, so it cannot serve normal connection requests.

Defense: The system will decrease the connection timeout value, detect the suspicious sources and file attack records.

22.5.2.4 UDP DDoS

UDP DDoS attack defense protects against UDP Flood. In initial setup, UDP DDoS attack defense will take effect after the system DDoS attack defense function is enabled. The administrator can use the "**ddos udp {on|off}**" command to enable or disable UDP DDoS attack defense.

- **UDP Flood**

Symptom: UDP Flood is a type of resource exhaustion attack. As UDP is stateless, attacker will send a great number of UDP packets to the random ports of the target server. The server repeatedly looks up the applications listening on these ports. As no application is matched, the server needs to return massive ICMP destination unreachable packets. Finally, network bandwidths in both directions are quickly used up. Moreover, link congestion occurs.

Defense: The system will detect the suspicious source and file attack records.

22.5.3Session Layer

This section lists DDoS attacks frequently encountered on the session layer and the defense mechanisms that the APV appliance provides.

- **SSL Anomalies**

Symptom: Attackers initiates massive SSL packets that are malformed or irresolvable, resulting in system process crashes and service unavailability.

Defense: The system will filter out and drop invalid packets using the SSL state machine.

- **SSL Handshake Timeout**

Symptom: Attackers request to establish a large number of TCP connections but do not send SSL handshake messages or application data after TCP connections are established, consuming up SSL connection resources on the server.

Defense: The system has a default SSL handshake timeout mechanism. In initial startup, the system will check the duration of each SSL handshake based on the default timeout value (10,000 ms). If the system does not receive SSL handshake messages or application data after the connection has been established for 10,000 ms, it will reset the connection and file attack records.

As the actual traffic fluctuates, the system will generate a dynamic timeout value, and check the duration of SSL handshake based on the dynamically generated value. Administrators can also use the "**ddos ssl handshake timeout**" command to define a timeout value. When a self-defined timeout value exists, the dynamically generated timeout value does not take effect.

- **SSL Renegotiation**

Symptom: Attackers frequently send SSL renegotiation requests to trigger new renegotiation handshakes after the SSL connection is established. Therefore, one TCP connection will carry multiple SSL handshake sessions, which consumes up computing resources.

Defense: The system has a default interval threshold for SSL renegotiation requests. It will check the interval between two SSL renegotiation requests based on the default threshold (10,000 ms). If the renegotiation interval is smaller than 10,000 ms, the system will reset the connection and file attack records. Administrators can also use the "**ddos ssl renegotiation interval**" command to define a threshold.

22.5.4Application Layer

The most frequently encountered DDoS attacks on the application layer include HTTP DDoS and DNS DDoS attacks.

22.5.4.1 HTTP DDoS

- **HTTP Anomalies**

Symptom: Attackers initiate massive HTTP packets that are malformed or irresolvable, resulting in system process crashes and service unavailability.

Defense: The system will filter out and drop invalid packets using the HTTP state machine.

- **HTTP Slow Attack**

Symptom:

a. Slowloris: Attackers keep connections unclosed by sending an HTTP header at wide intervals, which uses up HTTP connection resources on the server, causing failures to establish connections for normal requests.

b. Slowpost: Similar to the Slowloris attack, attackers keep connections unclosed by sending a POST request at wide intervals.

Defense: The system has a default interval threshold for HTTP datagrams. In initial startup, the system will check the interval between two HTTP datagrams based on the default threshold (2000 ms). If the HTTP datagram interval is larger than 2000 ms, the system will record a timeout event. When three timeout events are accumulated, the system will reset the connection and file attack records.

As the actual traffic fluctuates, the system will generate a dynamic threshold, and check the HTTP datagram interval based on the dynamically generated threshold. Administrators can also use the "**ddos http slow interval**" command to define a threshold. When a self-defined threshold exists, the dynamically generated threshold does not take effect.

- **Long Form Submission and Challenge Collapsar (CC)**

Symptom: Processing long forms consumes massive system resources. Attackers implement such attacks by submitting long forms using HTTP methods like GET and POST. Similar to long form submissions, attackers initiating CC attackers send a great number of HTTP query requests to the target server, which will excessively occupy backend database resources. Finally, the server will be unable to serve normal requests.

Defense: The system has a default threshold for the number of cookies carried in an HTTP request, as well as a default threshold for the number of queries contained in a URL. In initial startup, the system will check the number of cookies carried in an HTTP request and the number of queries contained in a URL based on the default thresholds (both are 10). If the number of cookies carried in an HTTP request or the number of queries contained in a URL exceeds 10, the system will verify suspicious clients ("**ddos http verify on**" should be configured). If a client is confirmed as an attacker, the system will reset the connection and file attack records.

As the actual traffic fluctuates, the system will generate dynamic thresholds, and check the cookie and query count based on the dynamically generated thresholds. Administrators can also use the "**ddos http kvincookie**" and "**ddos http querycnt**" commands to define thresholds. When a self-defined threshold exists, the dynamically generated threshold does not take effect.

- HTTP Flood and Hashdos

Symptom: HTTP Flood aims to exhaust CPU resources, including GET Flood and POST Flood. Attackers request enormous resources at high rate to overload the server and cause the server unable to server normal requests.

Hashdos is similar to HTTP Flood. It exhausts the computing resources. The implementation of Hash tables is basically the same. If a Hash value is mapping to multiple keys, all entries will be indexed to the same position. This will leverage the performance of the Hash table.

Defense: The system has default thresholds for the response time of the HTTP servers in responding to GET, POST, HEAD, PUT and DELETE requests. In initial startup, the system will check the servers' response time based on the default thresholds (100 ms). If the average response time of all HTTP servers exceeds 100 ms and this fact persists for more than 30s, the system will follow these principles to take actions:

The system will collect and verify the top 10 traffic users ("**ddos http verify on**"), and add illegitimate clients to the ACL blacklist ("**ddos http acl blacklist on**"). For introductions about the ACL blacklist, please refer to section "18.4.6 ACL Blacklist". If "**ddos http verify on**" is not configured, the system will control the access of the top 10 traffic users according to HTTP ACL rules. For introductions about the HTTP ACL rule, please refer to section "18.3.3 HTTP ACL Rule".

As the actual traffic fluctuates, the system will generate dynamic thresholds, and check the server response time based on the dynamically generated thresholds. Administrators can also use the "**ddos http resptime**" command to define the thresholds for server response time. When a self-defined threshold exists, the dynamically generated threshold does not take effect.

22.5.4.2 DNS DDoS

The system provides the DNS DDoS attack defense mechanism that is can be independently controlled. Administrators can decide whether to enable DNS DDoS attack defense based on actual network needs. DNS DDoS attack defense protect against DNS Query Flood and DNS NXDomain Flood. In initial setup, DNS DDoS attack defense takes effect after the system DDoS attack defense function is enabled. Administrators can use the "**ddos dns {on|off}**" command to enable or disable DNS DDoS attack defense.

- **DNS Query Flood**

Symptom: DNS Query Flood is essentially a type of UDP flood. As DNS servers play a critical role in network running, DNS Query Flood always has a more serious impact. It is characteristics of sending massive DNS queries to the server, and the server will finally deny services because it is bound up in processing too many DNS queries.

Defense: The system has a default mechanism for detecting the response time of DNS servers. In addition, it supports dynamical DNS ACL rules for defending against the attack. During runtime, the system will calculate the average response time of DNS virtual services in processing DNS queries. If the current response time is found to be abnormally higher than the average value, and the current RPS is much larger than the RPS threshold, the system will decrease the RPS threshold

until the current response time returns to the average value and the RPS returns to a number below the threshold.

In SDNS application scenarios, the system has adequate resource capacities to defend against DNS Query Flood based on A, AAAA or CNAME resource record types. For DNS Query Flood based on other types of resource record types, the system provides protection for BIND9 DNS servers via DNS ACL rules.

For more introductions about the DNS ACL rule, please refer to section “18.3.4 DNS ACL Rule”.

- **DNS NXDomain Flood**

Symptom: DNS NXDomain Flood employs the same attack method as the DNS Query Flood, except that the domain name queried by DNS NXDomain Flood attackers are randomly generated or do not exist. When the server finds the domain name cannot be resolved, it will perform recursive query. The entire resolution process will overload and crash the DNS server.

Defense: The system will detect the suspicious source and file attack records.

22.5.5DDoS Blacklist

The DDoS blacklist keeps the latest 2000 attack records that the system detects. It can be displayed by the “**show ddos blacklist**” command. If the system is rebooted, these attack records will be cleared.

For example:

```
AN(config)#show ddos blacklist
1: ssl handshake attack. 2: ssl renegotiation attack.
1001: http slowloris attack.
1002: http slowpost attack.
1003: http query count abnormal.
1004: http cookie count abnormal.
1005: http resptime abnormal.
2001: rps exceeding limit.
3001: tcp attack.
3002: udp attack.
3003: slb dns attack.
3004: gslb dns attack.
start of blacklist
  2015 Jan 16 17:15:44 172.16.77.190,44647,v155,1
  2015 Jan 16 17:16:04 172.16.77.190,44648,v155,4097
2015 Jan 16 17:17:44 172.16.77.191,44649,v157,4098
  2015 Jan 16 17:18:04 172.16.77.192,44645,v156,2
```

Meaning of each item in the command output (“2015 Jan 16 17:15:44 172.16.77.190,44647,v155,1” is used as an example):

- 2015 Jan 16 17:15:44: indicates the time when an attack occurs.

- 172.16.77.190: indicates the IP address of the client launching an attack.
- 44647: indicates the port number of the client launching an attack.
- v155: indicates the virtual service name.
- 1: indicates the attack type.

The DDoS blacklist can be exported to a local file (in CSV format) for data viewing and analysis

22.5.6DDoS Attack Records

All DDoS attacks will be added to DDoS attack records. Administrators can use the "**show ddos record**" command to view DDoS attack records. They can also be backed up to a local host via the "**ddos record export**" command.

22.5.7ACL Blacklist

ACL blacklist records all IP addresses that are denied of access. If an IP address is added to the ACL blacklist, all traffic coming from it will be dropped.

The system supports dynamical blacklisting of an IP address. Administrators can also manually configure ACL blacklists.

Administrators can use the "**ddos http acl blacklist {on|off}**" command to enable or disable the function of dynamical blacklisting. After this function is enabled, the system will automatically add clients' IP addresses to the ACL blacklist when it detects an HTTP DDoS attack and confirms that the clients are illegitimate via the HTTP verification mechanism. Within 60 minutes, all traffic coming from a blacklisted IP address will be dropped. By default, this function is disabled. When it is disabled, the detected attackers' IP addresses will not be automatically added to the ACL blacklist.

In addition, administrators can manually add an IP address or subnet to the ACL blacklist, or import an IP list from an external URL address and then apply it to the ACL blacklist.

Example:

1. Add subnet "192.168.0.0/24" to the ACL blacklist and set the timeout value to 10.

```
AN(config)#acl blacklist rule 192.168.0.0 24 10
```

Import a self-defined IP list from the URL "ftp://10.8.3.28", apply it to the ACL blacklist and set the timeout value to 10.

```
AN(config)#acl blacklist import "ftp://10.8.3.28/iplist"
AN(config)#acl blacklist ipfile apply iplist 10
```

23 Advanced IPv6 Configuration

23.2 Overview

As the IPv4 addresses exhaust, how to transit from the IPv4 network to the IPv6 network becomes a challenge for many enterprises and organizations.

The APV appliance provides comprehensive support for IPv6 to help enterprises and organizations with the IPv4-to-IPv6 transition without any business interruption. With the IPv4/IPv6 dual stack support on APV, the IPv4 resources can be delivered to the IPv6 users, and vice versa. As a result, the IPv4-based and IPv6-based networks can be easily interconnected and intercommunicated. What's more, the APV appliance in the IPv6 network can achieve the same level of secure and efficient application delivery as it does in the IPv4 network.

This chapter will introduce functions and configurations about IPv6 SLB, DNS64/NAT64, DNS46/NAT46, IPv6 NAT and NDP.

23.3 IPv6 SLB

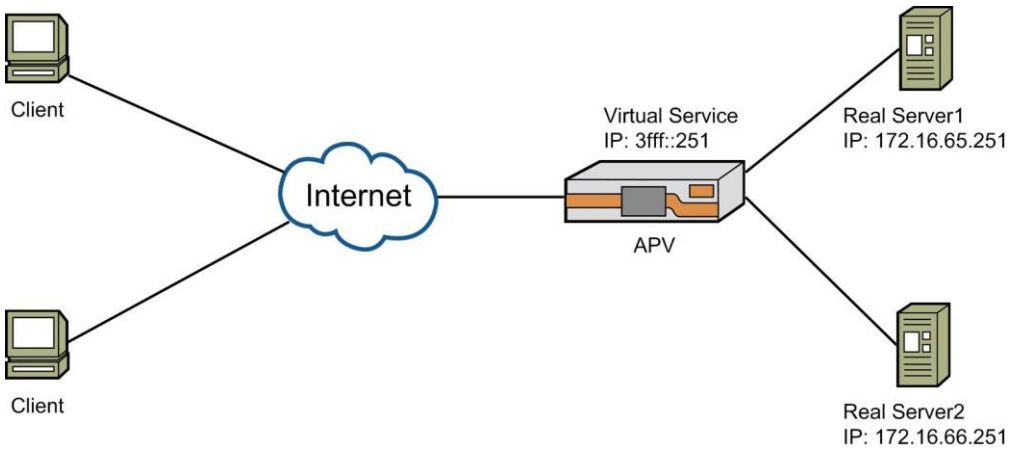
23.3.2Overview

APV provides comprehensive IPv6 support for the SLB module. For the service layer, SLB services from Layer 2 to 7 all support IPv6. For the service type, all of the service types except RDP, SIPUDP and SIPTCP support IPv6. For the SLB method and policy, all of the SLB methods except SNMP and all of the SLB policies support IPv6. Additionally, for the three SLB deployment modes, IPv6 is supported as follows:

- IPv6 to IPv4 (supported by the reverse proxy mode)
- IPv4 to IPv6 (supported by the reverse proxy mode)
- IPv6 to IPv6 (supported by the reverse proxy mode, transparent mode and triangle mode)

Among which, the reverse proxy mode can work in the IPv4/IPv6 co-existing network environment, as well as in the IPv4 only or IPv6 only network environment. The transparent mode and triangle mode can only work in the IPv4 only or IPv6 only network environment.

The following figure shows the topology of the "IPv6 to IPv4" deployment method.



23.3.2.1.1.1.1 "IPv6 to IPv4" of the SLB topology

Likewise, the "IPv4 to IPv6" deployment method can be adopted by configuring an IPv4 address for the virtual service and an IPv6 address for the real server.

For the "IPv6 to IPv6" deployment method, both the virtual service and real server should be configured with IPv6 addresses.

23.3.3Configuration Example

1. Execute the following command to configure the IPv6 virtual service:

slb virtual http *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]*

For example:

AN(config)#**slb virtual http virtualserv 3fff::251 8080 1000 tcp 3 3**

Execute the following command to configure the SLB policy:

slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*

For example:

AN(config)#**slb default virtualserv g1**

Execute the following command to configure the real server and SLB group:

slb real http *<real_name> <ip> [port] [max_conn]*
[http|tcp|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|none] [hc_up] [hc_down]
slb group method *<group_name> [algorithm]*

For example:

AN(config)#**slb real http Serv1 172.16.65.251**
AN(config)#**slb real http Serv2 172.16.66.251**
AN(config)#**slb group method g1 rr**

23.4 DNS64 and NAT64

23.4.2Overview

The DNS64 function supports conversion of DNS A, AAAA, MX and PTR queries and responses to help IPv6 clients to access IPv4 web servers or mail servers. For example, when receiving DNS AAAA queries sent from IPv6 clients, the DNS64 function converts them to DNS A queries and then converts the DNS A responses to DNS AAAA responses. The APV appliance returns the translated IPv6 addresses to IPv6 clients. When IPv6 clients use these IP addresses to access IPv6 servers, the NAT64 (Network Address Translation IPv6 to IPv4) function converts the IPv6 packets sent from these clients to IPv4 packets. When the APV appliance receives IPv4 packets from IPv4 servers, the NAT64 function converts IPv4 packets to IPv6 packets. This ensures that IPv6 clients can communicate with IPv4 servers normally.

The DNS64 function supports the application scenario where a client sends multiple DNS queries of the same type and different types over the same connection. In addition, it can defend against DNS query replay and response spoofing attacks.

The DNS64 and NAT64 functions can be deployed on two APV appliances separately, or deployed on one APV appliance.

23.4.3Priority Mode

For a DNS query owning both A and AAAA resources, the system supports configuration of the priority mode for determining which will be preferentially used. By default, DNS64 will adopt the AAAA priority mode.

- A priority mode: preferentially sends the DNS A query to the DNS A server and selects the DNS A response.
- AAAA priority mode: preferentially sends the DNS AAAA query to the DNS AAAA server and selects the DNS AAAA response.
- Response priority mode: sends the DNS A query to the DNS A server and DNS AAAA query to DNS AAAA server and preferentially selects the resource record in the response that is received first.

No matter which priority mode is used, DNS64 will ensure that the resource type of the returned response is the same with that of the query.

23.4.4Timer

The DNS64 function support priority timer and response timer.

23.4.4.1 Priority Timer

The priority timer works on only A and AAAA priority modes, in which DNS64 starts the priority timer after the preferred query is successfully sent. If no response is returned to this preferentially sent query, DNS64 will send the cached query. The priority timer is defined by the “ipv6 tune dnsnat priority time” command and defaults to 100 ms. The priority timer should neither be too short nor too long. If it is very short, the sending interval between the preferred query and the cached query will be extremely close. If it is very long, the client might resend a query after a long wait. It is recommended to define it in the range 100 ms to 500 ms.

23.4.4.2 Response Timer

In the A and AAAA priority modes, the response timer is started after the cached query is successfully sent. In the response priority mode and other circumstances where the priority mode is not used, such as the PTR query, the response timer is started whenever a query is successfully sent. If no response is received before the response timer ends, the corresponding transaction ID record will be deleted, rather than being deleted after the connection times out. The response timer is defined by the “ipv6 tune dnsnat priority time” command and defaults to 5000 ms.

23.5 Working Mechanism

DNS64 supports conversion of A, AAAA, MX and PTR queries and response.

- **DNS AAAA query and response**

When receiving a DNS AAAA query, DNS64 will generate a DNS A query based on it and then process it based on the priority mode configuration.

- A priority mode: DNS64 will cache the DNS AAAA query and send the DNS A query to the DNS A authoritative server.
  - If the DNS A query is sent successfully, DNS64 will start the priority timer (100 ms by default). If a DNS A response is received before the timer ends, it will convert it to a DNS AAAA response and then send it to the client.
  - If DNS64 fails to send the DNS A query or fails to receive a response before the timer ends, it will send the cached DNS AAAA query to the DNS AAAA authoratative server and start the response timer. If a DNS AAAA response is received before the timer ends, it will be directly sent to the client. If no response is received before the timer ends, DNS64 will delete the corresponding transaction ID record.
- AAAA priority mode: DNS64 will cache the DNS A query and send the DNS AAAA query to the DNS AAAA authoritative server.
  - If the DNS AAAA query is sent successfully, DNS64 will start the priority timer (100 ms by default). If a DNS AAAA response is received before the timer ends, it will be directly sent to the client.

- If DNS64 fails to send the DNS AAAA query or fails to receive a response before the timer ends, it will send the cached DNS A query to the DNS A authoritative server and start the response timer. If a DNS A response is received before the timer ends, it will be converted to a DNS AAAA response and then sent to the client. If no response is received before the timer ends, DNS64 will delete the corresponding transaction ID record.

- Response priority mode: DNS64 will simultaneously send the DNS A and DNS AAAA queries to the DNS A and DNS AAAA authoratative servers respectively and start the response timer. In normal circumstances, a DNS AAAA response will be received prior to a DNS A response. Therefore, it will be directly sent to the client without being converted. If a DNS A response is received first, it will be converted to a DNS AAAA response before being sent to the client.

➢ **DNS A query and response**

In the DNS64 application scenario, DNS A queries will be directly sent to the DNS A authoratative server and at the same time the response timer is started. If a DNS A response is received before the timer ends, it will be directly sent to the client. If DNS64 fails to send the DNS A query or fails to receive a response before the timer ends, it will delete the corresponding transaction ID record.

➢ **MX query and response**

DNS64 will create a duplicate of the MX query, for example, MX_DUP, and then send the query and convert the response based on the priority mode. The processing principles are almost the same as for DNS AAAA queries. The difference lies in the MX response, which might contain both DNS A and DNS AAAA resource records. In such conditions, DNS64 will preferentially select the DNS A resource record in the A priority mode, DNS AAAA resource record in the AAAA priority mode, and the resource record that is first parsed in the response priority mode. A conversion will be conducted as required.

➢ **PTR query and response**

For a PTR query, DNS64 will first parse it to obtain the type of the IP address that it queries and then process the PTR query based on the IP type, disregarding the priority mode configuration.

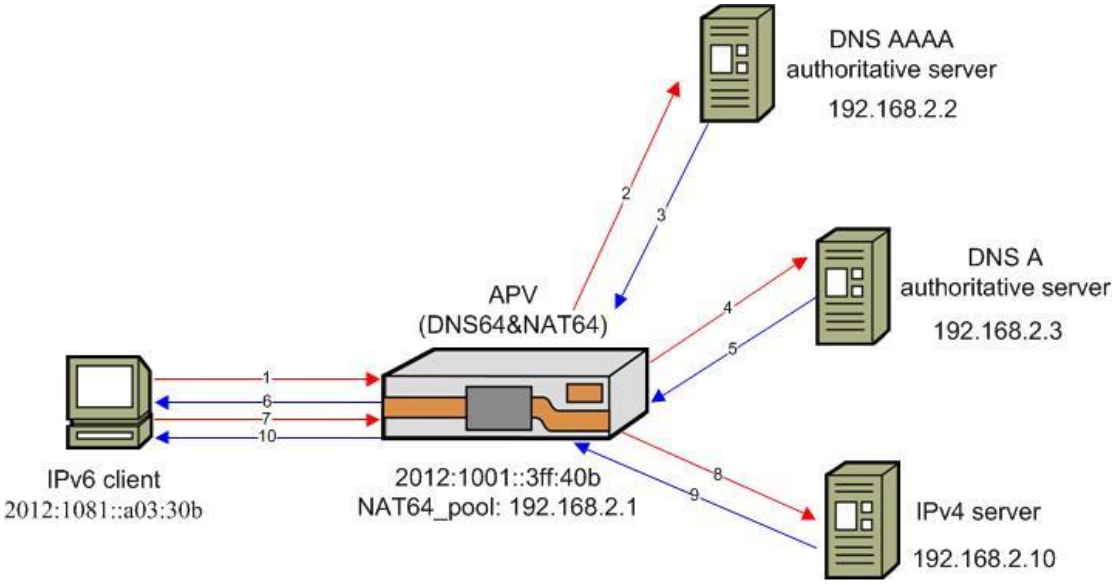
- If it queries an IPv4 address, the PTR query will be directly sent to the DNS A authoritative server and at the same time the response timer is started. If a response is received before the timer ends, it will be directly sent to the client.

- If it queries an IPv6 address, but it is one converted from an IPv4 address, DNS64 will first conduct a reverse conversion and then send the IPv4 PTR query to the DNS A authoritative server. At the same time, the response timer is started. If a response is received before the timer ends, it will be converted to an IPv6 PTR response and then sent to the client.

- If it queries an IPv6 address that is not a converted one, the PTR query will be directly sent to the DNS AAAA authoritative server and at the same time the response timer is started. If a response is received before the timer ends, it will be directly sent to the client.

If DNS fails to send any of the PTR query or fails to receive a response before the response timer ends, it will delete the corresponding transaction ID record.

The DNS64 and NAT64 functions are applicable to the “IPv6 to IPv4” scenario, as shown in the following figure.



23.5.2.1.1.1.1 “IPv6 to IPv4” Application Scenario

Following is an example in which a DNS AAAA query is processed in the AAAA priority mode. The working process of the DNS64 and NAT64 functions is as follows:

1. An IPv6 client (2012:1081::a03:30b) sends a DNS AAAA query to the APV appliance (2012:1001::3ff:40b) to resolve the domain name “www.example.com”.

The APV appliance generates a DNS A query based on the received DNS AAAA query, places it in the cache, and then sends the DNS AAAA query to the DNS AAAA authoritative server for the domain name.

If the DNS AAAA authoritative server has no AAAA record for the domain name, it will return an empty DNS AAAA response to the APV appliance. The APV appliance will ignore this response.

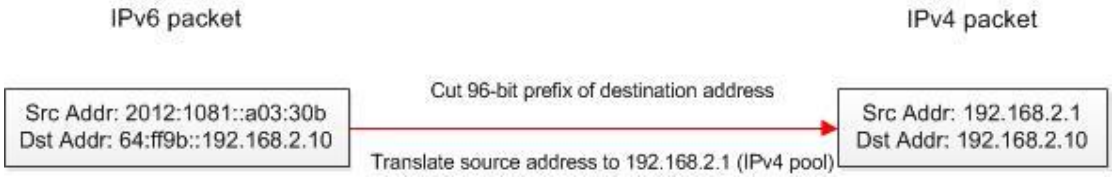
The APV appliance starts the priority timer (100 ms by default) after sending the DNS AAAA query. If the APV appliance does not receive any valid DNS AAAA response, it will send the cached DNS A query to the DNS A authoritative server for the domain name.

The APV appliance receives the DNS A response (for example, A: www.example.com - 192.168.2.10) from the DNS A authoritative server.

The APV appliance converts the DNS A response to a DNS AAAA response (for example, AAAA: www.example.com - 64:ff9b::192.168.2.10) by adding the configured IPv6 prefix. Then, the APV appliance returns the converted DNS AAAA response to the IPv6 client.

The IPv6 client uses the converted IPv6 address to access “www.example.com”.

The APV appliance converts the IPv6 packet (src: 2012:1081::a03:30b; dst: 64:ff9b::192.168.2.10) sent from the client to an IPv4 packet (src: 192.168.2.1; dst: 192.168.2.10), and sends the IPv4 packet to the target IPv4 server.



23.5.2.1.1.1.2 NAT64 Address Translation

The IPv4 sever returns an IPv4 packet (src: 192.168.2.10; dst: 192.168.2.1) to the APV appliance.

The APV appliance converts the IPv4 packet to an IPv6 packet (src: 64:ff9b::192.168.2.10; dst: 2012:1081::a03:30b) and returns the IPv6 packet to the IPv6 client.

23.5.3Application Notes

The DNS64 function can be enabled on only one DNS virtual service. This virtual service, acting as the DNS proxy, converts DNS AAAA queries to DNS A queries and then converts DNS A responses to DNS AAAA responses.

To make the DNS64 function work properly, you need to configure the “default” and “backup” policies for this virtual service. The APV appliance forwards DNS AAAA queries based on the “default” policy and forwards DNS A queries based on the “backup” policy. Therefore, the real servers associated with the “default” policy should be DNS servers that can answer AAAA records, and those associated with the “backup” policy should be DNS servers that can answer A records.

Note:

1. The DNS64 function does not support jumbo frames.
2. The DNS64 function only supports conversion of UDP based DNS queries and responses. Conversion of TCP based DNS queries and responses is not supported.

23.5.4Configuring DNS64 and NAT64

1. Execute the following command to add an IPv4 address pool:

ip pool <*pool_name*> <*start_ip*> *[end_ip]*

For example:

AN(config)#**ip pool NAT64_pool 192.168.2.1**

Execute the following commands to complete SLB configurations:

slb real dns *<real_name> <ip> <port> [max_conn]*
[dns|icmp|script-tcp|script-udp|sip-tcp|sip-udp|dns|none] [hc_up] [hc_down] [timeout]
slb real enable *<real_name>*
slb group method *<group_name> [algorithm]*
slb group member *<group_name> <real_name> [weight|cookie|url] [priority]*
slb virtual dns *<virtual_name> <vip> [vport] [arp|noarp] [max_conn]*
slb policy default *{virtual_name|vlink_name} {group_name|vlink_name}*
slb policy backup *{virtual_name|vlink_name} {group_name|vlink_name}*

For example:

AN(config)#**slb real dns dns_rs1 192.168.2.2**
AN(config)#**slb real dns dns_rs2 192.168.2.3**
AN(config)#**slb group method g1 rr**
AN(config)#**slb group member g1 dns_rs1**
AN(config)#**slb group method g2 rr**
AN(config)#**slb group member g2 dns_rs2**
AN(config)#**slb virtual dns dns_vs1 2012:1001::3ff:40b**
AN(config)#**slb policy default dns_vs1 g1**
AN(config)#**slb policy backup dns_vs1 g2**
AN(config)#**slb real enable dns_rs1**
AN(config)#**slb real enable dns_rs2**

Note: The DNS virtual service on which the DNS64 function is enabled can be associated with groups by using the “default” or “backup” policy only.

Execute the following commands to configure the DNS64 function:

ipv6 dns64 on *<dns_vs_name> [priority_mode]*
ipv6 dns64 prefix *<dns64_prefix>*

For example:

AN(config)#**ipv6 dns64 on dns_vs1 AAAA**
AN(config)#**ipv6 dns64 prefix 64:ff9b::**

Execute the following commands to configure the NAT64 function:

ipv6 nat64 ippool *<ipv4_pool_name>*
ipv6 nat64 prefix *<nat64_prefix>*
ipv6 nat64 timeout *<idle_timeout>*
ipv6 nat64 on

For example:

AN(config)#**ipv6 nat64 ippool NAT64_pool**
AN(config)#**ipv6 nat64 prefix 64:ff9b::**
AN(config)#**ipv6 nat64 timeout 300**
AN(config)#**ipv6 nat64 on**

Note:

- If the DNS64 and NAT64 functions need to work together on one APV appliance, make sure that the values of the parameters “dns64_prefix” and “nat64_prefix” are the same.
- After an IPv4 address pool is configured for the NAT64 function and the function is enabled, please do not delete the IPv4 address pool. Otherwise, the NAT64 function will be disabled and its related configuration will be deleted.

23.6 DNS46 and NAT46

23.6.2Overview

The DNS46 function supports conversion of DNS A, AAAA, MX and PTR queries and responses to help IPv4 clients to access IPv6 web servers or mail servers. For example, when receiving DNS A queries sent from IPv4 clients, the DNS46 function converts them to DNS AAAA queries and then converts the DNS AAAA responses to DNS A responses. It also creates mapping records between the IPv6 addresses and IPv4 addresses. The APV appliance returns the translated IPv4 addresses to IPv4 clients. When IPv4 clients use these IPv4 addresses to access IPv6 servers, the NAT46 function converts IPv4 packets sent from clients to IPv6 packets based the created mapping records. When the APV appliance receives IPv6 packets from IPv6 servers, the NAT46 function converts IPv6 packets to IPv4 packets. This ensures that IPv4 clients can communicate with IPv6 servers normally.

The DNS46 function supports the application scenario where a client sends multiple DNS queries of the same type and different types over the same connection. In addition, it can defend against DNS query replay and response spoofing attacks.

Because the DNS46 and NAT46 functions both use the mapping records, they must be deployed on one APV appliance.

23.6.3Priority Mode

The DNS46 function also supports the same priority modes as the DNS64 function (see section “23.4.3 Priority Mode”).

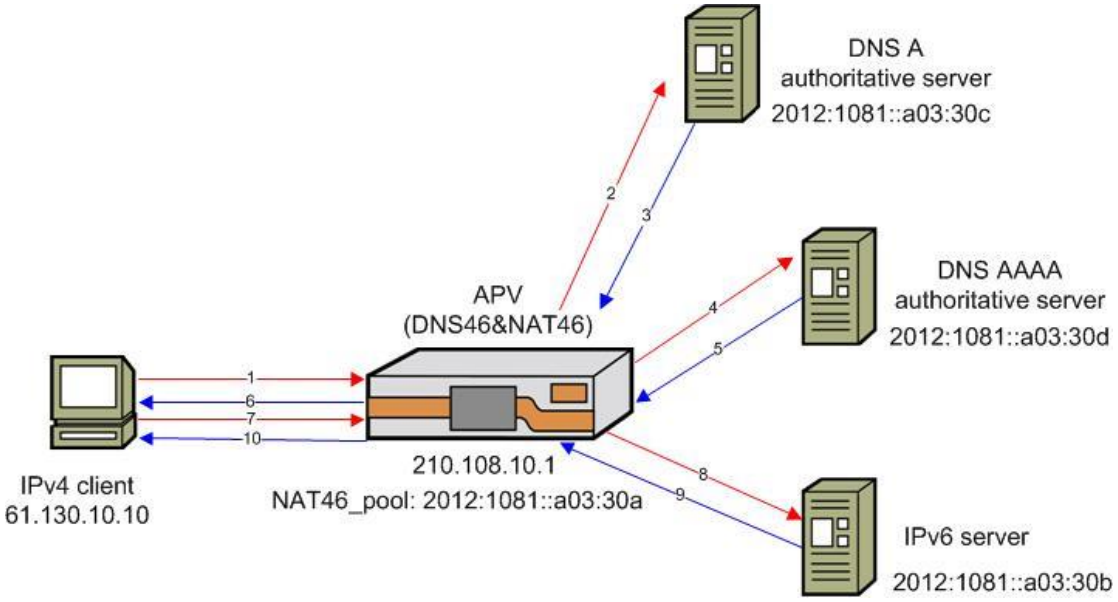
23.6.4Timer

The DNS46 function supports the same priority timer and response timer as the DNS64 function (see section “23.4.4 Timer”).

23.6.5Working Mechanism

Similar to the DNS64 function, the DNS46 function also supports conversion of A, AAAA, MX and PTR queries and responses. The working mechanism is the same with the DNS64 function (see section “23.5 Working Mechanism”).

The DNS46 and NAT46 functions are applicable to the “IPv4 to IPv6” scenario, as shown in the following figure.



23.6.5.1.1.1.1 “IPv4 to IPv6” Application Scenario

Following is an example in which a DNS A query is processed in the A priority mode. The working process of the DNS46 and NAT46 functions is as follows:

1. An IPv4 client (61.130.10.10) sends a DNS A query to the APV appliance (210.108.10.1) to resolve the domain name “www.example.com”.
2. The APV appliance generates a DNS AAAA query based on the received DNS A query, places it in the cache, and then sends the DNS A query to the DNS A authoritative server for the domain name.
3. If the DNS A authoritative server has no A record for the domain name, it will return an empty DNS A response to the APV appliance. The APV appliance will ignore this response.
4. The APV appliance starts the priority timer (100 ms by default) after sending the DNS A query. If the APV appliance does not receive any valid DNS A response, it will send the cached DNS AAAA query to the DNS AAAA authoritative server for the domain name.
5. The APV appliance receives the DNS AAAA response (for example, AAAA: www.example.com - 2012:1081::a03:30b) from the DNS AAAA authoritative server.
6. The APV appliance converts the DNS AAAA response to a DNS A response (for example, A: www.example.com - 210.108.10.10) based on the configured address pool (that is, the IPv4

mapping subnet configured by using the command "**ipv6 dnsnat46 ipmap**"). Then, the APV appliance returns the converted DNS A response to the IPv4 client. Meanwhile, the system creates a mapping record between 2012:1081::a03:30b and 210.108.10.10.

7. The IPv4 client uses the converted IPv4 address to access "www.example.com".

8. The APV appliance converts the IPv4 packet (src: 61.130.10.10; dst: 210.108.10.10) sent from the client to an IPv6 packet (src: 2012:1081::a03:30a; dst：2012:1081::a03:30b) based on the created address mapping record, and sends the IPv6 packet to the target IPv6 server.

23.6.5.1.1.1.2 NAT46 Address Translation

The IPv6 server returns an IPv6 packet (src: 2012:1081::a03:30b; dst: 2012:1081::a03:30a) to the APV appliance.

The APV appliance converts the IPv6 packet to an IPv4 packet (src: 210.108.10.10; dst: 61.130.10.10) and returns the IPv6 packet to the IPv4 client.

23.6.6Application Notes

The DNS46 and NAT46 functions can be enabled on only one DNS virtual service. This virtual service, acting as the DNS proxy, converts DNS A queries to DNS AAAA queries and then converts DNS AAAA responses to DNS A responses.

To make the DNS46 and NAT46 functions work properly, you need to configure the "default" and "backup" policies for this virtual service. The APV appliance forwards DNS AAAA queries based on the "default" policy and forwards DNS A queries based on the "backup" policy. Therefore, the real servers associated with the "default" policy should be DNS servers that can answer AAAA records, and those associated with the "backup" policy should be DNS servers that can answer A records.

Note:

1. The DNS46 function does not support jumbo frames.
2. The DNS46 function only supports conversion of UDP based DNS queries and responses. Conversion of TCP based DNS queries and responses is not supported.

23.6.7Configuring DNS46 and NAT46

1. Please refer to section "**Configuring DNS64 and NAT64**" to complete the address pool and SLB configurations via CLI.

For example:

```
AN(config)#ip pool NAT46_pool 2012:1081::a03:30a

AN(config)#slb real dns dns_rs1 2012:1081::a03:30c
AN(config)#slb real dns dns_rs2 2012:1081::a03:30d
AN(config)#slb group method g1 rr
AN(config)#slb group member g1 dns_rs1
AN(config)#slb group method g2 rr
AN(config)#slb group member g2 dns_rs2
AN(config)#slb virtual dns dns_vs1 210.108.10.1
AN(config)#slb policy default dns_vs1 g1
AN(config)#slb policy backup dns_vs1 g2
AN(config)#slb real enable dns_rs1
AN(config)#slb real enable dns_rs2
```

Execute the following command to enable both the DNS46 and NAT46 functions for a specified DNS virtual service:

ipv6 dnsnat46 on <*dns_vs_name*> *[priority_mode]*

For example:

```
AN(config)#ipv6 dnsnat46 on dns_vs1 A
```

Execute the following command to configure an IPv4 subnet used to create the address mapping table:

ipv6 dnsnat46 ipmap <*ipv4_address*> <*netmask*> *[timeout]*

For example:

```
AN(config)#ipv6 dnsnat46 ipmap 192.168.2.0 255.255.255.0 600
```

Execute the following command to specify the IPv6 address pool used by the NAT46 function:

ipv6 dnsnat46 ippool <*ipv6_pool_name*>

For example:

```
AN(config)#ipv6 dnsnat46 ippool NAT46_pool
```

Note: After an IPv6 address pool is configured for the NAT46 function and the function is enabled, please do not delete the IPv6 address pool. Otherwise, the NAT46function will be disabled and its related configuration will be deleted.

Execute the following command to set the idle timeout period for NAT46 TCP connections:

ipv6 dnsnat46 timeout <*idle_timeout*>

For example:

AN(config)#**ipv6 dnsnat46 timeout 300**

23.7 IPv6 support for NAT

23.7.2Overview

The APV appliance not only supports NAT64 and NAT46, but also supports “IPv6 to IPv6” NAT, which is able to translate the IPv6 addresses on the internal network to the IPv6 addresses on the Internet. In addition, NAT pool configurations also support IPv6.

Note:

- “IPv6 to IPv6” NAT supports TCP and UDP packets, but does not support ICMP and FTP packets.
- When configuring the “IPv6 to IPv6” NAT, the parameter “gateway” cannot be configured in the “**nat port** *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*” command.

23.7.3Configuration Example

Execute the following command to configure the IPv6 NAT:

nat port *{pool_name|vip} <source_ip> {netmask|prefix} [timeout] [gateway] [description]*

For example:

AN(config)#**nat port ipv6_pool 3fff::12 64**

23.8 NDP

23.8.2Overview

NDP (Neighbor Discovery Protocol), a key protocol of the IPv6 stack, can be used for obtaining the link address information of other neighbor nodes connected with the local nodes.

Similar to the ARP (Address Resolution Protocol) of the IPv4 stack, NDP can perform address transformation between the network layer and the link layer. The difference is that NDP uses ICMPv6 (Internet Control Message Protocol version 6) and multicast to manage the information exchanged among the neighbored nodes (within the same link), and keeps the address mapping between the network layer and the link layer in the same subnet.

23.8.3Configuration Example

Execute the following command to configure the NDP entry:

ipv6 ndp <*ipv6_address*> <*mac_address*>

For example:

AN(config)#**ipv6 ndp bb::55 00:21:9C:45:80:9F**

24ePolicy

24.2 Overview

ePolicy is a script-based function for extending the capabilities of the APV appliance. Using the scripts written in Tools Command Language (TCL), you can customize new features in addition to the existing functions on the APV appliance. For example, the APV appliance can be customized to support more application protocols, precisely control IP application traffic in both incoming and outgoing directions, or control the access of the specified client to real services.

24.3 ePolicy Elements

The elements of ePolicy are as follows:

- Setting
- Event
- Command
- Command invocation rule

24.3.2Setting

Setting defines the traffic scenario where the ePolicy function takes effect. That is, it defines the message type.

24.3.3Event

ePolicy uses an event-driven and message-response mechanism. The APV appliance defines an event for every action occurring in each Client-APV-Server connection. When such an event occurs, the APV appliance will process traffic according to preconfigured ePolicy commands.

24.3.4Command

ePolicy uses commands to instruct the APV appliance to process traffic after an event occurs, such as rewriting packet contents, selecting real servers, selecting groups, or querying whether a group has valid real servers.

24.3.5Command Invocation Rule

Command invocation rules indicate the relationship between events and commands. Based on the command invocation rules, you can flexibly combine the events and commands to intercept, detect, convert, or redirect the IP application traffic in both incoming and outgoing directions.

For detailed information of events, commands, and command invocation rules, contact Array Customer Support for related documents.

24.4 ePolicy Scripts

ePolicy employs event handler scripts to control the APV's behaviors. A script consists of the following parts:

- **Script Description**

In the front of the script, you can add comments as the description of the script to introduce its function. You can use the keyword "description:" to add the description in English and the keyword "description_cn:" to add the description in Simplified Chinese.

Example:

```
#description: Redirect HTTP request
#description_cn:  重定向 HTTP 请求
```

- **Scenario Setting**

Scenario setting specifies the traffic type of a virtual service.

For UDP, HTTP or HTTPS virtual services, you do not need to add the scenario setting for them. ePolicy can automatically parse the type of traffic for these virtual services.

For TCP and TCPS virtual services, you need to add the scenario setting to the script and attach it to them. The scenario setting will tell ePolicy what type of traffic that the virtual service is handling. If the TCP and TCPS virtual services are handling different types of traffic, the contents of scenario setting should be different, as shown in the following table.

24.4.2.1.1.1 Scenaio Setting Examples

Traffic Type	Scenario Setting
HTTP	message::type http
Diameter	message::type binary binary_message::length_start_offset 1 binary_message::length_end_offset 3
Generic TCP	message::type binary

- **Command Invocation Rule**

You can write command invocation rules using events and commands according to the actual requirement.

For the examples of the event handler scripts, contact Customer Support for related documents.

24.5 ePolicy Applications

With ePolicy scripts, an APV appliance can be customized to:

- Balance loads. When being applied to Server Load Balancing (SLB), ePolicy can work as SLB policies and collaborate with SLB methods to realize load balancing among real services.
- Provide respestive load balance for different operations of MySQL servers.
- Analyze the packet contents of the HTTP, Simple Object Access Protocol (SOAP), eXtensible Markup Language (XML), and Diameter protocols.
- Receive, send, analyze, and discard Generic TCP and TCPS packets.
- Perform pattern matching for txt data
- Control TCP connections
- Monitor and take statistics of traffic
- Parse and modify HTTP headers and body
- Parse SSL/TLS fields
- Parse and modify TCP options

Note: ePolicy supports only MySQL 5.x servers.

24.5.2SLB Methods Collaborating with ePolicy

In SLB, ePolicy can collaborate with the following methods:

- Round Robin (rr)
- Least Connection (lc)
- Shortest Response (sr)
- Persistent IP (pi)
- Hash IP (hi)
- Hash IP and port (hip)
- Consistent hash IP (chi)
- SNMP (snmp)

24.5.3SLB Polices and ePolicy

In SLB, ePolicy has higher priority than the existing SLB policies. When a virtual service is associated with ePolicy scripts, the system will preferentially select available real services based on the ePolicy script. If failing to select an available one, it will then try to use the SLB policies associated with the virtual service.

24.6 ePolicy Configurations

To complete the ePolicy configuration, perform the following steps:

1. Prepare the event handler script according to events, commands, and command invocation rules.
2. Import the event handler script.
3. Associate the virtual service with the event handler script.

The following takes balancing loads among servers according to HTTP request packet method to describe how to configure ePolicy.

24.6.2Preparing the Event Handler Script

The following example shows the contents of the event handler script:

```
#description: Selecting real service based on HTTP method
#description_cn: 根据 HTTP 方法选择后台服务

message::type http

when HTTP_REQUEST_HEADER {
    if { [http::method] == "GET" } {
        slb::select_server realserver_1
    } else {
        slb::select_server realserver_2
    }
```

In which, “http_slb.tcl”, “realserver_1” and “realserver_2” are the names of SLB real servers.

For detailed information of events, commands, and command invocation rules, contact Customer Support for related documents.

24.6.3Importing the Event Handler Script

Execute the following command to import the event handler script:

epolicy import script <*url*> <*script_name*>

For example:

AN(config)#**epolicy import script http://192.168.10.10/http_slb.tcl http_slb.tcl**

24.6.4Associating the Virtual Service with the Event Handler Script

Execute the following command to associate the virtual service with the event handler script:

epolicy attach script <*vs_name*> <*script_name*>

For example:

AN(config)#**epolicy attach script vs_epolicy http_slb.tcl**

24.6.5Configuration Results

After the preceding configurations are completed, the APV appliance will:

- Direct HTTP request packets whose method is GET to real server realserver_1.
- Direct HTTP request packets with other methods to real server realserver_2.

25Logging

25.2 Overview

The Logging mechanism used by the APV appliance is Syslog compliant. System error and HTTP access information during proxy application are logged by using the logging subsystem. Syslog is a standard program for Unix and there are also Syslog implementations for Windows. On the Unix platform, syslog is started by the syslogd daemon. The syslogd daemon takes charge of receiving and storing log messages from local machine or remote machine, which listens at UDP 514 port. APV appliance supports three remote log servers.

25.3 Understanding Logging

25.3.2Syslog

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

Syslog logging has eight valid levels of log message severity: emerg, alert, crit, err, warning, notice, info and debug. And the supported facilities are LOCAL0 to LOCAL7. Users can view the internal log buffer, select the transport protocol, and configure syslog source and destination ports and the alerts on log message string match.

25.3.3RFC 5424 Syslog

RFC5424 defines the standard format of syslogs. The APV appliance supports the RFC 5424 syslog function. When the RFC 5424 syslog function is enabled, the system will generate system logs in the standard format defined by RFC 5424. The format is "<PRI>VER TIMESTAMP HOSTNAME APPNAME PROCID MSGID STRUCTURED-DATA MSG-CONTENT". (The PROCID and STRUCTURED-DATA fields are not supported temporarily and are displayed as "-".) By default, the RFC 5424 syslog function is disabled. The configuration of "**log rfc5424 on**" takes effect only when the system logging function has been enabled by using the "**log on**" command.

25.3.4HTTP Access Logging

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

HTTP Access Logging supports four standard formats: Combined, WELF (WebTrends Enhanced Log), Common and Squid. And users can define their own logging format by using the "**log http custom**" command.

Note: The APV appliance will record an HTTP access log only after the HTTP communication between the client and the Web server is completed successfully.

25.3.5Log 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 ArrayOS 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.

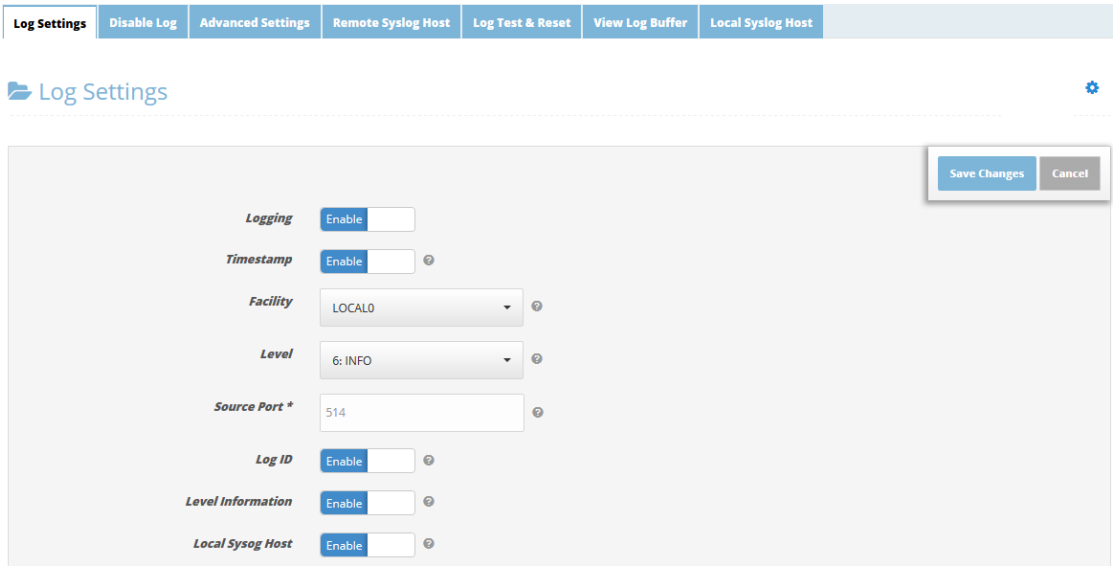
If multiple log filters are set on a syslog host, the logs matching one of the filter strings will go to the syslog host.

25.3.6Local Syslog Host

The system allows the administrator to enable one local syslog host on the appliance to receive and store the system logs. By default, the local syslog host is disabled.

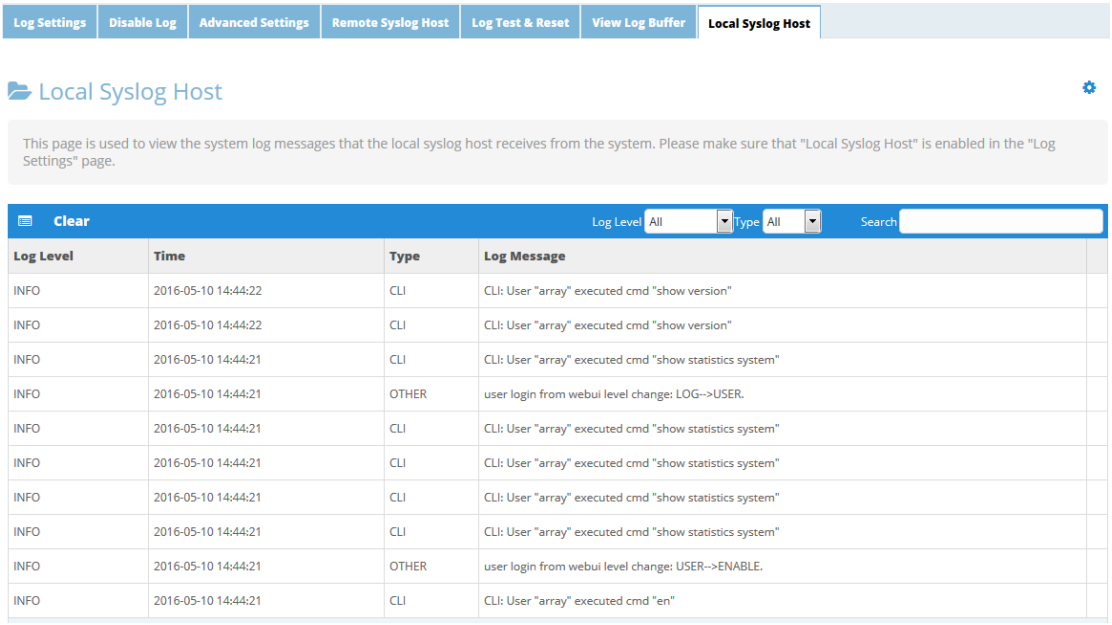
After being enabled, the local syslog host starts to receive the system logs sent by the system on the IP address 127.0.0.1 and the UDP port 515. The local syslog host can save a maximum of 500,000 system logs for every log level. These system logs stored on the local syslog host can be viewed and exported via the WebUI.

To enable the local syslog host, execute the "**log localhost on**" command in CLI. Alternatively, on the WebUI, select **Admin Tools > System Logs > Log Settings**, set the **Local Syslog Host** slider to **Enable** in the **Log Settings** area and click the **Save Changes** button, as shown in the following figure.



25.3.6.1.1.1.1 Enabling Local Syslog Host

To view the system logs stored on the local syslog host, click the **Local Syslog Host** tab, as shown in the following figure.



25.3.6.1.1.1.2 Viewing System Logs on the Local Syslog Host

25.4 Logging Configuration

25.4.2Configuration Guidelines

25.4.2.1.1.1 General Settings of Logging

Operation	Command
Enable the logging	**log {on|off}**

Enable RFC 5424 Syslog	**log rfc5424 {on|off}**
Configure the remote host	**log host** <*host_ip*> *[port] [udp|tcp] [host_id]*
Set log filters	**log filter** <*host_id*> <*filter_id*> <*filter_string*>
Set log level	**log level** <*level*>
Change log facility	**log facility** <*facility*>
Set HTTP access logging format	**log http** *{squid|common|combined|welf} [vip|novip] [host|nohost]* **log http custom** <*format*>

25.4.3Configuration Example via CLI

1. Enable Logging function.

The logging system is off by default.

```
AN(config)#log on
```

Enable the RFC 5424 Syslog function.

```
AN(config)#log rfc5424 on
```

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.

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

Set log filters for the configured host.

No more than 3 log filters can be set on one syslog host. Log filter canot be set on the syslog host whose ID is 0 (it is configured by the command "**log host**"). After this command is executed, only the logs matching this filter string go to the syslog host.

```
AN(config)#log filter 1 1 "index"
```

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.

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

Change the syslog facility.

The default facility is LOCAL0.

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

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.

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

Generate a test log.

You can run the command "**log test**" to generate an emerg-level log.

```
AN(config)#log test
```

View and clear logs.

You can run the following command "**show log buff** *{forward|backward} [match_str]*" to view logs in the log buffer. The parameters "backward" and "forward" are used to display the logs that are latest and first generated respectively.

```
AN(config)#show log buffer backward
start of buffer
<128>1 2012-07-17T06:35:26Z AN - - 100021002 - Array Networks test message
```

You can run the command "**clear log buff**" to clear logs from the log buffer.

```
AN(config)#clear log buffer
```

25.5 Application Visualization

During system runtime, learning its running state in real time is critical to prompt troubleshooting and assurance of uninterrupted services. However, to understand the system's running status requires the collection of a large amount of data. As IT deployment generally adopts distributed and fault tolerant architectures in consideration of scalability, security and other factors, it makes data collection even more difficult. What's more, with the large volume of data collected, it is hard to extract valuable information.

The APV appliance provides application visualization for SLB TCP, SLB HTTP, SSL, LLB and SDNS services based on the WebUI and ELK software-based platforms. The WebUI and ELK platforms will unify the recording, storage and mining of scattered system resources, configurations and application data and intuitively present the data after analysis in a visible way. Operation and maintenance personnel can visibly manage applications, understand application deliveries at any time, and acquire the system status and locate faults effectively.

25.5.2WebUI Platform

Application visualization capabilities provided by the WebUI focus on monitoring of system resources, configurations and performance.

25.5.2.1 System Dashboard

On the homepage (System Dashboard) of the WebUI, the system graphically presents data such as system uptime, system status, application status, network status, system load, network load and connection status. In addition, the homepage provides an individual monitoring tab for each of the SLB, LLB and GSLB modules.

- The SLB monitoring tab presents data including virtual service status, real service status and top 10 virtual services by total bytes input.
- The LLB monitoring tab presents data including link status and top 10 links by total bytes input.
- The GSLB monitoring tab presents data including service status and top 10 hosts by total requests.

25.5.2.2 SLB Monitoring Center

The administrator can choose **SLB** > **Monitoring Center** to navigate to the SLB monitoring center. In the SLB monitoring center, more information is displayed graphically.

- All virtual service status (Up or Down)
- All real service status (Up or Down)
- Virtual service total bytes input/output ranking, total connections ranking
- Real service total bytes input/output ranking, average response time ranking, total hits ranking
- All virtual services' total concurrent connections, average inbound/outbound bandwidth, total cache hits ratio, compression ratio for all data
- All SSL hosts' average inbound/outbound bandwidth
- All SSL virtual hosts' average inbound/outbound bandwidth
- All SSL real hosts' average inbound/outbound bandwidth
- Single virtual service's policy hits ranking, associated real service load ranking, number of concurrent connections, number of established connections per second, average inbound/outbound bandwidth, number of requests per second, client connection average RTT, etc. In the associated real service load ranking chart, if a real service is associated with multiple virtual services, the displayed value is a sum of loads on all virtual services.
- Single real service's number of concurrent connections, number of established connections per second, average inbound/outbound bandwidth, number of requests per second, average response time, number of returned HTTP response codes, health status, etc.
- Single SSL virtual host's number of concurrent connections, number of established connections per second, average inbound/outbound bandwidth

- Single SSL real host’s number of concurrent connections, number of established connections per second, average inbound/outbound bandwidth

25.5.3ELK Platform

ELK is an acronym for Elasticsearch, Logstash and Kibana. Elasticsearch is a full-text search engine based on Lucene. Logstash is mainly used to collect, analyze and filter logs. Kibana helps to aggregate, analyze and search important logs and provides Elasticsearch and Kibana with a Web friendly log analysis interface. Administrators can acquire more information about the features and supported data collection and management functions of the open-source ELK platform and other subscription plans at https://www.elastic.co/cn/subscriptions.

To implement application visualization, the logging module of the system is enhanced to support independent operation of SLB TCP logging, SLB HTTP logging, LLB logging, SDNS logging and SSL logging visualization mechanisms. Administrators can use the “log appvisual” command to enable visualization logging for the SLB TCP, SLB HTTP, LLB, SSL and SDNS modules, so that the system will collect more detailed logs needed by the ELK server. With the ELK server configured as the log server of the APV appliance, the APV appliance will send the collected logs to the ELK server.

After visualization logging is enabled for SLB TCP, SLB HTTP, LLB, SSL and SDNS, the APV appliance can implement visualized management and monitoring for the following applications on the ELK platform.

25.5.3.1 SLB HTTP

- HTTP request type distribution
- URL access ranking
- HTTP response latency across servers in the same group
- Single URL response latency
- Total number of access requests
- URL or server average latency trend
- URL or server maximum or minimum latency
- Heavy traffic URL distribution
- URL access failure rate analysis
- Response size distribution
- HTTP protocol version distribution
- Visitor location tracking
- HTTP response code distribution and trend

- HTTP response error rate statistics
- Request line distribution
- Browser type distribution

25.5.3.2 SLB TCP

- Virtual services' number of current connections
- Virtual services' number of established connections per second (CPS)
- Virtual services' average inbound/outbound bandwidth
- Virtual services' average inbound/outbound packets per second (PPS)
- Client-side handshake delay RTT
- Real services' number of current connections
- Real services' number of established connections per second (CPS)
- Real services' average inbound/outbound bandwidth
- Real services' average inbound/outbound packets per second (PPS)
- Server-side handshake delay (based on health check time)

25.5.3.3 SDNS

- Geographic distribution of requests
- Geographic distribution of responses
- Statistics of failed resolutions
- Intelligent analysis statistics
- Non-intelligent analysis statistics
- Total number requests, total number of failed resolutions, total number of responses
- Top 30 domain names of requests and responses
- Response status trend
- Query type histogram
- Total number of resolutions
- Total number of responses

25.5.3.4 SSL

- Top 20 SSL clients

- Top 20 SSL cipher suites
- SSL version information
- Total number of SSL handshakes
- SSL SNI information
- SSL handshake failure reason
- SSL handshake success/failure distribution

25.5.3.5 IPV6

- Total number of requests
- Total number of IPv6 clients
- Real-time number of requests
- IPV6/IPv4 request ratio

25.5.3.6 LLB

- Link health status (up or down)
- Total number of links
- Bandwidth of each link, including the total bandwidth, upstream bandwidth, downstream bandwidth
- Bandwidth usage of each link
- Application traffic ratio analysis (based on DPI)
- Link status monitoring (normal, busy, offline, no data)

ELK supports deployment physical machines and virtualization platforms such as VMware, KVM and Array AVX platforms. For more details about ELK deployment, please contact Customer Support to obtain the deployment guide.

26 System Management

26.2 Administrative Tools

26.2.2 Overview

This chapter will focus on various configuration maintenance elements, such as downloading new ArrayOS software, rebooting your APV appliance, reverting your configuration to a previously saved status or returning the APV appliance to its factory default settings among other closing strategies.

The final series of configuration options concern the running operation of your APV 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.

26.2.3 Administrative Tools Configuration

26.2.3.1 Configuration Guidelines

26.2.3.1.1.1 General Settings of Administrative Tools

Operation	Command
Configuring External Authentication	**admin aaa {on|off}** **admin aaa method** *[radius|tac_x]* **admin aaa server** *<server_id> <host_name|ip_address> <port> <secret>*
System shutdown and reboot	**system shutdown** *[halt|poweroff]* **system reboot** *[interactive|noninteractive]*
Configuration file maintenance	**clear config file** **clear config secondary** **clear config primary** **clear config all** **clear config factorydefault** **clear config timeout** **write memory** **write file** *<file_name>* **write net tftp** *<ip_tftp> <file_name>* **write net scp** *{remote_server_ip|name} <user_name> <config_file_name>* **config memory**

	config net tftp *<tftp_server_ip> <config_file_name>* **config file** *<file_name>*
Software upgrade	system update *<url> [immediate|deferred] [partition_number]*
Configuration Synchronization	**synconfig peer** *<peer_name> <peer_ip>* **synconfig challenge** *<code>* **synconfig to** *<name>* **synconfig from** *<name>*
SDNS Synchronization	**synconfig peer** *<peer_name> <peer_ip>* **synconfig challenge** *<code>* **synconfig sdns to** *<peer_name>*
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**
Remote access	**ssh remote** *"user@hostname"* **telnet** *"host port"*

26.2.3.2 Configuration Example via CLI

26.2.3.2.1 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 ArrayOS system.

```
AN(config)#admin aaa on
AN(config)#admin aaa method RADIUS
AN(config)#admin aaa server es01 "10.1.1.1" 1812 radiussecret
AN(config)#admin aaa server es02 radius_host 1812 radiussecret
```

26.2.3.2.2 System Maintenance

Simply enough, employing the "**quit**" command will allow you to exit the CLI. In the event you want to terminate all APV appliance interactions with your network, you will need to use the "**system shutdown**" command.

```
AN(config)#system shutdown
```

The APV appliance will prompt you with an alert to verify the shutting down process. By entering "**YES**", case sensitive, the APV appliance will commence the shutting down operation. After a brief, 60-second period, users may turn off the appliance.

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

```
AN(config)#system reboot
```

26.2.3.2.3 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 ArrayOS 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 APV appliance. Finally, you may export and import the configuration by using TFTP.

To clear the running configuration on the APV appliance:

```
AN(config)#clear config all
```

Now the APV 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 APV 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 APV appliance affords you the opportunity to save configurations to three separate places; the "memory" file which is where the APV appliance calls up configuration settings upon reboot, the "file" where the APV 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:

```
AN(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:

```
AN(config)#config memory
AN(config)#config file new_lb
AN(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 APV 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.

26.2.3.2.4 Software Upgrade Procedure

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

To upgrade to a newer release there are several steps to take.

First, contact Customer Support to gain access to the software and documentation repository. Contact your customer support representative or send email to customer support.

Once you have received a password and verified with a customer support engineer that ArrayOS needs upgrade, you can download the software image using the customer support 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 ArrayOS. 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:

```
AN(config)#system update http://192.168.10.10/build/ArrayOS-Beta_APV_10_4_0_20.array

This will upgrade your system from
http://192.168.10.10/build/ArrayOS-Beta_APV_10_4_0_20.array
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.

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 APV 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.
- Do not install a two-partition software version on a four-partition system disk.

The ArrayOS will then shut down all load balancing features and download the software image, verify that the software is produced at Array Networks 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 APV appliance will reboot. Upon reboot, you should use the "**show version**" command to verify that the upgrade is successful.

Caution:

- If executing this command via an SSH connection and if the connection is lost during update procedure, the APV appliance will not be able to complete the update process.
- Do not disconnect the connections to the APV appliance during the system updating process.

Software Licenses

Some software features of the APV appliance may be under software license key control. If you need these software features, please contact customer support to obtain a new license key.

26.2.3.2.5 Configuration Synchronization

The Configuration Synchronization feature of the APV appliance allows administrators to transfer configuration information among APV appliances. Configuration Synchronization is a set of commands that allow you to manage and configure boxes on the same network or on different networks but are interconnected via NAT. You may transfer configuration information from one APV appliance to other APV appliances. By using configuration synchronization, you can quickly setup an Active-Standby configuration. The rest of the section will cover how to use this feature.

Assume that IP addresses of APV1 and APV2 are 192.168.1.1 and 192.168.1.2. To synchronize the configuration from APV1 to APV2, the following configurations need to be made:

1. Configure configuration synchronization on APV1.

```
AN1(config)#synconfig challenge synpassword
AN1(config)#synconfig peer APV1 192.168.1.1
AN1(config)#synconfig peer APV2 192.168.1.2
```

Configure configuration synchronization on APV2.

```
AN2(config)#synconfig challenge synpassword
AN2(config)#synconfig peer APV1 192.168.1.1
AN2(config)#synconfig peer APV2 192.168.1.2
```

Start configuration synchronization from APV1 to APV2.

```
AN1(config)#synconfig to machine2
```

Note: If WebWall is enabled for the interface that is used by the "**synconfig**" command for configuration synchronization with peers, you need to add the corresponding accesslist rules to allow the traffic to come in through port 65519 on both Array appliances (both local and remote peers).

26.2.3.2.6 SDNS Configuration Synchronization

Administrators can synchronize SDNS configurations and BIND9 zone files except SDNS member configurations from a local APV appliance to remote peers.

In the following example, SDNS configurations and BIND9 zone files except SDNS member configurations on APV1 are synchronized to remote APV2.

1. Configure SDNS configuration synchronization on APV1.

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

Configure SDNS configuration synchronization on APV2.

```
AN2(config)#synconfig challenge synpassword
AN2(config)#synconfig peer peerlocal 172.16.83.180
AN2(config)#synconfig peer peerremote 172.16.83.120
```

Start SDNS configuration synchronization from APV1 to APV2.

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

26.2.3.2.7 NTP Time Synchronizer

The Network Time Protocol (NTP) time synchronizer enables the APV appliance to synchronize the system time with the specified NTP server.

After the NTP time synchronizer is enabled, the APV appliance will automatically synchronize the system time with the specified NTP server at the interval of about 15 minutes.

Attention:

- Do not change the system time of the APV appliance after enabling the NTP time synchronizer.
- APV appliance should be used as the NTP client rather than the NTP server.

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

1. Configure an NTP server.

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

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 172.16.85.12 4
ntp server 172.16.85.60 4
ntp on 0
time since restart:        20
time since reset:          20
packets received:          2
packets processed:         1
current version:           1
```

```
previous version:           0
declined:                   0
access denied:              0
bad length or format:       0
bad authentication:         0
rate exceeded:              0

     remote           local          st poll reach   delay   offset  disp
==============================================================================
=172.16.85.12    172.16.85.80     16   64    0    0.00000    0.000000 3.99217
*172.16.85.60    172.16.85.80      3   64    1    0.00189    0.000050 2.81735
```

Field	Meaning
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.
Declined	The number of declined requests for date setting due to the existence of an NTP program on the host.
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.

26.2.3.2.8 RESTful API

RESTful Application Programming Interface (API) is an alternative management method in addition to CLI, WebUI and XML-RPC.

With the RESTful API, the administrator can send RESTful API requests to the appliance via the HTTP or HTTPS protocol to perform Create, Read, Update, and Delete operations on system resources. The appliance will execute these operations and return the results in RESTful API responses. For the safety of RESTful API invocation, the administrator needs to include a valid credential in the RESTful API requests. Otherwise, the system will reject RESTful API requests.

Every manageable system resource is identified by a URL, in the format of “http(s):// management_ip:port/path[?query_name=value]”. For all RESTful APIs supported by the appliance, refer to the APV RESTful API Reference Guide.

The administrator can send RESTful API requests either using a RESTful API client installed on the Web browser or using a programming tool such as Java, Python, or Perl. For details, refer to the APV RESTful API User Guide.

Before using the RESTful API, you must enable the RESTful API service first. To enhance the system security, the administrator can configure RESTful API source IP restriction rules to control the sources that are allowed to access the RESTful API service.

In addition, to enhance the security of the RESTful API service, the administrator can customize its SSL settings:

- Set the SSL versions
- Set the SSL cipher suites
- Import the PEM-format certificate chain
- Import the client CA certificate
- Enable or disable the client authentication function
- Enable or disable the mandatory mode of the client authentication function

➢ **Configuration Example via CLI**

To set the RESTful API listening IP, execute the following command:

```
AN(config)#restapi ip 192.168.1.100
```

To configure a RESTful API source IP restriction rule, execute the following command:

```
AN(config)#restapi source 192.168.0.0 255.255.0.0
```

To enable the RESTful API service and define the protocol to be used and the port to be listened on, execute the following command:

```
AN(config)#restapi on https 9997
```

After the preceding configuration, the RESTful API service is enabled using the HTTPS protocol and listening on IP 192.168.1.100 and port 9997.

To create an administrator account with the API access privilege, execute the following command:

```
AN(config)#user rest password1 api
```

To set the SSL version supported by the RESTful API service, execute the following command:

```
AN(config)#restapi ssl settings protocol "TLSv12"
```

To set the SSL cipher suites supported by the RESTful API service, execute the following command:

```
AN(config)#restapi ssl settings ciphersuites
"ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
```

ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:AES128-GCM-SHA256:AES256-GCM-SHA384"

To import a certificate chain for the RESTful API service, execute the following command:

AN(config)#**restapi ssl import certificate ftp://10.8.6.20/cert/chain.pem**

To import a client CA certificate for the RESTful API service, execute the following command:

AN(config)#**restapi ssl import clientca ftp://10.8.6.20/cert/client.pem**

To enable the client authentication function for the RESTful API service, execute the following command:

AN(config)#**restapi ssl settings clientauth enable**

To enable the mandatory mode of the SSL client authentication function for the RESTful API service, execute the following command:

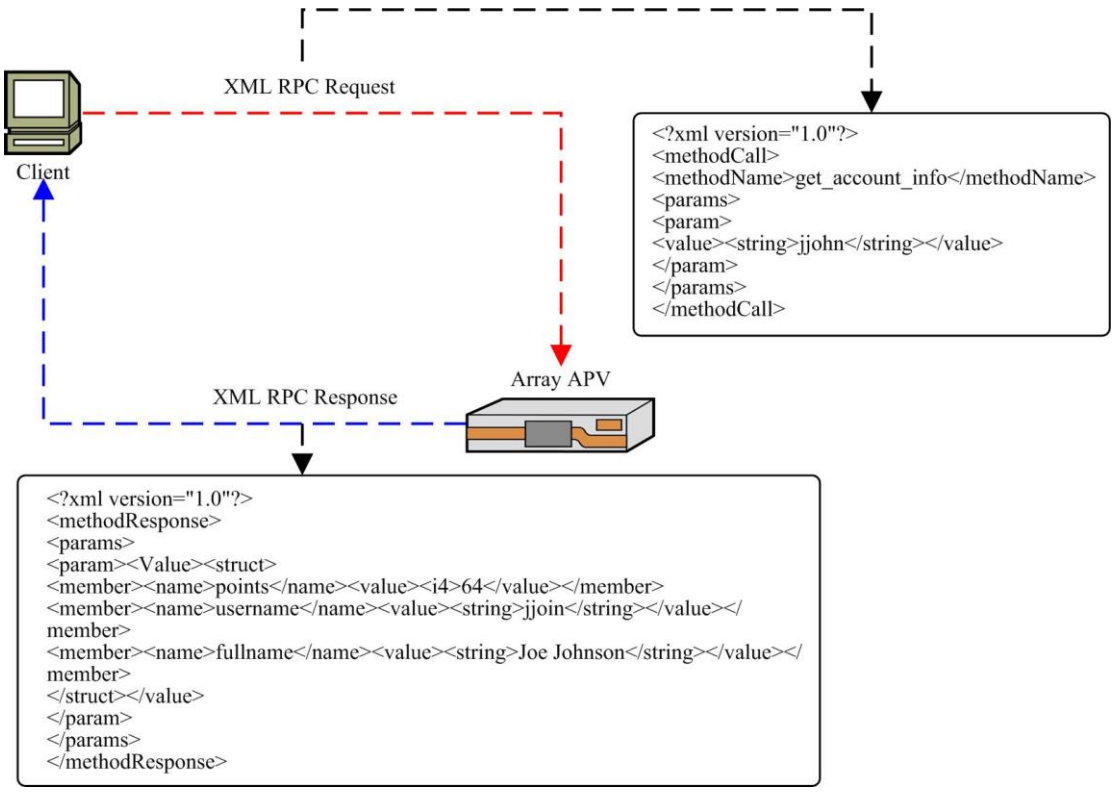
AN(config)#**restapi ssl settings authmandatory enable**

26.2.3.2.9 XML RPC

XML RPC allows clients to run some CLI commands remotely in ArrayOS. 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 figure below, Client sends an HTTP POST Request to APV. XML RPC message is the body of the HTTP Request, in which the commands to run and the commands' parameters are specified. Then, APV decodes the XML PRC message and executes the called commands. At last it returns the results formatted in XML to Client.



26.2.3.2.9.1.1 XML RPC Working Mechanism

To realize the communication between the Client and the APV appliance, a Perl script, called array_xmlrpc.pl, 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 APV 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>array</string>
  </value>
</member>
<member>
  <name>ip</name>
  <value>
    <string>10.1.1.1</string>
  </value>
</member>
<member>
  <name>port</name>
  <value>
    <int>80</int>
  </value>
</member>
<member>
  <name>maxconns</name>
  <value>
    <int>1000</int>
  </value>
</member>
<member>
  <name>hctype</name>
  <value>
    <string>tcp</string>
  </value>
</member>
<member>
  <name>hcup</name>
  <value>
    <int>1</int>
```

```
        </value>
      </member>
      <member>
        <name>hcdown</name>
        <value>
          <int>1</int>
        </value>
      </member>
    </struct>
  </value>
 </param>
</params>
</methodCall>
```

In this example, the first three lines (as below) constitute the HTTP Request Header, and the remaining part HTTP Request body.

```
POST    /cgi-bin/xmlrpc_server    HTTP/1.1
Content-Type: text/xml
Content-Length: xxx
```

In the first three lines of XML RPC message (as below), “slb_real” is the XML RPC method of the called command “**slb real** <*protocol*> <*name*> <*ip*> *[port] [maxconns] [hc_type] [hc_up] [hc_down]*”. XML PRC method is embedded in a <methodName> tag (Please refer to Appendix III, in which all XML RPC methods supported by APV 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, APV 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 APV, you need to configure two commands:

1. Turn on XML RPC.

```
AN1(config)#xml on https
```

Set the port for XML RPC to listen.

```
AN1(config)#xml port 9999
```

26.2.3.2.10 Remote Management

The Remote Management feature of the APV appliance allows administrators to access remote devices via Telnet & SSH.

To use the Telnet feature on the APV appliance, users can execute the command “**telnet** “host port”” as follows:

```
AN#telnet "'172.16.2.182 -4'"
Trying 172.16.2.182...
Connected to 172.16.2.182 -4.
Escape character is '^]'.
Trying SRA secure login:
User (root): array
```

```
Password:
[ SRA accepts you ].....succeed
```

To use the SSH feature on the APV appliance, users can execute the command "**ssh remote** *"user@hostname"*" as follows:

```
AN#ssh remote "root@172.16.85.240"
root@172.16.85.240's password:
Linux libh-server1 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686
GNU/Linux

Welcome to Ylmf_OS!
  * Information:    http://www.ylmf.com/

0 packages can be updated.
0 updates are security updates.

Last login: Wed Apr 20 00:39:35 2011 from 10.3.46.1
root@libh-server1:~#
```

26.2.3.2.11 Flight Deck

The APV appliance monitors a variety of useful statistics that provide a good indication of performance, user and network activity. The APV appliance provides a graphical interface that can be used to easily monitor various statistics and get a comprehensive picture of the status of the APV appliance. This graphical interface is called the Flight Deck.

The Flight Deck is an additional pop up browser window that, once set, can display a wide range of real time network operational data. Across the top of the browser window, you will discover readouts concerning the server health, request rate, cache hits and system usage. Moving to the left side of the window, you will find reading for the TCP, HTTP and SSL connections. The three connection figures sum up to total used "TCP pcb" displayed in the output of the "**show memory**" command. Sometimes, a pair of TCP connections is created for the same client request, for example, an SLB client request normally will generate two connections, one is from the client to APV appliance, and the other is from the APV appliance to the server.

The central portion of the Flight Deck is occupied by two configurable graphs. Simply use the pull-down menu to choose the desired data you wish to track in the real time graphical output.

You can access the Flight Deck from the APV 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 APV 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 APV 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 APV 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.

Note: For the sake of security, it is strongly recommended to modify the default SNMP community string to avoid possible system information interception.

The following statistics are available for viewing in the graphs:

- **TCP**

- **Active Opens**

  The number of times CLICKTCP connections have made to a direct transition to the SYN-SENT state from the CLOSED state

- **Passive Opens**

  The number of times CLICKTCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state

- **Open Failures**

  The number of times CLICKTCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state

- **Established Conns**

  The number of CLICKTCP connections for which the current state is either ESTABLISHED or CLOSE- WAIT

- **Resets Received**

  The number of times CLICKTCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state

- **Resets Sent**

  The number of CLICKTCP segments sent containing the RST flag

- **Retransmits**

  The total number of segments retransmitted - that is, the number of CLICKTCP segments transmitted containing one or more previously transmitted octets

- **Packet Errors**

  The total number of segments received in error (for example, 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 APV 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 APV 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

26.2.3.2.12 IPv6 External Link Conversion Tool

The IPv6 external link conversion tool is developed for the scenario when IPv4 external links are included in an IPv6 site, that is, the IPv6 skylight issue. The IPv6 external link conversion tool can parse out the IPv4 links contained in the content of the specified IPv6 site, and automatically generate the corresponding SLB configuration to help converse the IPv4 links to IPv6 links. The generated SLB configuration can be modified and then applied to the APV appliance.

Currently, administrators can use the IPv6 external link conversion tool online through the WebUI (**Admin Tools** > **IPv6** > **IPv6 Tool** > **Online Translation**), or download the tool through the WebUI (**Admin Tools** > **IPv6** > **IPv6 Tool** > **Windows Installer Download**) to use it in the local. For details on how to use this function, please refer to the IPv6 External Link Conversion Tool User Guide.

26.2.3.2.13 TCP RESET Error Code Statistics

The system provides the TCP RESET error code statistics to facilitate troubleshooting for administrators.

Administrators can view the TCP RESET error code statistics on WebUI by selecting **Admin Tools > Troubleshooting > TCP RESET Error Code Statistics**.

For explanation of TCP RESET error codes, please refer to the RESET Error Code Reference Guide (clicking the **Documentation** button on the top bar and select the **RESET Error Code Reference Guide** menu).

26.3 Administrator Configuration and Privilege Management

26.3.2Overview

The APV appliance allows creating system administrators and specifying the access control level (Enable or Config) for administrators. If more flexible control over the administrator privilege is needed, the role-based privilege management can be used to control the CLI commands an administrator can execute.

26.3.3System Administrator

The APV appliance allows creating two types of administrator: administrator of the Enable level and administrator of the Config level. Administrators of the Enable level can execute only the commands allowed by the Enable and User levels. Administrators of the Config level can execute all the commands allowed by the Config, Enable and User levels.

26.3.4Role-based Privilege Management

The role-based privilege management function can control the CLI commands that an administrator can execute by assigning roles to the administrator, thus realizing more flexible administrator privilege control.

One administrator can be assigned one or more roles, and the logic among the multiple roles is "OR". If any role assigned to an administrator permits the execution of a command, then the administrator can execute this command; if all roles assigned to an administrator deny (or none of the roles permits) the execution of a command, the administrator cannot execute this command. If an administrator is not assigned any role, the administrator is allowed to execute all the commands of his access control level.

Role is a set of privilege rules. A privilege rule consists two parts: rule string and operation privilege.

- **Rule String**

Rule string defines one or a group of command configurations. Three forms of rule string are supported:

- Complete form: slb real http 'r1' 172.16.2.250 80 1 tcp 3 3
- Incomplete form with some parameters: slb real http 'r1'
- Incomplete form with a part of the command body: slb real

To configure eligible rule strings, please pay attention to the following notes:

- The command body cannot be abbreviated.
- The parameters are case-sensitive.
- The whole rule string needs to be enclosed in double quotes. If double quotes are further needed by some parameters in the rule string, please use single quotes instead.
- If the entered string has multiple consecutive spaces, the spaces will be regarded as one space.

The system will check the command entered by the administrator against the rule strings from the first letter. If the command is identical with or comprises a rule string, the command is regarded as matching the rule string and will follow the rule.

- **Operation Privilege**

There are two kinds of operation privileges: “permit” and “deny”. The privilege rules are consequently divided into “permit” rules and “deny” rules, which are respectively used to control that one command or a group of commands can or cannot be executed by certain roles. If a role is not configured with any “permit” rule, the role is not permitted to execute any command.

When a command matches a “permit” rule and a “deny” rule at the same time, the system will assume that the “deny” rule has a higher priority.

For example:

role1:

- “permit” rule: “no slb group”
- “deny” rule: “no slb group method”

The system will deny the execution of commands starting with “no slb group method” by role1, but the execution of other commands starting with “no slb group” is permitted.

role2:

- “permit” rule: “no slb group method”
- “deny” rule: “no slb group”

Because the “deny” rule has a higher priority, the system will deny the execution of the commands starting with “no slb group” by role2, although the “permit” rule allows the execution of the commands starting with “no slb group method”.

To disallow a role from executing configuration, display and deletion operations about a feature (for example LLB), please configure the following privilege rule for this role:

- “deny” rule: “llb”
- “deny” rule: “no llb”
- “deny” rule: “show llb”
- “deny” rule: “clear llb”

Note:

- The system provides two pre-defined roles: “SLB” and “NETWORK”. You can execute the “**show role predefined**” command to see the privilege rules of the two roles.
- Monitoring WebUI requires the privilege for executing the “**show…**” CLI commands. Therefore, on the WebUI, after defining a role for an administrator account, you need to configure a “Permit” privilege rule to allow this role to perform “**show…**” commands.

26.3.5Administrative Privilege Separation

The administrative privilege separation function satisfies the requirement of dividing operation privilege for various scenarios. To support this function, the system provides three predefined roles and their corresponding rules. The predefined roles are administrator (role_administrator), operator (role_operator) and auditor (role_auditor).

When the administrative role separation function is enabled, the system automatically creates three administrative accounts, namely, administrator, operator and auditor, and automatically assigns the users their respective roles. When this function is on, newly created users cannot execute CLI until a role is assigned to them.

When this function is disabled, users created under the administrative role separation mode cannot log in through SSH connection or WebUI.

Privileged operations of the three administrative accounts are as follows:

- Administrator: can change the passwords of all users except other administrators and can execute all CLI except log operations.
- Operator: can change only its own password and can execute all CLI except log, role and user operations.
- Auditor: can change only its own password and can execute “show” CLI and part of the log operations.

For more details about the privileged operations of the three administrative accounts, execute the commands “**show role name**”, “**show role permit**” and “**show role deny**” to check.

26.3.6Configuration Examples

26.3.6.1 Creating System Administrator

Execute the following command to add an administrator account and specify the access control level:

user <*user_name*> <*password*> *[level]*

For example:

AN(config)#**user admin1 abcabc config**

26.3.6.2 Configuring Role-based Privilege Management

1. Execute the following command to add a role:

role name <*role_name*>

For example:

```
AN(config)#role name role1
```

Execute the following commands to configure the privilege rules for the role:

role deny <*role_name*> <*filter_string*>
role permit <*role_name*> <*filter_string*>

For example:

```
AN(config)#role deny role1 "clear config"
AN(config)#role permit role1 "show running config"
```

Execute the following command to assign the role to the administrator:

role user <*user_name*> <*role_name*>

For example:

```
AN(config)#role user admin1 role1
```

26.4 Segment Management

The segment management function achieves the purpose of SLB/SSL service isolation in multi-tenant environment. A physical application can serve multiple tenants simultaneously, and tenant-specific user management systems and service delivery systems are isolated from each other. This will greatly decrease the operation and maintenance cost of services and improve resource utilization.

The number of segments supported by the system depends on the system memory. 1024 segments are supported at most, that means, a physical appliance can provide isolated SLB and SSL services for a maximum of 1024 tenants simultaneously.

26.4.2Segment Administrator

Segment administrator accounts should be created by system administrators and then associated with segments. Each segment must be assigned at least one segment administrator and other resources such as IP addresses, interfaces and VLANs. Each segment can only use resources assigned to it. Each segment administrator account can be associated with only one segment, while each segment can be managed by multiple segment administrators. The total number of segment administrator accounts should not exceed 2048.

System administrators can use the "**segment user**" command to create segment administrator accounts. While creating a segment administrator account, the system administrator can specify its password and access level:

- enable: segment administrator accounts with this privilege level are only allowed to execute enable-level CLI commands supported in the segment.
- config: segment administrator accounts with this privilege level are allowed to execute all config-level CLI commands supported in the segment.

- api: segment administrator accounts with this privilege level are allowed to access API-based Web service supported in the segment, but are not allowed to execute any CLI commands.

After login with the created segment administrator account, the segment administrator will enter the segment associated with the account to perform and manage configurations. The password of a segment administrator account can be modified via the "**segment passwd**" command in the system mode or the "**password**" command in the segment mode.

Note: If the system administrator needs to modify resources assigned to segments, such as segment names, segment administrator accounts and IP addresses, the modifications must be saved by executing **"write memory**" in both the system mode and segment mode.

26.4.3Functions Supported in Segments

A segment has its own CLI commands for configuring SLB and SSL, and all segments support the same set of CLI commands. For config-level segment administrator accounts, all CLI commands supported in the segment will be displayed with the input of the question mark in config mode.

With the segment management function, each segment can have its own default route and static route. The system will generate Eroutes for each segment based on the default route and static route. The system administrator can execute the "**show segment ip eroute**" command to view the Eroutes generated for each segment.

In addition, the segment management function supports NAT rules, which allows overlap of network segments and IP addresses between segments. System administrators can configure NAT rules for each segment using the "**segment nat**" command.

Note:

- Non-service traffic such as health check and ping must be forwarded via static routes or direct routes but not default routes within segments.
- Except real services, within segments, the system does not support cooperation with externals servers whose URL addresses consist of domain names instead of IP addresses, for example, importing or backup of files via FTP servers, downloading CRL files from LDAP servers, sending alert mails to external mail servers, verifying certificates based on URL addresses in certificates.

Appendix I Abbreviations

Acronym	Full Spelling
AAA	Authentication, Authorization & Accounting
ACL	Access Control List
ADC	Application Delivery Controller
API	Application Programming Interface
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
ATCP	Array TCP
BGP	Border Gateway Protocol
CA	Certificate Authority
CDN	Content Distribution Network
CDP	CRL Distribution Point
CGI	Common Gateway Interface
CLI	Command Line Interface
CNAME	Canonical Name
CPS	Connections Per Second
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRL	Certificate Revocation List
CRS	Core Rule Set
CSR	Certificate Signing Request
DMZ	DeMilitarized Zone
DNS	Domain Name System
DoS	Denial Of Service
DPS	Dynamic Proximity System
FFO	Fast Failover
FIFO	First-In First-Out
FTP	File Transfer Protocol
FTPS	FTP over SSL
GMT	Greenwich Mean Time
GRE	Generic Routing Encapsulation
GSLB	Global Server Load Balance (also known as SDNS)
HA	High Availability
HC	Health Check
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Sockets Layer
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6

Acronym	Full Spelling
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IIS	Internet Information Server
IMS	Information Management System
IP	Internet Protocol
ISP	Internet Service Provider
LACP	Link Aggregation Control Protocol
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LED	Light Emitting Diode
LLB	Link Load Balancing
Local DNS	Local Domain Name System
MAC	Media Access Control
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extensions
MNET	Multi-Netting
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NDP	Neighbor Discovery Protocol
NIC	Network Interface Card
NMS	Network Management Station
NTP	Network Time Protocol
NUMA	Non-uniform Memory Access
OCSP	Online Certificate Status Protocol
OID	Object Identifier
OSI	Open System Interconnection
OSPF	Open Shortest Path First
OSPFv2	Open Shortest Path First version 2
OSPFv3	Open Shortest Path First version 3
OWA	Outlook Web Access
PCI	Peripheral Component Interface
PEM	Privacy Enhanced Mail
PHY	Physical Layer
PKI	Public Key Infrastructure
PLR	Packet Loss Rate
POP3	Post Office Protocol - Version 3
PPP	Point-to-Point Protocol
PPTP	Point-to-Point Tunneling Protocol
PST	Pacific Standard Time
QoS	Quality of Service
RADIUS	Remote Authentication Dial-In User Service

Acronym	Full Spelling
RAM	Random Access Memory
RDP	Remote Desktop Protocol
RFC	Request For Comments
RHI	Route Health Injection
RIP	Routing Information Protocol
RIPv1	Routing Information Protocol version 1
RIPv2	Routing Information Protocol version 2
RIPng	RIP next generation
RPS	Requests Per Second
RTS	Return to Sender
RTSP	Real Time Streaming Protocol
RTT	Round Trip Time
SCP	Session Control Protocol
SDNS	Smart DNS (also known as GSLB)
SIP	Session Initiation Protocol
SLB	Server Load Balancing
SMTP	Simple Mail Transfer Protocol
SNI	Server Name Indication
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSF	Stateful Session Failover
SSH	Secure Shell Protocol
SSI	Single System Image
SSL	Secure Sockets Layer
SSLv3	Secure Sockets Layer version 3
TACACS	Terminal Access Controller Access Control System
TCI	Tag Control Information
TCL	Tools Command Language
TCP	Transmission Control Protocol
TCPS	TCP with SSL
TELNET	Terminal Emulation Protocol in a TCP/IP Environment
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security Protocol
TPID	Tag Protocol Identifier
TTL	Time to Live
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VCID	Virtual Cluster ID
VIP	Virtual IP
VLAN	Virtual Local Area Network
VOD	Video On Demand

Acronym	Full Spelling
VoIP	Voice over Internet Protocol
VRRP	Virtual Router Redundancy Protocol
VXLAN	Virtual eXtensible Local Area Network
VNI	VXLAN Network Identifier
VTEP	VXLAN Tunnel End Point
WebUI	Web User Interface
WELF	WebTrends Enhanced Log Format
XSS	Cross Site Scripting

Appendix II XML RPC Methods

The following table lists all XML RPC methods supported by APV. 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.

<table>
<tr><th colspan="5">Generic XML RPC Method</th></tr>
<tr><th>Method Name</th><th>Command</th><th>{Parameter Name, Parameter Type}</th><th>Optional</th><th>Notice</th></tr>
<tr><td>arrayos_cli_enable</td><td>All commands in Enable mode</td><td rowspan="2">{num, int}, {cli_string0, string}, {cli_string1, string}, {cli_string2, string}, {cli_string3, string} ……</td><td rowspan="2">num</td><td rowspan="2">If no “num” value is given, it defaults to 1 and “cli_string0” should be configured.
The names of CLI strings should start from “cli_string0” and end at “cli_string{n-1}”.
If some intermediate CLI strings are missing, the XML RPC system will just ignore and not complain.</td></tr>
<tr><td>arrayos_cli_config</td><td>All commands in Config mode</td></tr>
<tr><td>arrayos_cli_config_with_input</td><td>All commands in Config mode</td><td>{cli_string, string}, {num, int}, {input_string0, string}, {input_string1, string} ...</td><td>num, input_string0, input_string1, ...</td><td>If an interactive CLI command (for example, a command that requires input of “YES” to continue the command execution) is called, this method should be used. You can use this method to execute only one command once.
If no “num” value is given, it defaults to 1. “input_string” starts from “input_string0” to “input_string(num-1)”. If “input_string(n)” is not included, it defaults to null. However, if the CLI must require a valid input at this place, the invocation may be hung or an error may be returned. You can use this method to execute only one command once.</td></tr>
</table>

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
slb_real	slb real	{protocol, string}, {name, string}, {ip, string}, {port, int}, {maxconns, int}, {hctype, string}, {hcup, int}, {hcdown, int}, {sess_timeout, int}	port maxconns hctype hcup hcdown、sess_timeout	“sess_timeout” parameter is valid for UDP real services only. “name” is the name of real service.
no_slb_real	no slb real	{protocol, string}, {name, string}		
slb_virtual	slb virtual	{protocol, string}, {name, string}, {ip, string}, {port, int}, {noarp, int}		“noarp” is of type integer. “name” is the name of virtual 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	“threshold” and “threshold_rr” are optional parameters for “lc” and “sr” methods. “cookie” and “path_attr” are optional parameters for “ic” method. “cookie” is required and “offset” is optional for “rc” method. “header” is a required parameter for “hh” method. “sess_timeout” is an optional parameter for “sslsid” method. Parameters in the XML RPC message which do not apply to the given method are ignored.
no_slb_gro	no slb group	{name, string}		

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
up_method	method			
slb_group_member	slb group member	{name, string}, {real_name, string} {value, int}	value	"name" is the name of the group and "real_name" is the name of real service. "value" is an optional parameter and applies only to certain methods.
no_slb_group_member	no slb group member	{name, string}, {real_name, string}		"name" is the name of group and "real_name" is the name of real_service.
slb_policy	slb policy	{policy, string}, {name, string}, {virtual_name, string}, {group_name, string}, {real_name, string}{regex, string}, {policy_string, string}, {precedence, int}, {ip,string}, {netmask, string}		"policy" is the type of policy. The values of "policy" can be one of "static", "default", "regex", "icookie", "rcookie", "persistent url", "persistent cookie", "qos cookie", "qos hostname", "qos url", and "qos network". "policy" and "virtual_name" are required for all policies. "group_name" is required for all policies except "static" policy. "name" is required for all policies except "static" and "default". "real_name" is required only for "static" policy. "precedence" is required for "regex", "icookie", "rcookie", "persistent url", "persistent cookie", "qos cookie", and "qos hostname", "qos url" and "qos network". "regex" is required for "regex" policy. "policy_string" is required for "persistent url", "persistent cookie", "qos url", "qos cookie", and "qos

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
				hostname". "ip" and "netmask" are required for "qos network" policy.
no_slb_policy	no slb policy	{policy, string}, {name, string}, {virtual_name, string}		policy" is the type of policy, "name" is the name of policy and "virtual_name" is the name of "virtual_service". "virtual_name" is required for "default" and "static" policies and "name" is required for the rest of them.
cache_evict	cache evict	{hostname, string}, {regex, string}		
cluster_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_pree	no cluster virtual preempt	{vcid, int}, {interface, string}		

Specific XML RPC Method				
Method Name	**Command**	**{Parameter Name, Parameter Type}**	**Optional**	**Notice**
mpt				
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_interval	no cluster virtual interval	{vcid, int}, {interface, string}		
cluster_virtual_auth	cluster virtual auth	{vcid, int}, {interface, string}, {auth, int}, {auth_passwd, string}		“auth” value should be 0 or 1. If it is set to 1, “auth_passwd” parameter is required.
no_cluster_virtual_auth	no cluster virtual auth	{vcid, int}, {interface, string}		