AT&T

Services Configuration Guide, 17.2.0

November 2017

Supporting AT&T Vyatta Network Operating System

Contents

Copyright Statement

The training materials and other content provided herein for assistance in training on the Vyatta vRouter may have references to Brocade as the Vyatta vRouter was formerly a Brocade product prior to AT&T's acquisition of Vyatta. Brocade remains a separate company and is not affiliated to AT&T.

About This Guide

This guide describes how to configure the DHCP, DHCPv6, DNS, flow monitoring, NHRP, sFlow, TWAMP, and port monitoring services on the AT&T Vyatta Network Operating System (referred to as a virtual router, vRouter, or router in the guide).

DHCP

DHCP overview

DHCP allows dynamic assignment of reusable IP addresses and other configuration information to DHCP clients. This assignment reduces costs, configuration effort, and management burden associated with Internet access. On the other hand, it also increases network and service overhead.

In DHCP, the server assigns an IP address and other configuration parameters to a client for a limited period of time. This period of time is called the lease. The lease is valid for the period you configure on the AT&T Vyatta vRouter or until the client explicitly relinquishes the address.

To use the DHCP service, you define a pool of IP addresses for each subnet assigned by the DHCP server. Each DHCP address pool is mapped to a subnet associated with the system. For each address pool, you can specify the length of time an address is valid (its lease duration). The default lease duration is 24 hours. You can also specify a number of different servers (for example DNS, WINS, SMTP, and others) that are available to clients on the subnet.

To create an IP address pool for clients on a subnet to which the router is not directly connected (that is, without having an interface into that network), you can use service dhcp-server listento interface <dp-interface> *(page 42)*. See Configuring for networks indirectly connected to the system *(page 25)*.

You can statically map an IP address to the MAC address of a device. The DHCP service listens on UDP port 67 for lease requests from DHCP clients. The request packet allows the system to determine the interface on which the client is located. It then assigns an IP address from the appropriate pool and binds it to the client.

In addition to providing a DHCP server, individual interfaces on the AT&T Vyatta vRouter can be configured as DHCP clients. For details, see the AT&T Vyatta vRouter documentation for the interface you are interested in configuring as a DHCP client.

The AT&T Vyatta vRouter also supports DHCP relay.

A DHCP relay agent receives DHCP packets from DHCP clients and forwards them to a DHCP server. This allows you to place DHCP clients and DHCP servers on different networks; that is, across router interfaces.

The relay agent is configured with addresses of DHCP servers to which they should relay client DHCP message. The relay agent intercepts the broadcast, sets the gateway address (the **giaddr** field of the DHCP packet) and, if configured, inserts the Relay Agent Information option (option 82) in the packet and forwards it to the DHCP server.

The DHCP server echoes the option back verbatim to the relay agent in server-to-client replies, and the relay agent strips the option before forwarding the reply to the client.

DHCP classless static routes

By default, the AT&T Vyatta vRouter enables classless static routes through DHCP using option 121. For more information about classless static routes and option 121, refer to RFC 3442 at https://tools/ietf.org/html/rfc3442.

DHCP packets from the vRouter include the classless static option in the parameter request list. When replies from a DHCP server include this option, the default router (option 3) route is ignored per the RFC. Classless static routes may include a default route that is installed similar to other option 121 routes.

> **Note:**
>
> The classless static route option is only available on the AT&T Vyatta vRouter IPv4 DHCP client. A AT&T Vyatta vRouter DHCP server does not support the classless static routes option.

For the occasion when you must disable the classless static route option on the vRouter DHCP client, use the following commands

- interfaces bridge <brx> dhcp-options no-rfc3442 *(page 34)*—Disables support for the DHCP classless static route option for a bridge group.
- `interfaces dataplane` *interface-name* **`dhcp-options no-rfc3442`**—Disables support for the classless static route option for DHCP on a data plane interface.
- `interfaces dataplane` *interface-name* **`vif`** *vif-id* **`dhcp-options no-rfc3442`**—Disables support for the classless static route option for DHCP on a virtual interface.

For information on data plane and virtual interfaces, refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.

> **Note:** Before using these commands, configure the respective bridge or data plane address for the DHCP client.

DHCP configuration

DHCP configuration includes enabling the DHCP server, setting up the DHCP servers for failover, configuring DHCP address pools, setting up DHCP relay, and setting up additional DHCP configuration parameters.

Enabling the DHCP server

To use the DHCP server on the AT&T Vyatta vRouter, you must enable the DHCP service.

To enable the DHCP service, perform the following steps in configuration mode.

Table 1: Enabling the DHCP service

Step	Command
Enable DHCP.	`vyatta@R1# set service dhcp-server`

In addition, at least one DHCP shared network (address pool) must be configured.

Creating a static mapping

Situations exist in which it makes sense to map a specific IP address to a specific host rather than dynamically assign an IP address from a pool of addresses. This mapping is known as a “static mapping.”

A static mapping is defined by using the static-mapping option of the service dhcp-server configuration node. The following example shows how to create a static mapping by associating the 172.16.0.101 IP address to the device with a MAC address of 00:15:c5:b3:2e:65.

Table 2: Creating a static mapping

Step	Command
Create a static mapping called lab and specify the static IP address.	`vyatta@R1# set service dhcp-server static-mapping lab ip-address 172.16.0.101`
Specify the associated MAC address within the lab static mapping.	`vyatta@R1# set service dhcp-server static-mapping lab mac-address 00:15:c5:b3:2e:65`
Commit the changes.	`vyatta@R1# commit`

Step	Command

Show the configuration.	`vyatta@R1# show service dhcp-server`
	` static-mapping lab {`
	`     ip-address 172.16.0.101`
	`     mac-address 00:15:c5:b3:2e:65`
	` }`

Configuring DHCP address pools

Configure DHCP address pools for the system to act as a DHCP server for the network.

Configuring for networks directly connected to the system

Table 1 *(page 23)* shows how to create three address pools:

- dp0p1p1_POOL. This address pool serves the 172.16.0.0/24 subnet, which is connected to the dp0p1p1 interface. The lease time remains at the default, 24 hours (86,400 seconds). This address pool uses the DNS name server at 172.16.0.34.
- dp0p1p2_30_POOL. This address pool serves the 10.10.30.0/24 subnet, which is connected directly to the dp0p1p2 interface. The lease time remains at the default, 24 hours (86,400 seconds). This address pool uses the DNS name server at 10.10.40.34, which is directly connected to dp0p1p2.40 (that is, dp0p1p2 virtual interface [vif] 40).
- dp0p1p2_40_POOL. This address pool serves the 10.10.40.0/24 subnet, which is connected to the dp0p1p2.40 interface. The lease time remains at the default, 24 hours (86,400 seconds). This address pool uses the DNS name server at 10.10.40.34, which is connected to dp0p1p2.40.

In all these pools, the range of addresses is configured for .100 through .199.

The following figure shows the sample address pool configuration.

Figure 1: DHCP address pool configuration

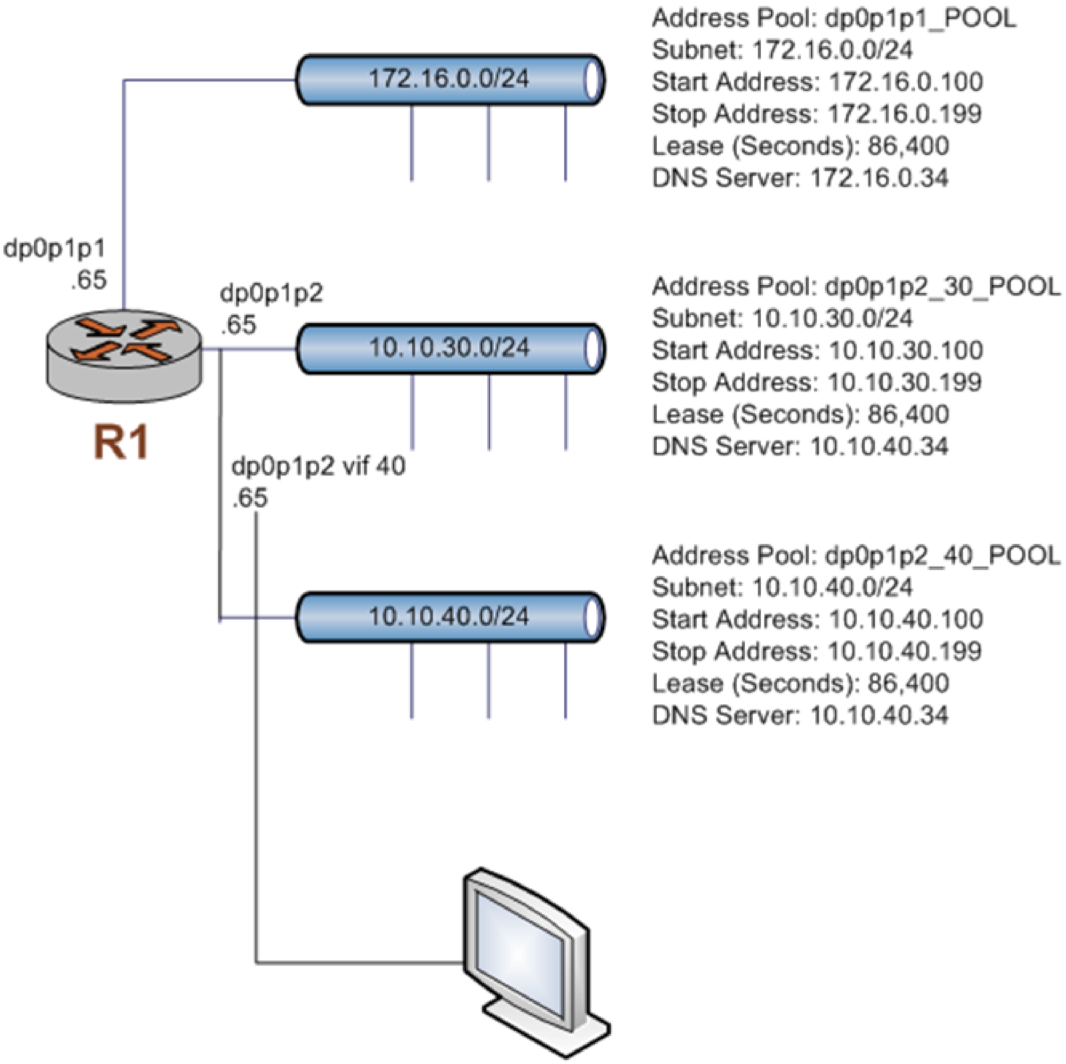


Table 3: Configuring DHCP address pools

Step	Command
Create the configuration node for dp0p1p1_POOL on the 172.16.0.0/24 subnet. Specify the start and stop IP addresses for the pool.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p1_POOL subnet 172.16.0.0/24 start 172.16.0.100 stop 172.16.0.199`
Specify the default router for dp0p1p1_POOL.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p1_POOL subnet 172.16.0.0/24 default-router 172.16.0.65`

Step	Command
Specify a DNS server for dp0p1p1_POOL.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p1_POOL subnet 172.16.0.0/24 dns-server 172.16.0.34`
Create the configuration node for dp0p1p2_30_POOL on the 10.10.30.0/24 subnet. Specify the start and stop IP addresses for the pool.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_30_POOL subnet 10.10.30.0/24 start 10.10.30.100 stop 10.10.30.199`
Specify the default router for dp0p1p2_30_POOL.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_30_POOL subnet 10.10.30.0/24 default-router 10.10.30.65`
Specify a DNS server for dp0p1p2_30_POOL.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_30_POOL subnet 10.10.30.0/24 dns-server 10.10.40.34`
Create the configuration node for dp0p1p2_40_POOL on the 10.10.40.0/24 subnet. Specify the start and stop IP addresses for the pool.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_40_POOL subnet 10.10.40.0/24 start 10.10.40.100 stop 10.10.40.199`
Specify the default router for dp0p1p2_40_POOL.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_40_POOL subnet 10.10.40.0/24 default-router 10.10.40.65`
Specify a DNS server for dp0p1p2_40_POOL.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_40_POOL subnet 10.10.40.0/24 dns-server 10.10.40.34`
Commit the changes.	`vyatta@R1# commit`

Step	Command

Show the configuration.	`vyatta@R1# show service dhcp-server`
	`  shared-network-name dp0p1p1_POOL {`
	`  subnet 172.16.0.0/24 {`
	`   default-router 172.16.0.65`
	`   dns-server 172.16.0.34`
	`   start 172.16.0.100 {`
	`    stop 172.16.0.199`
	`   }`
	`  }`
	` }`
	` shared-network-name dp0p1p2_30_POOL {`
	`  subnet 10.10.30.0/24 {`
	`   default-router 10.10.30.65`
	`   dns-server 10.10.40.34`
	`   start 10.10.30.100 {`
	`    stop 10.10.30.199`
	`   }`
	`  }`
	` }`
	` shared-network-name dp0p1p2_40_POOL {`
	`  subnet 10.10.40.0/24 {`
	`   default-router 10.10.40.65`
	`   dns-server 10.10.40.34`
	`   start 10.10.40.100 {`
	`    stop 10.10.40.199`
	`   }`
	`  }`
	` }`

Show the interface configuration.	`vyatta@R1# show interfaces`
	` dataplane dp0p1p1 {`
	`     address 172.16.0.65/24`
	`     hw-id 00:0c:29:42:05:2b`
	` }`
	` dataplane dp0p1p2 {`
	`     address 10.10.30.65/24`
	`     hw-id 00:0c:29:42:05:35`
	`     vif 40 {`
	`         address 10.10.40.65/24`
	`     }`
	` }`

Configuring for networks indirectly connected to the system

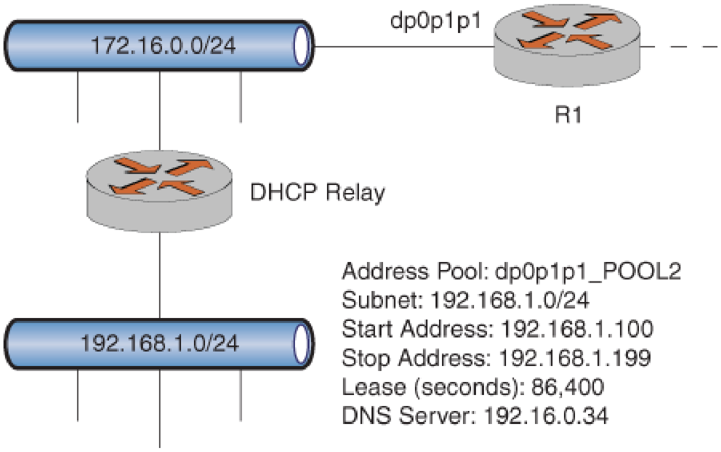
Table 1 *(page 26)* shows how to create an address pool (dp0p1p1_POOL2) for clients that are indirectly connected to the R1 vRouter, as shown below.

The dp0p1p1_POOL2 address pool serves the 192.168.1.0/24 subnet, which is on a different subnet than the subnet to which the dp0p1p1 data plane interface is connected.

The lease time remains at the default, 24 hours (86,400 seconds). This address pool uses the DNS name server at 172.16.0.34.

The following figure shows the sample address pool configuration.

Figure 2: DHCP address pool configuration for clients indirectly connected to the DHCP server host



Address Pool: dp0p1p1_POOL2
Subnet: 192.168.1.0/24
Start Address: 192.168.1.100
Stop Address: 192.168.1.199
Lease (seconds): 86,400
DNS Server: 192.16.0.34

To configure the dp0p1p1_POOL2 DHCP address pool, perform the following steps in configuration mode.

Table 4: Configuring DHCP address pools

Step	Command

Configure the router interface to listen to DHCP messages.	`vyatta@R1#set service dhcp-server listento`
	` interface dp0p1p1`

Create a shared network and associate it with the 192.168.1.0/24 subnet.	`vyatta@R1# set service dhcp-server`
	` shared-network-name dp0p1p1_POOL2 subnet`
	` 192.168.1.0/24`

Show the configuration.	`vyatta@R1# show service dhcp-server`
	`dhcp-server {`
	`    listento {`
	`        interface dp0p1p1`
	`    }`
	`    shared-network-name dp0p1p1_POOL2 {`
	`        subnet 192.168.1.0/24 {`
	`            dns-server 192.168.1.100`
	`            lease 86400`
	`            start 192.168.1.100 {`
	`                stop 192.168.1.199`
	`            }`
	`        }`
	`    }`
	`}`

Step	Command

Show the interface configuration.	`vyatta@R1# show interfaces`
	`interfaces {`
	`    dataplane dp0p1p1 {`
	`        address 172.16.0.0/24`
	`    }`
	`    loopback lo`
	`}`

Commit the changes.	`vyatta@R1# commit`

Setting up DHCP servers for failover

The AT&T Vyatta vRouter also provides a failover feature to allow for DHCP redundancy on a given subnet.

In a failover configuration, two DHCP servers act as failover peers, with one of the peers designated as the primary and the other as the secondary. For DHCP failover to work, the following conditions must be met.

- Both peers must be AT&T Vyatta vRouters and must be running the same version of Vyatta software.
- Each server must be configured to point to the other as the failover peer.
- The time on the servers must be exactly synchronized.
- The start-stop range must have at least one IP address for each subnet that has not been either excluded (by using service dhcp-server shared-network-name name subnet ipv4net exclude ipv4 *(page 52)*) or statically mapped (by using service dhcp-server static-mapping mapname *(page 68)*).

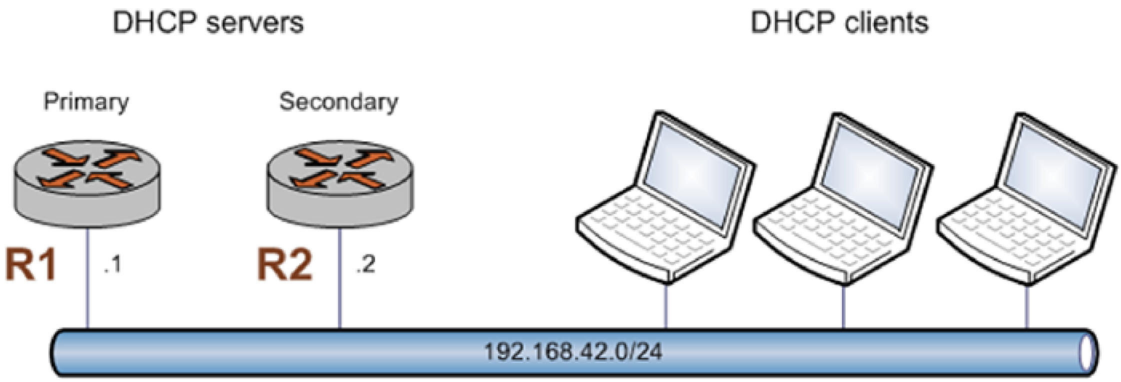
The system times should be synchronized before configuring DHCP failover. Use of NTP time synchronization is highly recommended. However, if difficulties arise because of incorrect system times, disable NTP, reset the times correctly, and then re-enable NTP.

Note that DHCP leases are assigned only in failover configurations if proper communication is established between the two failover peers. If the configuration is incorrect (if, for example, one failover peer is configured but the other is not), DHCP leases are not dispersed.

Also note that statically mapped addresses are not renewed by a failover server unless they are explicitly defined on that server by using service dhcp-server static-mapping mapname *(page 68)*.

The following figure shows the sample DHCP server failover configuration.

Figure 3: DHCP server failover configuration



To configure R1 as the primary DHCP server in this failover scenario, perform the following steps in configuration mode on R1.

Table 5: Setting up DHCP failover on R1

Step	Command
Create the configuration node for DHCP1 on the 192.168.42.0/24 subnet. Specify the start and stop IP addresses for the pool.	`vyatta@R1# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 start 192.168.42.100 stop 192.168.42.199`
Specify the default router for DHCP1.	`vyatta@R1# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 default-router 192.168.42.254`
Specify a DNS server for DHCP1.	`vyatta@R1# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 dns-server 192.168.42.253`
Specify the local IP address for the DHCP server for failover.	`vyatta@R1# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover local-address 192.168.42.1`
Specify the IP address of the peer DHCP server for failover.	`vyatta@R1# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover peer-address 192.168.42.2`
Specify the role that the DHCP server plays in the failover group.	`vyatta@R1# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover status primary`
Specify the name of the failover group.	`vyatta@R1# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover name FAILOVER`
Commit the changes.	`vyatta@R1# commit`

Step	Command

Show the configuration.	`vyatta@R1# show service dhcp-server shared-network-name DHCP1`
	` shared-network-name DHCP1 {`
	`  subnet 192.168.42.0/24 {`
	`   default-router 192.168.42.254`
	`   dns-server 192.168.42.253`
	`   failover {`
	`    local-address 192.168.42.1`
	`    name FAILOVER`
	`    peer-address 192.168.42.2`
	`    status primary`
	`   }`
	`   start 192.168.42.100 {`
	`    stop 192.168.42.199`
	`   }`
	`  }`
	` }`

To configure R2 as the secondary DHCP server in this failover scenario, perform the following steps in configuration mode on R2.

Table 6: Setting up DHCP failover on R2

Step	Command
Create the configuration node for DHCP1 on the 192.168.42.0/24 subnet. Specify the start and stop IP addresses for the pool.	`vyatta@R2# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 start 192.168.42.100 stop 192.168.42.199`
Specify the default router for DHCP1.	`vyatta@R2# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 default-router 192.168.42.254`
Specify a DNS server for DHCP1.	`vyatta@R2# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 dns-server 192.168.42.253`
Specify the local IP address for the DHCP server for failover.	`vyatta@R2# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover local-address 192.168.42.2`
Specify the IP address of the peer DHCP server for failover.	`vyatta@R2# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover peer-address 192.168.42.1`

Step	Command
Specify the role that the DHCP server plays in the failover group.	`vyatta@R2# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover status secondary`
Specify the name of the failover group.	`vyatta@R2# set service dhcp-server shared-network-name DHCP1 subnet 192.168.42.0/24 failover name FAILOVER`
Commit the changes.	`vyatta@R2# commit`

Show the configuration.	`vyatta@R2# show service dhcp-server shared-network-name DHCP1`
	` shared-network-name DHCP1 {`
	`  subnet 192.168.42.0/24 {`
	`   default-router 192.168.42.254`
	`   dns-server 192.168.42.253`
	`   failover {`
	`    local-address 192.168.42.2`
	`    name FAILOVER`
	`    peer-address 192.168.42.1`
	`    status secondary`
	`   }`
	`   start 192.168.42.100 {`
	`    stop 192.168.42.199`
	`   }`
	`  }`
	` }`

Setting up DHCP relay

Configure DHCP relay if you want the AT&T Vyatta vRouter to forward DHCP requests to another DHCP server.

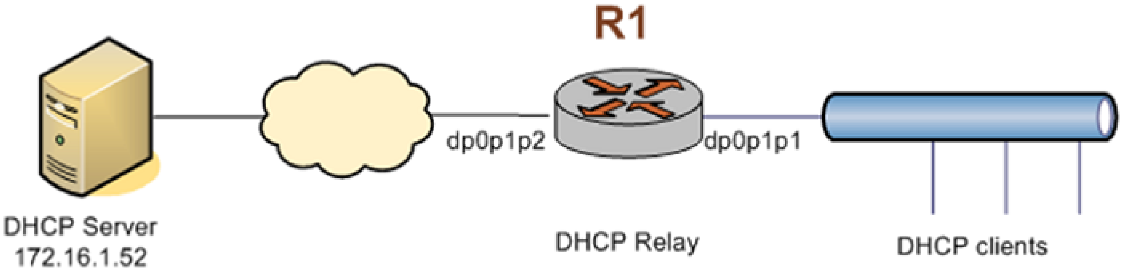
Every interface involved in the DHCP relay must be configured and must be capable of broadcasting. So, for example, if requests are coming in on the dp0p1p1 interface and the DHCP server specified in the configuration is reached through the dp0p1p2 interface, both dp0p1p1 and dp0p1p2 must be configured for DHCP.

The figure below shows how to accomplish the following tasks:

- Configures both dp0p1p1 and dp0p1p2 for DHCP. The router is expected to receive client requests for the DHCP server through the dp0p1p1 interface. It forwards client-to-server DHCP messages to the DHCP server at 172.16.1.52 out through the dp0p1p2 interface.
- Enables relay options. This directs the system to add the Relay Agent Information option (option 82) to the DHCP message before forwarding, as specified by RFC 3046.
- Does not permit reformatting of DHCP messages by this system. If a packet is received that already contains relay information, the packet is discarded.
- Leaves other relay option parameters at default values. This means that the router uses port 67 for DHCP messaging, allows a maximum DHCP packet size of at most 576 bytes, and has a maximum hop count of 10 hops.

Figure 1 *(page 31)* shows the sample DHCP relay configuration.

Figure 4: DHCP relay configuration



To configure DHCP relay, perform the following steps in configuration mode.

Table 7: Setting up DHCP relay

Step	Command
Enable DHCP relay on the dp0p1p1 interface on which client requests are received.	`vyatta@R1# set service dhcp-relay listen-interface dp0p1p1`
Enable DHCP relay on the dp0p1p2 interface on which client messages are forwarded to the DHCP server.	`vyatta@R1# set service dhcp-relay upstream-interface dp0p1p2`
Specify the IP address of the DHCP server.	`vyatta@R1# set service dhcp-relay server 172.16.1.52`
Set the router to discard messages containing relay information. Leave other parameters at default values.	`vyatta@R1# set service dhcp-relay relay-options relay-agents-packets discard`
Commit the changes.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show service dhcp-relay`
	` listen-interface dp0p1p1`
	` upstream-interface dp0p1p2`
	` server 172.16.1.52`
	` relay-options {`
	`  relay-agents-packets discard`
	` }`

Setting additional DHCP configuration parameters

Caution:

This feature is advanced and should be used only by expert users in special situations.

The AT&T Vyatta vRouter DHCP server commands provide a set of commonly used DHCP server features. However, many additional features are available. Information regarding the available DHCP server features are located on the dhcpd.conf man page. To access it, type the following at the Vyatta command prompt:

```
man dhcpd.conf
```

To access these additional features, use one of the following commands, depending on the required scope of the feature. The commands are listed from widest to narrowest scope.

- service dhcp-server global-parameters params *(page 41)*
- service dhcp-server static-mapping mapname static-mapping-parameters params *(page 71)*
- service dhcp-server shared-network-name name shared-network-parameters params *(page 45)*
- service dhcp-server shared-network-name name subnet ipv4net subnet-parameters params *(page 63)*

The precedence of scope of these commands is from narrowest to widest. That is, if more than one command is specified and a given host address falls within the scope of both, it is governed by parameters specified in the command with the narrowest scope.

Multiple parameter strings can be specified. Each parameter string that is specified adds a separate line to the `dhcpd.conf` file.

Note that no validation is done by the AT&T Vyatta vRouter before passing the parameter string from these commands to the DHCP server process (dhcpd). Because of this lack of validation, it is imperative that the syntax described in the `dhcpd.conf` documentation be strictly followed. Failure to do so could result in a failure of the DHCP server. It is advisable to check the system log for errors when using these parameter strings. In addition, the `show system processes` command can be used to determine if the dhcpd process is still running.

The following example shows how the additional DHCP server parameters can be accessed. To configure additional DHCP server parameters, perform the following steps in configuration mode.

Table 8: Setting up a DHCP server with additional parameters

Step	Command
Enable the DHCP server and define an option that does not already have a keyword defined in the dhcpd process. See the dhcpd man page for further information.	`vyatta@R1# set service dhcp-server global-parameters ‘option rfc3442-static-route code 121 = string;’`
Specify the value to be used for the option for all shared networks, subnets, and static mappings defined in the DHCP server configuration.	`vyatta@R1# set service dhcp-server global-parameters ‘option rfc3442-static-route 01:01:01:01:01:01:01:01;’`
Specify an IP address to statically map to a host with a specific MAC address.	`vyatta@R1# set service dhcp-server static-mapping MAP1 ip-address 172.16.117.15`
Specify the MAC address of a host to be statically mapped to an IP address.	`vyatta@R1# set service dhcp-server static-mapping MAP1 mac-address 09:09:09:09:09:09`
Override the global value of the parameter defined previously for a specific host.	`vyatta@R1# set service dhcp-server static-mapping MAP1 static-mapping-parameters ‘option rfc3442-static-route 01:01:01:01:01:01:01:02’`

Step	Command
Specify that the DHCP server is authoritative for the specified shared network.	vyatta@R1# set service dhcp-server shared-network-name NET1 authoritative enable
Specify the subnet and address pool to use.	vyatta@R1# set service dhcp-server shared-network-name NET1 subnet 172.16.117.0/24 start 172.16.117.10 stop 172.16.117.20
Commit the changes.	vyatta@R1# commit

Show the configuration.	vyatta@R1# show service dhcp-server
	global-parameters “option rfc3442-static-route code 121 = string;”
	global-parameters “option rfc3442-static-route 01:01:01:01:01:01:01:01;”
	shared-network-name NET1 {
	authoritative enable
	subnet 172.16.117.0/24 {
	start 172.16.117.10 {
	stop 172.16.117.20
	}
	}
	}
	static-mapping MAP1 {
	ip-address 172.16.117.15
	mac-address 09:09:09:09:09:09
	static-mapping-parameters “option-rfc3422-static-route 01:01:01:01:01:01:02”
	}

DHCP Commands

interfaces bridge <brx> dhcp-options no-rfc3442

Disables support for the DHCP classless static route option for a bridge group.

Syntax:
`set interfaces bridge` *brx* **`dhcp-options no-rfc3442`**

Syntax:
`delete interfaces bridge` *brx* **`dhcp-options no-rfc3442`**

Syntax:
`show interfaces bridge` *brx* **`dhcp-options`**

The classless static route option for DHCP is enabled.

brx
The identifier for a bridge group. The identifier ranges from br0 through br999.

`no-rfc3442`
Removes the classless static route option (121) from the parameter request list that a DHCP client sends to the DHCP server. For further information, refer to RFC 3442 at https://tools.ietf.org/html/rfc3442.

Configuration mode

```
interfaces {
    bridge brx {
        dhcp-options {
            no-rfc3442
        }
    }
}
```

Note: This command is relevant only if the **`dhcp`** option has been set by using the `interfaces bridge` *brx* **`address`** *address* command.

Note: Normally, this command is not required. It would be used only if the remote DHCP server is configured to provide classless static routes, but these routes are not required on the router that is configured to use the DHCP address.

Use the `set` form of this command to disable DHCP classless static route option support for a bridge group.

Use the `delete` form of this command to re-enable DHCP classless static route option support for a bridge group.

Use the `show` form of this command to display the status of the DHCP classless static route option for a bridge group.

release dhcp interface <interface>

Releases the current DHCP client lease from an interface.

Syntax:
`release dhcp interface` *interface*

interface

An interface that uses DHCP to obtain an IP address.

Operational mode

Use this command to release the current DHCP client lease from an interface. The interface must be a DHCP client that obtained an IP address from a DHCP server.

renew dhcp interface <interface>

Renews the current DHCP client lease on an interface.

Syntax:

`renew dhcp interface` *interface*

interface

An interface that uses DHCP to obtain an IP address.

Operational mode

Use this command to renew the current DHCP client lease on an interface. The interface must be a DHCP client that obtained an IP address from a DHCP server.

reset dhcp server lease ip <address>

Removes the DHCP lease for an IP address.

Syntax:

`reset dhcp server lease ip` [*ipv4-address* | *ipv6-address*]

ipv4

An IPv4 address.

ipv6

An IPv6 address.

Operational mode

Use this command to remove the DHCP lease for an IP address. The command applies to leases provided by the DHCP server. The server is configured by using service dhcp-server *(page 39)*.

reset dhcp server leases

Removes all DHCP leases.

Syntax:

```
reset dhcp server leases
```

Operational mode

```
service dhcp-server
```

Use this command to remove all DHCP leases. The command applies to leases provided by the DHCP server. The server is configured by using service dhcp-server *(page 39)*.

restart dhcp relay-agent

Restarts the DHCP relay agent.

Syntax:

```
restart dhcp relay-agent
```

Operational mode

Use this command to stop the DHCP relay agent if it is running, then start it if it is configured. This command can be used if the DHCP relay agent is not operating properly.

restart dhcp server

Restarts the DHCP server.

Syntax:

```
restart dhcp server
```

Operational mode

Use this command to stop and restart the DHCP server. This command can be used if the DHCP relay agent is not operating properly.

service dhcp-relay

Configures the system to relay DHCP client messages to an off-network DHCP server.

Syntax:

```
set service dhcp-relay
```

Syntax:

```
delete service dhcp-relay
```

Syntax:

```
show service dhcp-relay
```

Configuration mode

```
service {
 dhcp-relay {
 }
}
```

Use this command to configure the system as a DHCP relay agent.

A DHCP relay agent receives DHCP packets from DHCP clients and forwards them to a DHCP server. This allows you to place DHCP clients and DHCP servers on different networks; that is, across router interfaces.

The relay agent is configured with addresses of DHCP servers to which they should relay client DHCP messages. The relay agent intercepts the broadcast, sets the gateway address (the **giaddr** field of the DHCP packet) and, if configured, inserts the Relay Agent Information option (option 82) in the packet and forwards it to the DHCP server.

The DHCP server echoes the option back verbatim to the relay agent in server-to-client replies, and the relay agent strips the option before forwarding the reply to the client.

All interfaces involved in the DHCP relay for both clients and servers must be explicitly defined by using service dhcp-relay listen-interface <interface> *(page 36)*.

Use the `set` form of this command to define DHCP relay configuration.

Use the `delete` form of this command to remove DHCP relay configuration.

Use the `show` form of this command to view DHCP relay configuration.

service dhcp-relay listen-interface <interface>

Enables DHCP relay on an interface for receiving DHCP requests from DHCP clients.

Syntax:

`set service dhcp-relay listen-interface` *dp-interface*

Syntax:

`delete service dhcp-relay listen-interface` *dp-interface*

Syntax:
`show service [ `**`dhcp-relay listen-interface`**` ] ]`

interface

A data plane interface on the router. At least one interface must be specified.

You can assign multiple interfaces to be used for DHCP by creating multiple **`listen-interface`** configuration nodes.

Configuration mode

```
service {
 dhcp-relay {
  listen-interface interface
 }
}
```

Use this command to enable DHCP-relay on an interface to receive DHCP requests from DHCP clients.

At least one DHCP-relay server must be configured.

Use the `set` form of this command to specify the interface.

Use the `delete` form of this command to remove the interface.

Use the `show` form of this command to view the interface.

service dhcp-relay relay-options

Specifies whether to add the Relay Agent Information option (option 82) to the client-to-server packet.

Syntax:
`set service dhcp-relay relay-options [ `**`hop-count`** *count* | **`max-size`** *size* | **`port`** *port* | **`relay-agents-packets`** *policy*]

Syntax:
`delete service dhcp-relay relay-options [ `**`hop-count`** | **`max-size`** | **`port`** | **`relay-agents-packets`**]

Syntax:
`show service dhcp-relay relay-options [ `**`hop-count`** | **`max-size`** | **`port`** | **`relay-agents-packets`**]

`hop-count` ***count***

Optional. Sets the hop count for outgoing relayed messages. After the hop count is reached, the packet is discarded. The hop count should be set high enough that relayed packets are able to reach the DHCP server. The count ranges from 0 through 255. The default count is 10.

`max-size` ***size***

Optional. Sets the maximum size of the DHCP packet to be created after appending the relay agent information option. If, after appending the information, the packet exceeds this size, the packet is forwarded without appending the information. This size should be set to the lowest MTU size in your network. The size ranges from 64 through1400. The default size is 576.

If this option is not configured, the router does not forward DHCP packets that exceed the MTU of the interface on which relaying is configured.

`port` ***port***

Optional. Specifies the port on this interface that relays DHCP client messages. This should be done only for debugging because the behavior changes; responses are broadcast rather than being sent to port 68 of the requesting client. The port ranges from 1 through 65535.

`relay-agents-packet` ***policy***

Optional. Sets the reforwarding policy for a DHCP relay agent. The router takes this action if the DHCP message already contains relay information. The policy is one of the following:

append: The DHCP relay agent may append its own set of relay options to the packet, leaving the supplied option field intact.

discard: If the packet already contains relay information, it is discarded.

forward: The packet is forwarded regardless of whether it contains relay information.

replace: The DHCP relay agent may replace the supplied option field with its own set of relay options.

The default policy is **forward**.

Configuration mode

```
service {
 dhcp-relay {
  relay-options {
   hop-count count
   max-size size
   port port
   relay-agents-packets policy
  }
 }
}
```

Use this command to configure the Relay Agent Information option (option 82) in the client-to-server packet, as specified by RFC 3046, and configure DHCP relay options.

Setting the port to a value other than 67 should be done only for debugging. When this is done, DHCP requests from clients are still accepted on port 67, but the responses from DHCP servers are forwarded to broadcast address 255.255.255.255 port 0 rather than on port 68, where DHCP clients listen.

Use the `set` form of this command to set DHCP relay options.

Use the `delete` form of this command to restore default DHCP relay options.

Use the `show` form of this command to view DHCP relay option configuration.

service dhcp-relay server <ipv4>

Specifies the IP address of a DHCP server.

Syntax:
`set service dhcp-relay server` *ipv4*

Syntax:
`delete service dhcp-relay server` *ipv4*

Syntax:
`show service dhcp-relay server`

ipv4

Mandatory. Multinode. The IP address of a DHCP server.

You can relay messages to more than one DHCP server by creating multiple **server** configuration nodes.

Configuration mode

```
service {
 dhcp-relay {
  server ipv4 {
  }
 }
}
```

Use this command to specify the IP address of a DHCP server.

Use the `set` form of this command to specify the IP address of a DHCP server in a DHCP relay configuration.

Use the `delete` form of this command to remove DHCP server configuration in a DHCP relay configuration.

Use the `show` form of this command to view DHCP server configuration in a DHCP relay configuration.

service dhcp-relay upstream-interface <interface>

Specifies an interface for forwarding DHCP requests to the DCHP server.

Syntax:
`set service dhcp-relay upstream-interface` *dp-interface*

Syntax:
`delete service dhcp-relay upstream-interface` *interface*

Syntax:
`show service [` **`dhcp-relay`** **`upstream-interface`** `] ]`

interface

A data plane interface to forward DHCP requests. At least one interface must be specified.

You can assign multiple interfaces to be used for DHCP forwarding by creating multiple **`upstream-interface`** configuration nodes.

Configuration mode

```
service {
 dhcp-relay {
  upstream-interface interface
 }
}
```

Use this command to specify an interface for forwarding DHCP requests to the DHCP server.

At least one DHCP-relay server must be configured.

Use the `set` form of this command to specify an interface for forwarding DHCP requests to the DHCP server.

Use the `delete` form of this command to remove the interface.

Use the `show` form of this command to view the interface.

service dhcp-server

Enables DHCP server functionality.

Syntax:
`set service dhcp-server`

Syntax:
`delete service dhcp-server`

Syntax:
`show service dhcp-server`

Configuration mode

```
service {
 dhcp-server {
 }
}
```

Use this command to configure a pool of addresses the system can use for DHCP.

At least one address pool must be configured for DHCP to be available as a service.

At least one address pool must lie within a configured subnet on any of the broadcast interfaces.

Each subnet that is specified contains a distinct address pool. A given interface can support more than one address pool (that is, more than one subnet).

Use the `set` form of this command to enable DHCP server functionality.

Use the `delete` form of this command to remove DHCP server functionality.

Use the `show` form of this command to view DHCP server configuration.

service dhcp-server disabled <state>

Disables the DHCP server without discarding configuration.

Syntax:
`set service dhcp-server disabled` *state*

Syntax:
`delete service dhcp-server disabled`

Syntax:
`show service dhcp-server disabled`

DHCP server functionality is disabled.

state

The administrative state of the DHCP server. The state is either of the following:

`true`: Disables the DHCP server without discarding configuration.

`false`: Enables the DHCP server.

Configuration mode

```

service {
 dhcp-server {
  disabled state
 }
}
```

Use this command to disable the DHCP server without discarding configuration.

Use the `set` form of this command to specify whether the DHCP server should be disabled.

Use the `delete` form of this command to restore the default state, that is, DHCP server functionality is disabled.

Use the `show` form of this command to view DHCP server configuration.

service dhcp-server dynamic-dns-update enable <state>

Specifies whether to dynamically update DNS.

Syntax:
`set service dhcp-server dynamic-dns-update enable` *state*

Syntax:
`delete service dhcp-server dynamic-dns-update enable`

Syntax:

```
show service dhcp-server dynamic-dns-update enable
```

DNS updates are not sent by the DHCP server.

state

The state of dynamic DNS updates. The state is either of the following:

`true`: Sends updates dynamically.

`false`: Does not send updates.

Configuration mode

```
service {
 dhcp-server {
  dynamic-dns-update {
   enable state
  }
 }
}
```

Use this command to control DNS updates from the DHCP server.

Use the `set` form of this command to specify whether dynamic DNS updates should be sent.

Use the `delete` form of this command to restore the default state, that is, DNS updates are not sent.

Use the `show` form of this command to view the dynamic DNS update configuration.

service dhcp-server global-parameters <params>

Specifies additional global DHCP server parameters.

Syntax:
`set service dhcp-server global-parameters` *params*

Syntax:
`delete service dhcp-server global-parameters` *params*

Syntax:
`show service dhcp-server global-parameters`

params

A string of parameters to be used by the DHCP server. The string must be enclosed in single quotation marks (‘).

Configuration mode

```
service {
 dhcp-server {
  global-parameters params
 }
}
```

Danger:

This feature is advanced and should be used by only expert users in special situations.

Use this command to specify additional global DHCP server parameters that are not available with the `service dhcp-server` commands. The AT&T Vyatta vRouter DHCP server commands are a subset of those that are available for DHCP server configuration. This command provides access to all DHCP server configuration parameters. More information regarding DHCP server configuration is located on the `dhcpd.conf` man page. To access the page, type the following at the Vyatta command prompt:

`man dhcpd.conf`

The AT&T Vyatta vRouter does no validation before passing the parameter string to the DHCP server process (dhcpd). Because of this nonvalidation, it is imperative that the syntax described in the `dhcpd.conf` documentation be strictly followed. Failure to do so could result in a failure of the DHCP server. It is advisable to check the system log for errors when using these parameter strings. In addition, the `show system processes` command determines if the dhcpd process is still running.

The scope of these parameters is global. They apply to all shared-networks, subnets, and static-mappings unless parameters with a narrower scope are specified by using the **`shared-network-parameters`**, **`subnet-parameters`**, or **`static-mapping-parameters`** version of this command.

Multiple parameter strings can be specified. Each parameter string that is specified adds a separate line to the `dhcpd.conf` file.

Use the `set` form of this command to specify additional global DHCP server parameters.

Use the `delete` form of this command to remove additional global DHCP server parameters.

Use the `show` form of this command to display additional global DHCP server parameters.

service dhcp-server listento interface <dp-interface>

Allows the DHCP server to create address pools for clients that are indirectly connected to a data plane network interface through a DHCP relay server.

Syntax:
`set service dhcp-server listento interface` *dp-interface*

Syntax:
`delete service dhcp-server listento interface`

Syntax:
`show service dhcp-server`

dp-interface
A data plane interface on the router. It must have a valid IP address.

Configuration mode.

```
service {
   dhcp-server {
      listento {
         interface dp-interface
      }
   }
}
```

Use this command to enable the DHCP server to create IP address pools for clients that are not directly connected to the router. For example, if clients on B subnet connect to the router through a DHCP relay server, the DHCP relay server connects to the router through a data plane interface on the A subnet, and the data plane interface has a valid IP address, using this command allows the DHCP server to create IP address pools for clients that are on the B subnet.

Use the `set` form of this command to create an IP address pool for clients that are indirectly connected to the router through a data plane network interface.

Use the `delete` form of this command to remove a data plane interface from the DHCP server configuration. If no data plane interfaces are configured, the DHCP server cannot create address pools.

Use the `show` form of this command to view the DHCP server configuration.

service dhcp-server shared-network-name <name>

Specifies the name for a DHCP address pool.

Syntax:
`set service dhcp-server shared-network-name` *name*

Syntax:
`delete service dhcp-server shared-network-name` *name*

Syntax:
`show service dhcp-server shared-network-name` *name*

name

Mandatory. Multinode. The name for a DHCP address pool.

You can define multiple address pools by creating multiple **`shared-network-name`** configuration nodes, each with a different name.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
  }
 }
}
```

Use this command to create a DHCP server address pool with the specified name.

Use the `set` form of this command to create a DHCP address pool.

Use the `delete` form of this command to remove a DHCP address pool.

Use the `show` form of this command to display a DHCP address pool.

service dhcp-server shared-network-name <name> authoritative <state>

Specifies whether the DHCP server is the authoritative server.

Syntax:
`set service dhcp-server shared-network-name` *name* **`authoritative`** *state*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`authoritative`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`authoritative`**

The DHCP server is not authoritative.

name

Mandatory. A DHCP address pool.

state

Specifies whether the DHCP server is the authoritative server. The state is either of the following:

`enable` Enables authoritative state.

`disable` Disables authoritative state.

The default state is **disable**.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   authoritative state
  }
 }
}
```

Use this command to set the DHCP server as the authoritative server.

Setting the server as authoritative sets the server as a master server and allows it to protect itself from rogue DHCP servers or misconfigured DHCP clients. If the server is authoritative, it sends a DHCPNAK to a misconfigured client; otherwise, the client cannot update its IP address until after the old lease expires.

Use the `set` form of this command to enable or disable the authoritative state for the DHCP server.

Use the `delete` form of this command to restore the default authoritative state, which is not authoritative.

Use the `show` form of this command to display whether the DHCP server is authoritative.

service dhcp-server shared-network-name <name> description <desc>

Provides a description of a shared network.

Syntax:
`set service dhcp-server shared-network-name` *name* **`description`** *desc*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`description`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`description`**

name
Mandatory. A DHCP address pool.

desc
A description of a shared network.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   description desc
  }
 }
}
```

Use this command to provide a description of a shared network.

Use the `set` form of this command to provide a description of a shared network.

Use the `delete` form of this command to delete the description of a shared network.

Use the `show` form of this command to display the description of a shared network.

service dhcp-server shared-network-name <name> disable

Disables DHCP configuration for a shared network.

Syntax:
`set service dhcp-server shared-network-name` *name* **`disable`**

Syntax:
`delete service dhcp-server shared-network-name` *name* **`disable`**

Syntax:
`show service dhcp-server shared-network-name` *name*

A shared-network configuration is enabled.

name
Mandatory. A DHCP address pool.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   disable
  }
 }
}
```

Use this command to disable DHCP configuration of a shared network.

Use the `set` form of this command to disable DHCP configuration of a shared network.

Use the `delete` form of this command to enable DHCP configuration of a shared network.

Use the `show` form of this command to display DHCP configuration of a shared network.

service dhcp-server shared-network-name <name> shared-network-parameters <params>

Specifies additional shared-network DHCP server parameters.

Syntax:
`set service dhcp-server shared-network-name` *name* **`shared-network-parameters`** *params*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`shared-network-parameters`** *params*

Syntax:
`show service dhcp-server shared-network-name` *name* **`shared-network-parameters`**

name
Mandatory. A DHCP address pool.

params
A string of parameters to be used by the DHCP server. The string must be enclosed in single quotation marks (').

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   shared-network-parameters params
  }
 }
}
```

Note:

This feature is advanced and should be used by only expert users in special situations.

Use this command to specify additional shared-network DHCP server parameters that are not available with the `service dhcp-server` commands. The AT&T Vyatta vRouter DHCP server commands are a subset of those that are available for DHCP server configuration. This command provides access to all DHCP server configuration parameters. More information regarding DHCP server configuration is located on the dhcpd.conf man page. To access the page, type the following at the Vyatta command prompt:

```
man dhcpd.conf
```

The AT&T Vyatta vRouter does no validation before passing the parameter string to the DHCP server process (dhcpd). Because of this nonvalidation, it is imperative that the syntax described in the `dhcpd.conf` documentation be strictly followed. Failure to do so could result in a failure of the DHCP server. It is advisable to check the system log for errors when using these parameter strings. In addition, the `show system processes` command determines if the dhcpd process is still running.

The scope of these parameters is for the specified shared network. They apply to all subnets, and static-mappings within this scope unless parameters with a narrower scope are specified by using the **`subnet-parameters`** or **`static-mapping-parameters`** version of this command.

Multiple parameter strings can be specified. Each parameter string that is specified adds a separate line to the `dhcpd.conf` file.

Use the `set` form of this command to specify additional shared-network DHCP server parameters.

Use the `delete` form of this command to remove additional shared-network DHCP server parameters.

Use the `show` form of this command to display additional shared-network DHCP server parameters.

service dhcp-server shared-network-name <name> subnet <ipv4net>

Specifies the IPv4 network to be served by a DHCP address pool.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net*

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network to be served with the addresses defined in the specified address pool. The format of the network designation is *ip-addr/prefix*.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
   }
  }
 }
}
```

Use this command to specify the IPv4 network to be served with the addresses that are defined in this named rule. DHCP requests from devices on this subnet are served static address assignments or an address from the defined range.

Use the `set` form of this command to specify the DHCP address pool subnet.

Use the `delete` form of this command to remove DHCP address pool subnet configuration.

Use the `show` form of this command to view DHCP address pool subnet configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> bootfile-name <bootfile>

Specifies a bootstrap file from which diskless PCs can boot.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`bootfile-name`** *bootfile*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`bootfile-name`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`bootfile-name`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

bootfile
The name of a bootstrap file to be used to boot.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    bootfile-name bootfile
   }
  }
 }
}
```

Use this command to specify a bootstrap file from which diskless PCs can boot.

Use the `set` form of this command to specify a bootstrap file.

Use the `delete` form of this command to remove boot file configuration.

Use the `show` form of this command to view boot file configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> bootfile-server <addr>

Specifies a bootstrap server from which diskless PCs can boot.

Syntax:

```
set service dhcp-server shared-network-name name subnet ipv4net bootfile-server addr
```

Syntax:

```
delete service dhcp-server shared-network-name name subnet ipv4net bootfile-server
```

Syntax:

```
show service dhcp-server shared-network-name name subnet ipv4net bootfile-server
```

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

addr

The IPv4 address or host name of the bootfile server.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    bootfile-server addr
   }
  }
 }
}
```

Use this command to specify a bootstrap server from which diskless PCs can boot.

Use the `set` form of this command to specify a bootstrap server.

Use the `delete` form of this command to remove boot server configuration.

Use the `show` form of this command to view boot server configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> client-prefix-length <prefix>

Specifies the length of a subnet prefix to be assigned to clients.

Syntax:

```
set service dhcp-server shared-network-name name subnet ipv4net client-prefix-length prefix
```

Syntax:

```
delete service dhcp-server shared-network-name name subnet ipv4net client-prefix-length
```

Syntax:

```
show service dhcp-server shared-network-name name subnet ipv4net client-prefix-length
```

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

prefix

Optional. The length of a subnet prefix that is assigned to each client. By default, the prefix length defined in the `subnet` parameter is assigned. The prefix length ranges from 0 through 32.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    client-prefix-length prefix
   }
  }
 }
}
```

Use this command to specify the length of a subnet prefix that is assigned to each client.

Use the `set` form of this command to specify the length of a prefix subnet that is assigned to each client.

Use the `delete` form of this command to delete the length of a subnet prefix.

Use the `show` form of this command to display the length of a subnet prefix.

service dhcp-server shared-network-name <name> subnet <ipv4net> ping-check

Pings the IP address of the shared network subnet to confirm if the address is not configured on another node. Ping-check is valid only for IPv4.

Syntax:
`set service dhcp-server shared-network-name`*name* **`subnet`** *ipv4net* **`ping-check`**

Syntax:
`delete service dhcp-server shared-network-name`*name* **`subnet`** *ipv4net* **`ping-check`**

Syntax:
`show service dhcp-server shared-network-name`*name* **`subnet`** *ipv4net* **`ping-check`**

name

Mandatory. A DHCP address pool.

You can define multiple address pools by creating multiple **`shared-network-name`** configuration nodes, each with a different name.

ipv4net

Mandatory. Multinode. The IPv4 network to be served with the addresses defined in the specified address pool. The format of the network designation is *ip-addr/prefix*.

Configuration mode

```
service {
   dhcp-server {
       shared-network-name name {
          subnet ipv4net {
             ping-check {

             }
          }
       }
   }
}
```

Use this command to ping the IP address of the shared network subnet to confirm if the address is not configured on another node. Ping-check is valid only for IPv4.

Use the `set` form of this command to specify the IPv4 subnet for ping-check.

Use the `delete` form of this command to remove the IPv4 subnet for ping-check.

Use the `show` form of this command to view the IPv4 subnet for ping-check.

The following example shows how to use the ping-check option for a shared network called `foo` and the IPv4 subnet address of `12.1.1.0/24`. This option generates an ICMP echo request before offering an address to the client.

```
vyatta@vyatta# set service dhcp-server shared-network-name foo subnet 12.1.1.0/24 ping-check
```

service dhcp-server shared-network-name <name> subnet <ipv4net> default-router <ipv4>

Specifies the address of the default router for DHCP clients on a subnet.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`default-router`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`default-router`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`default-router`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4
Optional. The address of the default router for DHCP clients on this subnet. The default router should be on the same subnet as the client. The format is an IP address.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    default-router ipv4
   }
  }
 }
}
```

Use this command to specify the address of the default router for DHCP clients on a subnet.

Use the `set` form of this command to specify the address of the default router for DHCP clients on a subnet.

Use the `delete` form of this command to remove the default-router configuration.

Use the `show` form of this command to view the default-router configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> dns-server <ipv4>

Specifies the address of a DNS server for DHCP clients.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`dns-server`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`dns-server`** *ipv4*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`dns-server`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

Optional. Multinode. The IPv4 address of a DNS server.

You can specify more than one DNS server by entering this command multiple times.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    dns-server ipv4
   }
  }
 }
}
```

Use this command to specify the address of a DNS server that is available to DHCP clients.

Use the `set` form of this command to specify the address of a DNS server.

Use the `delete` form of this command to remove DNS server configuration.

Use the `show` form of this command to view DNS server configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> domain-name <domain-name>

Provides the domain name for DHCP clients on a subnet.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`domain-name`** *domain-name*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`domain-name`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`domain-name`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

domain-name

Optional. The domain name to be given to DHCP clients on this subnet. A domain name can include letters, numbers, hyphens (-), and one period (.). For example, att.com.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    domain-name domain-name
   }
  }
 }
}
```

Use this command to specify the domain name for DHCP clients on a subnet.

Use the `set` form of this command to specify the client domain name.

Use the `delete` form of this command to remove client domain name configuration.

Use the `show` form of this command to view client domain name configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> exclude <ipv4>

Excludes an IP address from a DHCP address pool.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`exclude`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`exclude`** *ipv4*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`exclude`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

Optional. Multinode. An IP address to exclude from the lease range.

You can exclude more than one IP address by creating multiple **`exclude`** configuration nodes.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    exclude ipv4
   }
  }
 }
}
```

Use this command to exclude an IP address from a DHCP address pool. An excluded address is never leased to DHCP clients. The exception is an IP address that is statically mapped by using service dhcp-server static-mapping mapname *(page 68)*. This address is not excluded.

Use the `set` form of this command to exclude an IP address from the lease range.

Use the `delete` form of this command to delete an IP address from the list of excluded addresses.

Use the `show` form of this command to display excluded IP addresses.

service dhcp-server shared-network-name <name> subnet <ipv4net> failover

Enables DHCP failover functionality for a DHCP address pool on a subnet.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover`**

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    failover {
    }
   }
  }
 }
}
```

Use this command to enable DHCP failover for an address pool on a given network, allowing two DHCP servers to share an address pool.

In a failover configuration, two DHCP servers act as failover peers, with one of the peers designated as the primary and the other as the secondary. For DHCP failover to work, the following conditions must be met.

- Both peers must be AT&T Vyatta vRouters and must be running the same version of AT&T Vyatta vRouter software.
- Each server must be configured to point to the other as the failover peer.
- The time on the servers must be exactly synchronized.
- At least one IP address must exist in the start-stop range for each subnet that has not been either excluded (by using service dhcp-server shared-network-name name subnet ipv4net exclude ipv4 *(page 52)*) or statically mapped (by using service dhcp-server static-mapping mapname *(page 68)*).

The system times should be synchronized before configuring DHCP failover. Use of NTP time synchronization is highly recommended. However, if difficulties arise because of incorrect system times, then disable NTP, reset the times correctly, and re-enable NTP.

Note that DHCP leases are assigned only in failover configurations if proper communication is established between the two failover peers. If the configuration is incorrect (if, for example, one failover peer is configured but the other is not), DHCP leases are not dispersed.

Note also that statically mapped addresses are not renewed by a failover server unless they are explicitly defined on that server by using service dhcp-server static-mapping mapname *(page 68)*.

Use the `set` form of this command to define DHCP failover configuration.

Use the `delete` form of this command to remove DHCP failover configuration.

Use the `show` form of this command to view DHCP failover configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> failover local-address <ipv4>

Specifies the IP address of the local failover peer.

Syntax:

```
set service dhcp-server shared-network-name name subnet ipv4net failover local-address ipv4
```

Syntax:

```
delete service dhcp-server shared-network-name name subnet ipv4net failover local-address
```

Syntax:

```
show service dhcp-server shared-network-name name subnet ipv4net failover local-address
```

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

The IP address of the local failover peer.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    failover {
     local-address ipv4
    }
   }
  }
 }
}
```

Use this command to specify the IP address of the local failover peer.

Use the `set` form of this command to specify the IP address of the local failover peer.

Use the `delete` form of this command to remove local failover IP address configuration.

Use the `show` form of this command to view local failover IP address configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> failover name <peer-name>

Specifies the DHCP failover IP address for the local peer.

Syntax:

```
set service dhcp-server shared-network-name name subnet ipv4net failover name peer-name
```

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover name`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover name`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

peer-name
The DHCP failover peer name for the local peer.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    failover {
     name peer-name
    }
   }
  }
 }
}
```

Use this command to specify a name for the local peer in a DHCP failover pair.

Use the `set` form of this command to specify the DHCP failover peer name.

Use the `delete` form of this command to remove the local peer name configuration.

Use the `show` form of this command to view local peer name configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> failover peer-address <ipv4>

Specifies the IP address of the local failover peer.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover peer-address`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover peer-address`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover peer-address`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4
The IP address of the local failover peer.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    failover {
     peer-address ipv4
    }
   }
  }
 }
}
```

Use this command to specify the IP address of the local failover peer.

Use the `set` form of this command to specify the IP address of the local failover peer.

Use the `delete` form of this command to remove the IP address configuration.

Use the `show` form of this command to view the IP address configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> failover status <status>

Specifies the DHCP failover status of the local system.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover`** **`status`** *status*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover`** **`status`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`failover`** **`status`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

status

The DHCP failover status of a peer in the failover configuration. The status is either of the following:

- **`primary`**—Indicates the local system is the primary peer.
- **`secondary`**—Indicates the local system is the secondary peer.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    failover {
     status status
    }
   }
  }
 }
}
```

Use this command to specify the DHCP failover status of the local system.

Use the `set` form of this command to specify the DHCP failover status as primary or secondary.

Use the `delete` form of this command to remove failover status configuration.

Use the `show` form of this command to view failover status configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> ip-forwarding enable <state>

Specifies whether a client configures its IP layer for packet forwarding.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`ip-forwarding enable`** *state*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`ip-forwarding enable`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`ip-forwarding enable`**

The DHCP server does not direct a client to configure its IP layer for packet forwarding.

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

state

Whether a client configures its IP layer for packet forwarding. The state is either of the following:

- **`true`** —Indicates that a client does configure its IP layer for packet forwarding.
- **`false`** —Indicates that a client does not configure its IP layer for packet forwarding.

The default state is **`false`**.

Configuration mode

```
service {
   dhcp-server {
      shared-network-name name {
         subnet ipv4net {
            ip-forwarding {
               enable state
            }
         }
      }
   }
}
```

Use this command to specify whether the DHCP server directs a client to configure its IP layer for packet forwarding.

Use the `set` form of this command to specify whether a client configures its IP layer for packet forwarding.

Use the `delete` form of this command to restore the default configuration.

Use the `show` form of this command to view IP forwarding configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> lease <seconds>

Specifies how long the address assigned by the DHCP server is valid.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`lease`** *seconds*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`lease`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`lease`**

The default number of seconds is 86,400 (24 hours).

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

seconds
Optional. The number of seconds the address that is assigned by the DHCP server is valid. The number of seconds ranges from 120 through 4294967296.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    lease seconds
   }
  }
 }
}
```

Use this command to specify how long the address assigned by the DHCP server is valid.

Use the `set` form of this command to specify how long the address assigned by the DHCP server is valid.

Use the `delete` form of this command to remove the lease configuration.

Use the `show` form of this command to view the lease configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> ntp-server <ipv4>

Specifies the address of a Network Time Protocol (NTP) server that is available to clients.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`ntp-server`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`ntp-server`** *ipv4*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`ntp-server`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

Optional. The IP address of an NTP server. Multiple NTP server addresses can be specified in separate commands. The NTP servers should be specified in a preferred order.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    ntp-server ipv4
   }
  }
 }
}
```

Use this command to specify the address of an NTP server that is available to clients.

Use the `set` form of this command to specify the address of an NTP server.

Use the `delete` form of this command to remove the NTP server configuration.

Use the `show` form of this command to view the NTP server configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> pop-server <ipv4>

Specifies the address of a Post Office Protocol 3 (POP3) server that is available to clients.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`pop-server`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`pop-server`** *ipv4*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`pop-server`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

Optional. The IP address of a POP3 server. Multiple POP3 server addresses can be specified in separate commands. The POP3 servers should be specified in a preferred order.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    pop-server ipv4
   }
  }
 }
}
```

Use this command to specify the address of a POP3 server that is available to clients.

Use the `set` form of this command to specify the address of a POP3 server.

Use the `delete` form of this command to remove the POP3 server configuration.

Use the `show` form of this command to view the POP3 server configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> server-identifier <ipv4>

Specifies the address for the DHCP server identifier.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`server-identifier`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`server-identifier`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`server-identifier`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4
Optional. The address for the DHCP server identifier.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    server-identifier ipv4
   }
  }
 }
}
```

Use this command to specify the address for the DHCP server identifier.

The server identifier is a field in a DHCP message that identifies the DHCP server as the destination address from clients to servers. When the DHCP server includes this field in a DHCPOffer, a client uses it to distinguish between multiple lease offers. The server identifier must be an address that can be reached from the client.

Use the `set` form of this command to specify the address for the DHCP server identifier.

Use the `delete` form of this command to remove the address for the DHCP server identifier.

Use the `show` form of this command to view the DHCP server identifier configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> smtp-server <ipv4>

Specifies the address of a Simple Mail Transfer Protocol (SMTP) server that is available to clients.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`smtp-server`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`smtp-server`** *ipv4*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`smtp-server`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

Optional. The IP address of an SMTP server. Multiple SMTP server addresses can be specified in separate commands. The SMTP servers should be specified in a preferred order.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    smtp-server ipv4
   }
  }
 }
}
```

Use this command to specify the address of an SMTP server that is available to clients.

Use the `set` form of this command to specify the address of an SMTP server.

Use the `delete` form of this command to remove the SMTP server configuration.

Use the `show` form of this command to view the SMTP server configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> start <ipv4> stop <ipv4>

Specifies the range of addresses that are assigned to DHCP clients.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`start`** *ipv4* **`stop`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`start`** [*ipv4* [**`stop`**]]

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`start`** [*ipv4*]

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

`start`

Optional. Multinode. The beginning address in a range of addresses. This address is the first address in the range that can be assigned.

You can define multiple address ranges within an address pool by creating multiple **`start`** configuration nodes.

`stop`

Mandatory. The ending address in this range of addresses. This address is the last address in the range that can be assigned.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    start ipv4 {
     stop ipv4
    }
   }
  }
 }
}
```

Use this command to specify the range of addresses that are assigned to DHCP clients.

Use the `set` form of this command to specify the range of addresses that are assigned to DHCP clients.

Use the `delete` form of this command to remove the address range configuration.

Use the `show` form of this command to view the address range configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> static-route destination-subnet <ipv4net>

Specifies the destination subnet of a static route for clients to store in their routing cache.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`static-route destination-subnet`** *ipv4net2*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`static-route destination-subnet`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`static-route destination-subnet`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4net2

The destination IP subnet of a static route for clients to store in their routing table.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    static-route {
     destination-subnet ipv4net2
   }
  }
 }
}
```

Use this command to specify the destination subnet of a static route for clients to store in their routing cache. The other part of the static route is defined by using service dhcp-server shared-network-name name subnet ipv4net static-route router ipv4 *(page 63)*. Only one static route can be defined for a given subnet.

Use the `set` form of this command to specify the destination subnet of a static route for clients to store in their routing cache.

Use the `delete` form of this command to remove the destination subnet configuration.

Use the `show` form of this command to view the destination subnet configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> static-route router <ipv4>

Specifies the router for the destination of a static route that clients can store in their routing cache.

Syntax:

```
set service dhcp-server shared-network-name name subnet ipv4net static-route router ipv4
```

Syntax:

```
delete service dhcp-server shared-network-name name subnet ipv4net static-route router
```

Syntax:

```
show service dhcp-server shared-network-name name subnet ipv4net static-route router
```

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

The IP address of the router for the destination of a static route for clients to store in their routing cache.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    static-route {
     router ipv4
   }
  }
 }
}
```

Use this command to specify the router for the destination of a static route that clients can store in their routing cache. The other part of the static route is defined by using service dhcp-server shared-network-name name subnet ipv4net static-route destination-subnet ipv4net *(page 62)*.

Use the `set` form of this command to specify the router for the destination of a static route that clients can store in their routing cache.

Use the `delete` form of this command to remove the router configuration.

Use the `show` form of this command to view the router configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> subnet-parameters <params>

Specifies additional subnet parameters for a DHCP server.

Syntax:

`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`subnet-parameters`** *params*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`subnet-parameters`** *params*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`subnet-parameters`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

params
A string of parameters to be used by the DHCP server. The string must be enclosed in single quotation marks (’).

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    subnet-parameters params
   }
  }
 }
}
```

Danger:

This feature is advanced and should be used by only expert users in special situations.

Use this command to specify additional subnet parameters for a DHCP server that are not available with the `service dhcp-server` commands. The AT&T Vyatta vRouter DHCP server commands are a subset of those that are available for DHCP server configuration. This command provides access to all DHCP server configuration parameters. More information regarding DHCP server configuration is located on the dhcpd.conf man page. To access the page, type the following at the Vyatta command prompt:

```
man dhcpd.conf
```

The AT&T Vyatta vRouter does no validation before passing the parameter string to the DHCP server process (dhcpd). Because of this nonvalidation, it is imperative that the syntax described in the `dhcpd.conf` documentation be strictly followed. Failure to do so could result in a failure of the DHCP server. It is advisable to check the system log for errors when using these parameter strings. In addition, the `show system processes` command determines if the dhcpd process is still running.

The scope of these parameters is for the specified subnet. They apply to all **`static-mappings`** within this scope unless parameters with a narrower scope are specified by using the **`static-mapping-parameters`** version of this command.

Double quotation marks (“) cannot be used within a string enclosed in single quotation marks (‘). Instead, replace all double quotation marks with &quot;.

Multiple parameter strings can be specified. Each parameter string that is specified adds a separate line to the `dhcpd.conf` file.

Use the `set` form of this command to specify additional subnet parameters for a DHCP server.

Use the `delete` form of this command to delete additional subnet parameters from a DHCP server.

Use the `show` form of this command to display additional subnet parameters for a DHCP server.

service dhcp-server shared-network-name <name> subnet <ipv4net> tftp-server-name <servername>

Specifies the name of a Trivial File Transfer Protocol (TFTP) server that is available to clients.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`tftp-server-name`** *servername*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`tftp-server-name`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`tftp-server-name`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

servername
The name of a TFTP server that is available to clients.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    tftp-server-name servername
   }
  }
 }
}
```

Use this command to specify the name of a TFTP server that is available to clients.

Use the `set` form of this command to specify the name of a TFTP server.

Use the `delete` form of this command to remove the TFTP server configuration.

Use the `show` form of this command to view the TFTP server configuration.

service dhcp-server shared-network-name <name> subnet <ipv4net> time-offset <seconds>

Specifies a time offset in seconds from Universal Time Coordinated (UTC) of the subnet of a client.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`time-offset`** *seconds*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`time-offset`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`time-offset`**

name
Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

seconds

Time offset in seconds from UTC of the subnet of a client.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    time-offset seconds
   }
  }
 }
}
```

Use this command to specify a time offset in seconds from UTC of the subnet of a client.

Use the `set` form of this command to specify a time offset.

Use the `delete` form of this command to remove a time offset.

Use the `show` form of this command to display a time offset.

service dhcp-server shared-network-name <name> subnet <ipv4net> time-server <ipv4>

Specifies the address of an RFC868 time server that is available to clients.

Syntax:

`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`time-server`** *ipv4*

Syntax:

`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`time-server`** *ipv4*

Syntax:

`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`time-server`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

Optional. The IP address of an RFC868 time server. Multiple time server addresses can be specified in separate commands. The time servers should be specified in a preferred order.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    time-server ipv4
   }
  }
 }
}
```

Use this command to specify the address of an RFC 868 time server that is available to clients.

Use the `set` form of this command to specify the address of a time server.

Use the `delete` form of this command to delete the address of a time server.

Use the `show` form of this command to display the address of a time server.

service dhcp-server shared-network-name <name> subnet <ipv4net> wins-server <ipv4>

Specifies the address of a Windows Internet Naming Server (WINS) that is available to DHCP clients.

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`wins-server`** *ipv4*

Syntax:
`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`wins-server`** *ipv4*

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`wins-server`**

name

Mandatory. A DHCP address pool.

ipv4net

Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

ipv4

Optional. Multinode. The address of a WINS. The WINS provides name-resolution services that Microsoft DHCP clients can use to correlate host names to IP addresses.

You can specify more than one WINS by entering this command multiple times. The format is an IP address.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    wins-server ipv4
   }
  }
 }
}
```

Use this command to specify the address of a WINS that is available to DHCP clients.

Use the `set` form of this command to specify the address of a WINS server.

Use the `delete` form of this command to delete the configuration of a WINS.

Use the `show` form of this command to display the configuration of a WINS.

service dhcp-server shared-network-name <name> subnet <ipv4net> wpad-url <url>

Specifies the Web Proxy Autodiscovery (WPAD) URL

Syntax:
`set service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`wpad-url`** *url*

Syntax:

`delete service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`wpad-url`**

Syntax:
`show service dhcp-server shared-network-name` *name* **`subnet`** *ipv4net* **`wpad-url`**

name
Mandatory. A DHCP address pool.

ipv4net
Mandatory. Multinode. The IPv4 network served by the DHCP address pool. The format is *ip-addr/prefix*.

url
Optional. The WPAD URL.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv4net {
    wpad-url url
   }
  }
 }
}
```

Use this command to specify the WPAD URL.

Use the `set` form of this command to specify the WPAD URL.

Use the `delete` form of this command to delete the WPAD URL.

Use the `show` form of this command to display the WPAD URL.

service dhcp-server static-mapping <mapname>

Statically maps a DHCP client, based on its MAC address, to an IP address.

Syntax:
`set service dhcp-server static-mapping` *mapname*

Syntax:
`delete service dhcp-server static-mapping` *mapname*

Syntax:
`show service dhcp-server static-mapping` *mapname*

mapname
Optional. Multinode. Allows you to statically map an IP address within an address pool to the MAC address of a device on the network.

You can define multiple static mappings of this type by creating multiple **`static-mapping`** configuration nodes.

Configuration mode

```
service {
    dhcp-server {
        static-mapping mapname {
        }
    }
}
```

Use this command to statically map a DHCP client, based on its MAC address, to an IP address.

Use the `set` form of this command to statically map a DHCP client, based on its MAC address, to an IP address.

Use the `delete` form of this command to remove the static mapping configuration.

Use the `show` form of this command to view the static mapping configuration.

service dhcp-server static-mapping <mapname> disable

Disables DHCP configuration for a static mapping.

Syntax:
`set service dhcp-server static-mapping` *mapname* **`disable`**

Syntax:
`delete service dhcp-server static-mapping` *mapname* **`disable`**

Syntax:
`show service dhcp-server static-mapping` *mapname*

The static-mapping configuration is enabled.

mapname

Optional. Multinode. Allows you to statically map an IP address within an address pool to the MAC address of a device on the network.

You can define multiple static mappings of this type by creating multiple **`static-mapping`** configuration nodes.

Configuration mode

```
service {
    dhcp-server {
        static-mapping mapname {
        disable
        }
    }
}
```

Use this command to disable DHCP configuration of a static mapping.

Use the `set` form of this command to disable DHCP configuration of a static mapping.

Use the `delete` form of this command to enable configuration of a static mapping.

Use the `show` form of this command to display the configuration of a static mapping.

service dhcp-server static-mapping <mapname> ip-address <ipv4>

Specifies a static IP address for a DHCP client.

Syntax:
`set service dhcp-server static-mapping` *mapname* **`ip-address`** *ipv4*

Syntax:
`delete service dhcp-server static-mapping` *mapname* **`ip-address`**

Syntax:
`show service dhcp-server static-mapping` *mapname* **`ip-address`**

mapname

Multinode. Allows you to statically map an IP address within an address pool to the MAC address of a device on the network.

You can define multiple static mappings of this type by creating multiple **`static-mapping`** configuration nodes.

ipv4

Mandatory. The IP address to be statically assigned to the device.

Configuration mode

```
service {
    dhcp-server {
        static-mapping mapname {
        ip-address ipv4
        }
    }
}
```

Use this command to specify a static IP address for a DHCP client, based on its MAC address.

Use the `set` form of this command to specify a static IP address for a DHCP client, based on its MAC address.

Use the `delete` form of this command to remove the static mapping configuration.

Use the `show` form of this command to view the static mapping configuration.

service dhcp-server static-mapping <mapname> mac-address <mac>

Specifies the MAC address of a DHCP client to which an IP address is assigned.

Syntax:
`set service dhcp-server static-mapping` *mapname* **`mac-address`** *mac*

Syntax:
`delete service dhcp-server static-mapping` *mapname* **`mac-address`**

Syntax:
`show service dhcp-server static-mapping` *mapname* **`mac-address`**

mapname

Multinode. Allows you to statically map an IP address within an address pool to the MAC address of a device on the network.

You can define multiple static mappings of this type by creating multiple **`static-mapping`** configuration nodes.

mac

Mandatory. The MAC address to be statically mapped to the specified IP address.

Configuration mode

```
service {
    dhcp-server {
        static-mapping mapname {
        mac-address mac
        }
    }
}
```

Use this command to specify the MAC address of a DHCP client to which an IP address is assigned.

Use the `set` form of this command to specify the MAC address of the DHCP client to which an IP address is assigned.

Use the `delete` form of this command to remove the static mapping configuration.

Use the `show` form of this command to view the static mapping configuration.

service dhcp-server static-mapping <mapname> static-mapping-parameters <params>

Specifies additional static-mapping parameters for a DHCP server.

Syntax:
`set service dhcp-server static-mapping` *mapname* **`static-mapping-parameters`** *params*

Syntax:
`delete service dhcp-server static-mapping` *mapname* **`static-mapping-parameters`** *params*

Syntax:
`show service dhcp-server static-mapping` *mapname* **`static-mapping-parameters`**

mapname

Optional. Multinode. Allows you to statically map an IP address within an address pool to the MAC address of a device on the network.

You can define multiple static mappings of this type by creating multiple **`static-mapping`** configuration nodes.

params

A string of parameters to be used by the DHCP server. The string must be enclosed in single quotation marks (').

Configuration mode

```
service {
    dhcp-server {
        static-mapping mapname {
        static-mapping-parameters params
        }
    }
}
```

Note:

This feature is advanced and should be used by only expert users in special situations.

Use this command to specify additional static-mapping parameters for a DHCP server that are not available with the `service dhcp-server` commands. The Vyatta DHCP server commands are a subset of those that are available for DHCP server configuration. This command provides access to all DHCP server configuration parameters. More information regarding DHCP server configuration is located on the dhcpd.conf man page. To access the page, type the following at the Vyatta command prompt:

```
man dhcpd.conf
```

The AT&T Vyatta vRouter does no validation before passing the parameter string to the DHCP server process (dhcpd). Because of this nonvalidation, it is imperative that the syntax described in the `dhcpd.conf` documentation be strictly followed. Failure to do so could result in a failure of the DHCP server. It is advisable to check the system log for errors when using these parameter strings. In addition, the `show system processes` command determines if the dhcpd process is still running.

The scope of these parameters is for the specified map name. They apply to all **`static-mappings`** within this scope unless parameters with a narrower scope are specified by using the **`static-mapping-parameters`** version of this command.

Multiple parameter strings can be specified. Each parameter string that is specified adds a separate line to the `dhcpd.conf` file.

Use the `set` form of this command to specify additional static-mapping parameters for a DHCP server.

Use the `delete` form of this command to remove additional static-mapping parameters from a DHCP server.

Use the `show` form of this command to display the additional static-mapping parameters for a DHCP server.

show dhcp client leases

Displays DHCP information for an interface that is configured as a DHCP client.

Syntax:

`show dhcp client leases` [**`interface`** *interface*]

interface

The identifier of an interface. Supported interface types are the following:

- **`lo`** : A loopback interface.
- `dp`*xpypz*—The name of a data plane interface, where
  — `dp`*x* specifies the data plane identifier (ID). Currently, only dp0 is supported.

  — *py* specifies a physical or virtual PCI slot index (for example, p129).

  — *pz* specifies a port index (for example, p1). For example, dp0p1p2, dp0p160p1, and dp0p192p1.

Operational mode

Use this command to display DHCP information for an interface that is configured as a DHCP client.

When used with no option, this command displays DHCP information for all interfaces that are configured as DHCP clients. When an interface is specified, this command displays DHCP information for that interface.

To configure an interface as a DHCP client, refer to documentation for that interface.

The following example shows how to display DHCP information for all interfaces that are configured as DHCP clients.

```
vyatta@R1> show dhcp client leases
interface  : dp0p1p1
ip address : 192.168.1.157      [Active]
subnet mask: 255.255.255.0
router     : 192.168.1.254
name server: 192.168.1.254 74.150.163.68 74.150.163.100
dhcp server: 192.168.1.254
lease time : 86400
last update: Wed Feb 17 02:18:20 GMT 2010
expiry     : Thu Feb 18 02:18:18 GMT 2010
reason     : BOUND
vyatta@R1>
```

show dhcp server leases

Displays current DHCP lease information.

Syntax:

`show dhcp server leases` [**`expired`** | **`pool`** *pool-name*]

`expired`

Displays expired leases.

`pool` ***pool-name***

Displays lease information for the specified address pool.

Operational mode

Use this command to display current DHCP lease information for subscribers or expired leases.

When used with no option, this command displays all current lease information. When an address pool is specified, this command displays lease information for that address pool. When the **`expired`** option is specified, only expired leases are displayed.

DHCP is configured by using service dhcp-server *(page 39)*.

The following example shows how to display all current DHCP lease information.

```
vyatta@R1> show dhcp server leases
IP address       Hardware Address   Lease expiration     Pool     Client Name
----------       ----------------   ----------------     ----     -----------
192.168.11.101   00:12:3f:e3:af:67  2007/06/23 16:28:26  POOL1    Laptop 9
vyatta@R1>
```

show dhcp server statistics

Displays DHCP server statistics.

Syntax:

`show dhcp server statistics` [**`pool`** *pool-name*]

`pool` ***pool-name***

Displays DHCP statistics for the specified address pool

Operational mode

Use this command to see current lease information for DHCP subscribers.

When used with no option, this command displays all current lease information. When address pool is provided, this command displays lease information for the specified address pool.

DHCP is configured by using service dhcp-server *(page 39)*.

The following example shows how to display all DHCP server statistics.

```
vyatta@vyatta:~$ show dhcp server statistics

Start time:                             Thu Sep  3 13:29:36 2015
Up time:                                01:19:44

Message                                 Received
DHCPDISCOVER                            1
DHCPREQUEST                             1
DHCPDECLINE                             0
DHCPRELEASE                             1
DHCPINFORM                              0

Message                                 Sent
DHCPOFFER                               1
DHCPACK                                 1
DHCPNAK                                 0

pool                                    pool size   # leased    # avail
```

```
----                                    ---------   --------    -------
myserver102                             21          0           21
myredding100                            21
```

Related commands

The following table lists related commands that are documented elsewhere.

Related commands documented elsewhere	
`set interfaces dataplane` *interface-name* **`address dhcp`**	Configure a data plane interface as a DHCP client. (Refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.)
`set interfaces dataplane` *interface-name* **`dhcp-options no-rfc3442`**	Disables support for the classless static route option for DHCP on a data plane interface. (Refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.)
`set interfaces dataplane` *interface-name* **`vif`** *vif-id* **`dhcp-options no-rfc3442`**	Disables support for the classless static route option for DHCP on a virtual interface. (Refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.)

DHCPv6

DHCPv6 overview

In general, Dynamic Host Configuration Protocol (DHCP) allows dynamic assignment of reusable IP addresses and other configuration information to DHCP clients. DHCP is described in DNS *(page 102)*. DHCPv6 provides a stateful address auto-configuration service and a stateful auto-configuration protocol for the IPv6 suite of protocols.

Although it bears many features in common with DHCP and shares a common architectural model, DHCPv6 is a separate protocol and is configured independently of DHCP. It is defined in separate protocol specification documents and the functions it provides differ in significant ways from those provided by DHCP. For example, DHCP and DHCPv6 use different UDP port numbers and they provide different sets of configuration parameters.

The AT&T Vyatta vRouter provides DHCPv6 server functionality, DHCPv6 client-side functionality (currently only available on data plane interfaces), and a DHCPv6 relay function.

There are two common usage scenarios for DHCPv6 server. The first is one in which addresses are assigned by using SLAAC and the DHCPv6 server is used only to assign parameters to the clients. The second is one in which both addresses and parameters are supplied by the DHCPv6 server. In either case, default router discovery is provided by the Neighbor Discovery (ND) protocol and so the DHCPv6 server does not need to provide that parameter.

DHCPv6 configuration

DHCPv6 configuration includes enabling the DHCPv6 server, creating a static mapping, configuring a DHCPv6 address pool, setting up a DHCPv6 relay, and setting up a DHCPv6 client.

Enabling the DHCPv6 server

To use the DHCPv6 server on the AT&T Vyatta vRouter, you must enable the DHCPv6 service. To enable the DHCPv6 service, perform the following steps in configuration mode.

Table 9: Enabling the DHCPv6 service

Step	Command
Enable the DHCPv6 server.	`vyatta@R1# set service dhcpv6-server`
Commit the information.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show service`
	` dhcpv6-server {`
	` }`

Creating a static mapping

Situations exist in which it makes sense to map a specific IPv6 address to a specific host rather than dynamically assign an IP address from a pool of addresses. This mapping is known as a static mapping.

A static mapping is defined by using the **`static-mapping`** option of the **`service dhcp-server`** configuration node.

The following example shows how to map the 2001:db8:100::101 IP address to the device with a MAC address of 00:0c:29:34:91:45.

Table 10: Creating a static mapping

Step	Command
Create a static mapping called **lab** and specify the static IP address.	`vyatta@R1#set service dhcpv6-server static-mapping lab ipv6-address 2001:db8:100::101`
Specify the host identifier string ("00:0c:29:34:91:45"- 6 bytes of host MAC address) within the static mapping called **lab**.	`vyatta@R1#set service dhcpv6-server static-mapping lab identifier 00:0c:29:34:91:45`
Commit the information.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show service dhcp-server shared-network-name LAB-NET`
	`static-mapping lab`
	`{`
	`ipv6-address 2001:db8:100::101`
	`identifier 00:0c:29:34:91:45`
	`}`

Configuring DHCPv6 address pools

Configure DHCPv6 address pools if you want the system to act as a DHCPv6 server for the network.

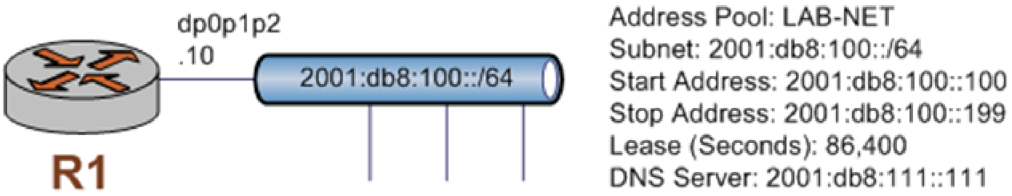
Configuring for networks directly connected to the system

The following example shows how to create an address pool within the LAB-NET shared network.

LAB-NET. This shared network serves the 2001:db8:100::/64 subnet, which is connected directly to the dp0p1p2 interface. The lease time remains at the default, 24 hours (86,400 seconds). The address pool uses the DNS name server at 2001:db8:111::111, which is on a separate subnet (not shown). The range of addresses is configured for .100 through .199.

Figure 1 *(page 76)* shows the sample address pool configuration.

Figure 5: DHCPv6 address pool configuration



Address Pool: LAB-NET
Subnet: 2001:db8:100::/64
Start Address: 2001:db8:100::100
Stop Address: 2001:db8:100::199
Lease (Seconds): 86,400
DNS Server: 2001:db8:111::111

To configure the DHCPv6 address pool, perform the following steps in configuration mode.

Table 11: Configuring a DHCPv6 address pool

Step	Command
Create the configuration node for LAB-NET on the 2001:db8:100::/64 subnet. Specify the start and stop IPv6 addresses for the pool.	`vyatta@R1# set service dhcpv6-server shared-network-name LAB-NET subnet 2001:db8:100::/64 address-range start 2001:db8:100::100 stop 2001:db8:100::199`
Specify a DNS server for LAB-NET.	`vyatta@R1# set service dhcpv6-server shared-network-name LAB-NET subnet 2001:db8:100::/64 name-server 2001:db8:111::111`
Commit the changes.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show service dhcpv6-server`
	`shared-network-name LAB-NET {`
	`subnet 2001:db8:100::/64 {`
	`address-range {`
	`start 2001:db8:100::100 {`
	`stop 2001:db8:100::199`
	`}`
	`}`
	`name-server 2001:db8:111::111`
	`}`
	`}`

Show the interface configuration.	`vyatta@R1# show interfaces`
	`dataplane dp0p1p2 {`
	`address 2001:db8:100::10/64`
	`hw-id 00:0c:29:42:05:35`
	`}`

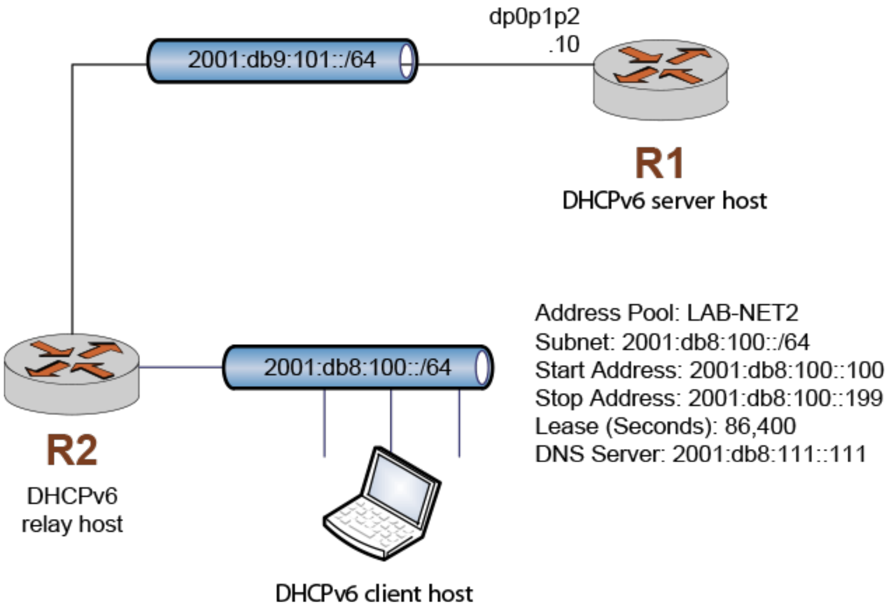
Configuring for networks indirectly connected to the system

The following example shows how to create an address pool within the LAB-NET2 shared network, which is indirectly connected through a DHCP relay server to the DHCP server on the AT&T Vyatta vRouter.

- LAB-NET2. This shared network serves the 2001:db8:100::/64 subnet, which is connected to a DHCP relay (R2), which is directly connected to the dp0p1p2 interface. The lease time remains at the default, 24 hours (86,400 seconds). The address pool uses the DNS name server at 2001:db8:111::111, which is on a separate subnet (not shown). The range of addresses is configured for .100 through .199.

The following figure shows the sample address pool configuration.

Figure 6: DHCPv6 address pool configuration using the listento option



Note: To configure the DHCPv6 address pool, perform the following steps in configuration mode.

Table 12: Configuring a DHCPv6 address pool

Step	Command
Configure the router interface to listen to DHCP messages.	`vyatta@R1# set service dhcpv6-server listento interface dp0p1p2`
Create the configuration node for LAB-NET on the 2001:db8:100::/64 subnet. Specify the start and stop IPv6 addresses for the pool.	`vyatta@R1# set service dhcpv6-server shared-network-name LAB-NET2 subnet 2001:db8:100::/64 address-range start 2001:db8:100::100 stop 2001:db8:100::199`
Specify a DNS server for LAB-NET.	`vyatta@R1# set service dhcpv6-server shared-network-name LAB-NET subnet 2001:db8:100::/64 name-server 2001:db8:111::111`
Commit the changes.	`vyatta@R1# commit`

Step	Command

Show the configuration.	`vyatta@R1# show service dhcpv6-server`
	`dhcpv6-server {`
	`       listento {`
	`               interface dp0p1p2`
	`       }`
	`} shared-network-name LAB-NET2 {`
	`     subnet 2001:db8:100::/64 {`
	`        address-range {`
	`           start 2001:db8:100::100 {`
	`              stop 2001:db8:100::199`
	`           }`
	`        }`
	`        name-server 2001:db8:111::111`
	`        lease 86400`
	`     }`
	`  }`

Show the interface configuration.	`vyatta@R1# show interfaces`
	`   dataplane dp0p1p2 {`
	`     address 2001:db9:101::0/64`
	`     hw-id 00:0c:29:42:05:35`
	`   }`

Setting up DHCPv6 relay

Configure DHCPv6 relay if you want the AT&T Vyatta vRouter to forward DHCPv6 requests to another DHCPv6 server.

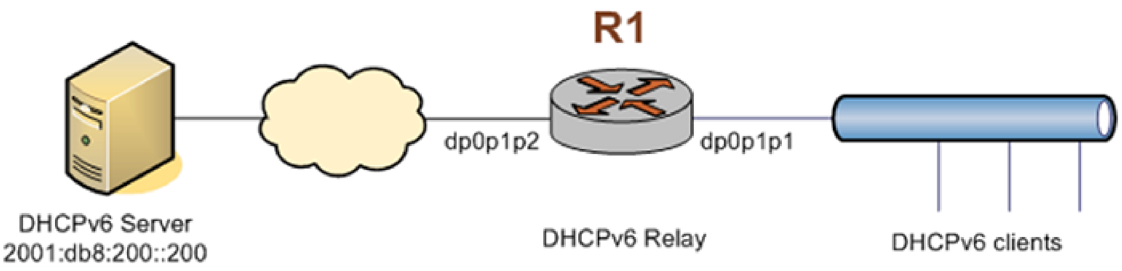
The DHCPv6 relay agent listens for requests sent by DHCPv6 clients and forwards them on to DHCPv6 servers. Because the client request packets and the relayed requests are often carried in IPv6 multicast packets, you must explicitly specify the interfaces on which the relay agent is to listen for requests and the interfaces on which it is to relay those requests.

The following procedure shows how to accomplish the following tasks:

- Configures both dp0p1p1 and dp0p1p2 for DHCPv6 relay. The system is expected to receive client requests for the DHCPv6 server through the dp0p1p1 interface. It forwards client-to-server DHCPv6 messages to the DHCPv6 server at 2001:db8:200::200 out through the dp0p1p2 interface. The DHCPv6 server refers to the interface on which client requests are received as the “listening interface,” and refers to the interface on which requests are relayed out as the “upstream interface.”
- Leaves other relay option parameters at default values. This means that R1 uses port 547 for DHCP messaging and has a maximum hop count of 10 hops.

Figure 1 *(page 79)* shows the sample DHCPv6 relay configuration.

Figure 7: DHCPv6 relay configuration



To configure DHCPv6 relay, perform the following steps in configuration mode.

Table 13: Setting up DHCPv6 relay

Step	Command

Enable DHCPv6 relay to listen on the dp0p1p1 interface.	`vyatta@R1# set service dhcpv6-relay listen-`
	`interface dp0p1p1`

Enable DHCPv6 relay to forward requests on the dp0p1p2 interface specifying the DHCPv6 server address.	`vyatta@R1# set service dhcpv6-relay upstream-`
	`interface dp0p1p2 address 2001:db8:200::200`

Commit the changes.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show service dhcpv6-relay`
	`   listen-interface dp0p1p1 {`
	`   }`
	`   upstream-interface dp0p1p2 {`
	`     address 2001:db8:200::200`
	`   }`

Setting up DHCPv6 client

Configure DHCPv6 client if you want the AT&T Vyatta vRouter to acquire an IPv6 address, parameters, or both from a DHCPv6 server. Refer to the “Ethernet Interfaces” chapter of the AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide for more information on configuring a DHCPv6 client.

To configure a DHCPv6 client, perform the following steps in configuration mode.

Table 14: Setting up a DHCPv6 client on an Ethernet interface

Step	Command

Enable DHCPv6 client on the dp0p1p1 interface.	`vyatta@R1# set interface dataplane dp0p1p1`
	` address dhcpv6`

Commit the change.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show interface dataplane dp0p1p1`
	`   address dhcpv6`
	`   hw-id b6:cc:6a:95:22:b2`

Note: After an interface acquires an IPv6 address from a DHCPv6 server, if you shut down the interface by using the **`set interfaces dataplane`** *interface-name* **`disable`** command, the interface immediately loses its IPv6 link local address. Without an IPv6 link local address, the DHCPv6 client cannot send a message to the DHCPv6 server to release the IPv6 lease and the lease remains in effect. To avoid this scenario, before shutting down a data plane interface, release the IPv6 lease for the interface by using the **`release dhcpv6 interface`** *interface-name* command.

DHCPv6 Commands

interfaces bridge <brx> dhcpv6-options

Specifies the way in which a DHCPv6 client is to acquire an address and/or parameters from a DHCPv6 server.

Syntax:
`set interfaces bridge` *brx* **`dhcpv6-options`** [**`parameters-only`** | **`temporary`**]

Syntax:
`delete interfaces bridge` *brx* **`dhcpv6-options`** [**`parameters-only`** | **`temporary`**]

Syntax:
`show interfaces bridge` *brx* **`dhcpv6-options`**

brx

The identifier for the bridge group. Supported identifiers are br0 through br999.

`parameters-only`

Acquires only configuration parameters (and not an IPv6 address) from the DHCPv6 server.

Only one of the **`parameters-only`** and the **`temporary`** parameter may be specified.

`temporary`

Acquires a temporary IPv6 address as described for IPv6 privacy addressing in RFC 4941.

Only one of the **`parameters-only`** and the **`temporary`** parameter may be specified.

Configuration mode

```
interfaces {
     bridge brx {
         dhcpv6-options [parameters-only | temporary]
     }
}
```

Use this command to specify in what way the DHCPv6 client is to acquire an IPv6 address and/or parameters from a DHCPv6 server.

Note that these parameters are only relevant if the **`dhcpv6`** option has been set for the `interfaces bridge` *brx* **`address`** *address* command. Otherwise, they are ignored.

The **`parameters-only`** option is typically used in conjunction with Stateless Address Autoconfiguration (SLAAC) or static address configuration. It and the **`temporary`** parameter are mutually exclusive.

Use the `set` form of this command to specify the DHCPv6 options.

Use the `delete` form of this command to remove the DHCPv6 options.

Use the `show` form of this command to view DHCPv6 option configuration.

release dhcpv6 interface <interface>

Releases the current DHCPv6 client lease on an interface.

Syntax:
`release dhcpv6 interface` *interface*

interface

An interface that uses DHCPv6 to obtain an IP address.

Operational mode

Use this command to release the DHCPv6 client lease on an interface. The interface must be configured to obtain an address through DHCPv6. If the DHCPv6 client is in the process of acquiring an address, it stops that process. The client does not attempt to acquire a new address through DHCPv6.

renew dhcpv6 interface <interface>

Renews the current DHCPv6 client lease on an interface.

Syntax:

`renew dhcpv6 interface` *interface*

interface

An interface that uses DHCPv6 to obtain an IP address.

Operational mode

Use this command to renew the DHCPv6 client lease on an interface. The interface must be configured to obtain an address through DHCPv6 server.

reset dhcpv6 server leases

Removes all DHCPv6 leases.

Syntax:

`reset dhcp server leases` [**`leases`** | **`lease ipv6`** *ipv6-address*]

ipv6-address

The IPv6 address with leases to be removed.

Operational mode.

This command applies to leases provided by the DHCPv6 server. The server is configured by using service dhcpv6-server *(page 86)*.

Use the `leases` command option to remove all DHCPv6 leases.

Use the `lease ipv6` command option to remove DHCPv6 leases from a particular IPv6 lease.

restart dhcpv6 relay-agent

Restarts the DHCPv6 relay agent.

Syntax:

```
restart dhcpv6 relay-agent
```

Operational mode

Use this command to stop the DHCPv6 relay agent if it is running, then start it if it is configured. This command can be used if the DHCPv6 relay agent is not operating properly.

restart dhcpv6 server

Restarts the DHCPv6 server.

Syntax:

```
restart dhcpv6 server
```

Operational mode

Use this command to stop and restart the DHCPv6 server. This command can be used if the DHCPv6 relay agent is not operating properly.

service dhcpv6-relay

Configures the system to relay DHCPv6 client messages to a DHCPv6 server.

Syntax:

```
set service dhcpv6-relay
```

Syntax:

```
delete service dhcpv6-relay
```

Syntax:

```
show service dhcpv6-relay
```

Configuration mode

```
service {
 dhcpv6-relay {
 }
}
```

Use this command to configure the system as a DHCPv6 relay agent.

You must configure the interfaces on which the system receives requests from DHCPv6 clients and the interfaces that send requests to DHCPv6 servers. The relay agent relays responses sent by the DHCPv6 servers back to the clients that sent the original request.

Use the `set` form of this command to define DHCPv6 relay configuration.

Use the `delete` form of this command to remove DHCPv6 relay configuration.

Use the `show` form of this command to view DHCPv6 relay configuration.

service dhcpv6-relay listen-interface <interface>

Specifies an interface for accepting DHCPv6 requests.

Syntax:

`set service dhcpv6-relay listen-interface` *interface* [**address** *ipv6*]

Syntax:

`delete service dhcpv6-relay listen-interface` *interface* [**address**]

Syntax:

`show service dhcpv6-relay listen-interface` *interface* [**address**]

interface

Mandatory. Multinode. An interface to accept DHCPv6 requests. At least one interface must be specified.

You can assign multiple interfaces to be used for DHCPv6 by creating multiple **`listen-interface`** configuration nodes.

ipv6

Optional. An IPv6 address on the specified interface on which to listen. If an address is not specified, one of the non-link-local addresses configured on the interface is used.

Configuration mode

```
service {
 dhcpv6-relay {
  listen-interface interface {
   address ipv6
  }
 }
}
```

Use this command to specify an interface for accepting DHCPv6 requests.

Use the `set` form of this command to specify an interface to accept DHCPv6 requests.

Use the `delete` form of this command to remove the specified value.

Use the `show` form of this command to view the specified value.

service dhcpv6-relay listen-port <port>

Specifies a port for accepting DHCPv6 requests.

Syntax:
`set service dhcpv6-relay listen-port` *port*

Syntax:
`delete service dhcpv6-relay listen-port` *port*

Syntax:
`show service dhcpv6-relay listen-port` *port*

The DHCPv6 Relay agent listens on port 547.

port

Optional. The port on which to listen for DHCPv6 requests.

Configuration mode

```
service {
 dhcpv6-relay {
  listen-port port
 }
}
```

Use this command to specify a port for accepting DHCPv6 requests.

Use the `set` form of this command to specify a port to use to accept DHCPv6 requests.

Use the `delete` form of this command to remove the specified value.

Use the `show` form of this command to view the specified value.

service dhcpv6-relay max-hop-count <count>

Specifies the maximum number of hops before discarding DHCPv6 packets.

Syntax:
`set service dhcpv6-relay max-hop-count` *count*

Syntax:
`delete service dhcpv6-relay max-hop-count` *count*

Syntax:
`show service dhcpv6-relay max-hop-count` *count*

The maximum hop count is 10.

count

Optional. The maximum hop count before discarding DHCPv6 packets. The default count is 10.

Configuration mode

```
service {
 dhcpv6-relay {
  max-hop-count count
 }
}
```

Use this command to specify the maximum number of hops before discarding DHCPv6 packets. This count is used to prevent loops.

Use the `set` form of this command to specify the maximum number of hops before discarding DHCPv6 packets.

Use the `delete` form of this command to remove the specified value.

Use the `show` form of this command to view the specified value.

service dhcpv6-relay upstream-interface <interface>

Specifies an interface for forwarding DHCPv6 requests.

Syntax:
`set service dhcpv6-relay upstream-interface` *interface* [**`address`** *ipv6*]

Syntax:
`delete service dhcpv6-relay upstream-interface` *interface* [**`address`**]

Syntax:
`show service dhcpv6-relay upstream-interface` *interface* [**`address`**]

interface

Mandatory. Multinode. An interface to forward DHCPv6 requests. At least one interface must be specified.

You can assign multiple interfaces to be used for DHCPv6 forwarding by creating multiple **`upstream-interface`** configuration nodes.

ipv6

Optional. An IPv6 address on the specified interface through which to forward queries. If an address is not specified, the queries are sent to the all DHCP relay agents and servers multicast group.

Configuration mode

```
service {
 dhcpv6-relay {
  upstream-interface interface {
   address ipv6
  }
 }
}
```

Use this command to specify an interface for forwarding DHCPv6 requests.

Use the `set` form of this command to specify an interface to use to forward DHCPv6 requests.

Use the `delete` form of this command to remove the specified value.

Use the `show` form of this command to view the specified value.

service dhcpv6-relay use-interface-id-option

Specifies that the relay agent is to insert the DHCPv6 interface ID option.

Syntax:

```
set service dhcpv6-relay use-interface-id-option
```

Syntax:

```
delete service dhcpv6-relay use-interface-id-option
```

Syntax:

```
show service dhcpv6-relay use-interface-id-option
```

The DHCPv6 interface ID option is not inserted if a single listening interface is defined, but is inserted automatically if more than one listening interface is defined.

Configuration mode

```
service {
 dhcpv6-relay {
  use-interface-id-option
 }
}
```

Use this command to specify that DHCPv6 is to insert the interface ID option. Note that this option is automatically inserted when two or more listening interfaces are configured, so this parameter affects just system behavior when only one listening interface is configured.

Use the `set` form of this command to specify that DHCPv6 is to insert the interface ID option.

Use the `delete` form of this command to return the system to its default behavior.

Use the `show` form of this command to view the specified value.

service dhcpv6-server

Enables DHCPv6 server functionality.

Syntax:

```
set service dhcpv6-server
```

Syntax:

```
delete service dhcpv6-server
```

Syntax:

```
show service dhcpv6-server
```

Configuration mode

```
service {
 dhcpv6-server {
 }
}
```

Use the `set` form of this command to enable DHCPv6 server functionality.

Use the `delete` form of this command to remove DHCPv6 server functionality.

Use the `show` form of this command to view DHCPv6 server configuration.

service dhcpv6-server listento interface <dp-interface>

Allows the DHCP server to create address pools for clients that are indirectly connected to a data plane network interface through a DHCP relay server.

Syntax:

`set service dhcpv6-server listento interface` *dp-interface*

Syntax:
`delete service dhcpv6-server listento interface`

Syntax:
`show service dhcpv6-server`

dp-interface
A data plane interface on the router. It must have a valid IP address.

Configuration mode.

```
service {
    dhcpv6-server {
        listento {
        interface dp-interface
        }
    }
}
```

Use this command to enable the DHCP server to create IP address pools for clients that are not directly connected to the router. For example, if clients on the B subnet connect to the router through a DHCP relay server, the DHCP relay server connects to the router through a data plane interface on the A subnet, and the data plane interface has a valid IP address, using this command allows the DHCP server to create IP address pools for clients that are on the B subnet.

Use the `set` form of this command to create an IP address pool for clients that are indirectly connected to the router through a data plane network interface.

Use the `delete` form of this command to remove a data-plane interface from the DHCP server configuration. If no data plane interfaces are configured, the DHCP server cannot create address pools.

Use the `show` form of this command to view the DHCP server configuration.

service dhcpv6-server preference <preference>

Specifies the DHCPv6 server preference.

Syntax:
`set service dhcpv6-server preference` *preference*

Syntax:
`delete service dhcpv6-server preference`

Syntax:
`show service dhcpv6-server preference`

The DHCPv6 server preference is not set.

preference
Optional. The preference for the DHCPv6 server. The preference ranges from 0 through 255.

Configuration mode

```
service {
 dhcpv6-server {
  preference preference
 }
}
```

Use this command to specify the DHCPv6 server preference to DHCPv6 clients. When clients receive advertise messages from multiple servers that include preferences, they choose the server with the highest preference.

Use the `set` form of this command to specify the DHCPv6 server preference.

Use the `delete` form of this command to restore the default state, that is, the DHCPv6 server preference is not set.

Use the `show` form of this command to display the DHCPv6 server preference.

service dhcpv6-server shared-network-name <name>

Assigns a name to a physical subnet.

Syntax:
`set service dhcpv6-server shared-network-name` *name*

Syntax:
`delete service dhcpv6-server shared-network-name` *name*

Syntax:
`show service dhcpv6-server shared-network-name` *name*

name

Multinode. The name for a physical subnet.

You can define multiple subnets by creating multiple **`shared-network-name`** configuration nodes, each with a different name.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
  }
 }
}
```

Use this command to assign a name to a physical subnet. The subnet created may be directly connected to the system. The name is arbitrary and need not match any name used for this subnet elsewhere within the system.

Use the `set` form of this command to assign a name to a physical subnet.

Use the `delete` form of this command to delete the name of a physical subnet.

Use the `show` form of this command to display the name of a physical subnet.

service dhcpv6-server shared-network-name <name> subnet <ipv6net>

Specifies an IPv6 subnet to which the DHCPv6 server provides access.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net*

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net*

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet to which the DHCPv6 server provides access. The format is *ipv6-addr/prefix*.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
   }
  }
 }
}
```

Use this command to specify an IPv6 subnet to which the DHCPv6 server provides access. The DHCPv6 server responds to clients on this subnet by using the parameters and addresses defined in this subtree.

Use the `set` form of this command to specify the DHCPv6 subnet.

Use the `delete` form of this command to remove DHCPv6 subnet configuration.

Use the `show` form of this command to view DHCPv6 subnet configuration.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> address-range

Specifies a range of IPv6 addresses that can be assigned to clients.

Syntax:

`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`address-range`**

Syntax:

`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`address-range`**

Syntax:

`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`address-range`**

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    address-range {
    }
   }
  }
 }
}
```

Use this command to specify a range of IPv6 addresses that can be assigned to clients. If no address range is provided, the DHCPv6 server operates in a stateless mode on this subnet, which means that it does not assign dynamic IPv6 addresses and thus does not maintain state information about those assignments.

Use the `set` form of this command to create the address-range configuration node.

Use the `delete` form of this command to remove the address-range configuration.

Use the `show` form of this command to view the address-range configuration.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> address-range prefix <pool-ipv6net>

Specifies a pool of IPv6 addresses that can be assigned to clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`address-range`** **`prefix`** *ipv6net* [**`temporary`**]

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`address-range`** **`prefix`** *ipv6net* [**`temporary`**]

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`address-range`** **`prefix`** *ipv6net*

name
The name of a physical subnet.

ipv6net
Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

pool-ipv6net
Optional. An IPv6 address prefix that defines a pool of consecutive addresses available for assignment to clients. The specified prefix must be a subset of the subnet prefix.

`temporary`
Optional. If specified, indicates that the range can be used for assigning privacy addresses (RFC 4941).

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    address-range {
     prefix pool-ipv6net {
      temporary
     }
    }
   }
  }
 }
}
```

Use this command to specify a pool of IPv6 addresses that can be assigned to clients.

Use the `set` form of this command to create the address-range prefix configuration.

Use the `delete` form of this command to remove the address-range prefix configuration.

Use the `show` form of this command to view the address-range prefix configuration.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> address-range start <start-ipv6>

Specifies the beginning and ending addresses in a range of IPv6 addresses that can be assigned to clients.

Syntax:

```
set service dhcpv6-server shared-network-name name subnet ipv6net address-range start start-ipv6 [
stop stop-ipv6 | temporary ]
```

Syntax:

```
delete service dhcpv6-server shared-network-name name subnet ipv6net address-range start start-ipv6 [
stop | temporary ]
```

Syntax:

```
show service dhcpv6-server shared-network-name name subnet ipv6net address-range start start-ipv6 [
stop | temporary ]
```

name
: The name of a physical subnet.

ipv6net
: Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

start-ipv6
: Optional. Multinode. The beginning address in a range of consecutive IPv6 addresses that are available for assignment to clients.

stop-ipv6
: Optional. The ending address in a range of consecutive IPv6 addresses that are available for assignment to clients. If not specified, only the beginning address is available for assignment.

`temporary`
: Optional. If specified, indicates that the range can be used for assigning privacy addresses (RFC 4941).

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    address-range {
     start start ipv6 {
      stop stop ipv6
       temporary
     }
    }
   }
  }
 }
}
```

Use this command to specify the beginning and ending addresses in a range of IPv6 addresses that can be assigned to clients.

Use the `set` form of this command to create the address-range configuration.

Use the `delete` form of this command to remove the address-range configuration.

Use the `show` form of this command to view the address-range configuration.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> description <desc>

Provides a description of a subnet.

Syntax:

```
set service dhcpv6-server shared-network-name name subnet ipv6net description desc
```

Syntax:

```
delete service dhcpv6-server shared-network-name name subnet ipv6net description
```

Syntax:

```
show service dhcpv6-server shared-network-name name subnet ipv6net description
```

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

desc

A description of the specified subnet.

Configuration mode

```
service {
 dhcp-server {
  shared-network-name name {
   subnet ipv6net {
    description desc
   }
  }
 }
}
```

Use this command to provide a description of a subnet.

Use the `set` form of this command to provide a description of a subnet.

Use the `delete` form of this command to delete the description of a subnet.

Use the `show` form of this command to display the description of a subnet.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> domain-search <domain>

Specifies a domain name to include in the domain search list.

Syntax:

```
set service dhcpv6-server shared-network-name name subnet ipv6net domain-search domain
```

Syntax:

```
delete service dhcpv6-server shared-network-name name subnet ipv6net domain-search domain
```

Syntax:

```
show service dhcpv6-server shared-network-name name subnet ipv6net domain-search
```

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

domain

Multinode. A domain name to include in the domain search list.

You can specify more than one domain name by including this parameter multiple times.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    domain-search domain
   }
   }
  }
 }
```

Use this command to specify a domain name to include in the domain search list. Hosts use the domain search list when resolving host names in DNS. Values are listed in the option, and communicated to the client, in the order entered.

Use the `set` form of this command to specify a domain name.

Use the `delete` form of this command to delete a domain name.

Use the `show` form of this command to view the domain name configuration.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> lease-time

Sets the client lease time.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`lease-time`** { **`default`** *default-time* | **`maximum`** *max-time* | **`minimum`** *min-time* }

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`lease-time`** { **`default`** | **`maximum`** | **`minimum`** }

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`lease-time`** { **`default`** | **`maximum`** | **`minimum`** }

name
The name of a physical subnet.

ipv6net
Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

default-time
The default lease time in seconds. The time is assigned to a client if the client does not request a specific lease time.

max-time
The maximum lease time in seconds that is assigned to a lease. If the client requests a time larger than the maximum lease time, the maximum time is used.

min-time
The minimum lease time in seconds that is assigned to a lease. If the client requests a time smaller than the minimum lease time, the minimum time is used.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    lease-time {
     default default-time
     maximum max-time
     minimum min-time
    }
   }
  }
 }
}
```

Use this command to specify the client lease time.

Use the `set` form of this command to specify the lease time.

Use the `delete` form of this command to delete the lease time.

Use the `show` form of this command to display the lease time.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> name-server <ipv6>

Specifies the address of a recursive DNS server (RDNSS) for DHCPv6 clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`name-server`** *ipv6*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`name-server`** *ipv6*

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`name-server`**

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

ipv6

Multinode. The IPv6 address of an RDNSS.

You can specify more than one server by specifying this parameter multiple times.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    name-server ipv6
   }
  }
 }
}
```

Use this command to specify the address of an RDNSS that is available to DHCPv6 clients. Addresses are listed in the order they are specified.

Use the `set` form of this command to specify the address of an RDNSS.

Use the `delete` form of this command to delete the address of an RDNSS.

Use the `show` form of this command to display the address of an RDNSS.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-domain <nis-domain-name>

Specifies the Network Information Service (NIS) domain for DHCPv6 clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nis-domain`** *nis-domain-name*

Syntax:

`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nis-domain`**

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nis-domain`**

name
The name of a physical subnet.

ipv6net
Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

nis-domain-name
The name of the NIS domain for DHCPv6 clients.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    nis-domain nis-domain-name
   }
  }
 }
}
```

Use this command to specify an NIS domain for DHCPv6 clients.

Use the `set` form of this command to specify the NIS domain.

Use the `delete` form of this command to delete the NIS domain.

Use the `show` form of this command to display the NIS domain.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-domain <nisplus-domain-name>

Specifies the Network Information Service Plus (NIS+) domain for DHCPv6 clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nisplus-domain`** *nisplus-domain-name*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nisplus-domain`**

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nisplus-domain`**

name
The name of a physical subnet.

ipv6net
Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

nisplus-domain-name
The name of an NIS+ domain for DHCPv6 clients.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    nisplus-domain nisplus-domain-name
   }
   }
  }
 }
```

Use this command to specify an NIS+ domain for DHCPv6 clients.

Use the `set` form of this command to specify the NIS+ domain.

Use the `delete` form of this command to delete the NIS+ domain.

Use the `show` form of this command to display the NIS+ domain.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> nisplus-server <ipv6>

Specifies the address of the Network Information Service Plus (NIS+) server for DHCPv6 clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nisplus-server`** *ipv6*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nisplus-server`** *ipv6*

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nisplus-server`**

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

ipv6

Multinode. The address of the NIS+ server for DHCPv6 clients.

You can specify more than one address by issuing this statement multiple times.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    nisplus-server ipv6
   }
  }
 }
}
```

Use this command to specify the address of the NIS+ server for DHCPv6 clients. Addresses are listed in the order they are specified.

Use the `set` form of this command to specify the NIS+ server address.

Use the `delete` form of this command to delete the address of the NIS+ server.

Use the `show` form of this command to display the address of the NIS+ server.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> nis-server <ipv6>

Specifies the address of the NIS server for DHCPv6 clients.

Syntax:

`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nis-server`** *ipv6*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nis-server`** *ipv6*

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`nis-server`**

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

ipv6

Multinode. The address of the NIS server for DHCPv6 clients.

You can specify more than one address by entering this command multiple times.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    nis-server ipv6
   }
  }
 }
}
```

Use this command to specify the address of the NIS server for DHCPv6 clients. Addresses are listed in the order they are specified.

Use the `set` form of this command to specify the address of the NIS server.

Use the `delete` form of this command to delete the address of the NIS server.

Use the `show` form of this command to display the address of the NIS server.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-address <ipv6>

Specifies the address of the Session Initiation Protocol (SIP) server for DHCPv6 clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sip-server-address`** *ipv6*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sip-server-address`** *ipv6*

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sip-server-address`**

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

ipv6

Multinode. The address of the SIP server for DHCPv6 clients.

You can specify more than one address by entering this command multiple times.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    sip-server-address ipv6
   }
  }
 }
}
```

Use this command to specify the address of the SIP server for DHCPv6 clients. Addresses are listed in the order they are specified.

Use the `set` form of this command to specify the address of the SIP server.

Use the `delete` form of this command to delete the address of the SIP server.

Use the `show` form of this command to display the address of the SIP server.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> sip-server-name <sip-server-name>

Specifies the name of the Session Initiation Protocol (SIP) server for DHCPv6 clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sip-server-name`** *sip-server-name*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sip-server-name`** *sip-server-name*

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sip-server-name`**

name

The name of a physical subnet.

ipv6net

Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

sip-server-name

Multinode. The name of the SIP server for DHCPv6 clients.

You can specify more than one name by entering this command multiple times.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    sip-server-name sip-server-name
   }
  }
 }
}
```

Use this command to specify the name of the SIP server for DHCPv6 clients. Addresses are listed in the order they are specified.

Use the `set` form of this command to specify the SIP server name.

Use the `delete` form of this command to delete the name of the SIP server.

Use the `show` form of this command to display the name of the SIP server.

service dhcpv6-server shared-network-name <name> subnet <ipv6net> sntp-server-address <ipv6>

Specifies the address of the Simple Network Time Protocol (SNTP) for DHCPv6 clients.

Syntax:
`set service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sntp-server-address`** *ipv6*

Syntax:
`delete service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sntp-server-address`** *ipv6*

Syntax:
`show service dhcpv6-server shared-network-name` *name* **`subnet`** *ipv6net* **`sntp-server-address`**

name
The name of a physical subnet.

ipv6net
Optional. Multinode. An IPv6 subnet served by the DHCPv6 server. The format is *ipv6-addr/prefix*.

ipv6
Multinode. The address of the SNTP server for DHCPv6 clients.

You can specify more than one address by entering this command multiple times.

Configuration mode

```
service {
 dhcpv6-server {
  shared-network-name name {
   subnet ipv6net {
    sntp-server-address ipv6
   }
  }
 }
}
```

Use this command to specify the address of the SNTP server for DHCPv6 clients. Addresses are listed in the order they are specified. SNTP is a subset of NTP and includes extensions to operate over IPv6. It is specified in: http://tools.ietf.org/html/rfc4330.

Use the `set` form of this command to specify the address of the SNTP server.

Use the `delete` form of this command to delete the address of the SNTP server.

Use the `show` form of this command to display the address of the SNTP server.

service dhcpv6#server static#mapping <mapping#name>

Specifies the IPv6 address for a client.

Syntax:
`set service dhcpv6-server static-mapping` *mapping#name* [**`ipv6-address`** *ipv6* | **`identifier`** *identifier*]

Syntax:
`delete service dhcpv6-server static-mapping` *mapping#name* [**`ipv6-address`** | **`identifier`**]

Syntax:
`show service dhcpv6-server static-mapping` *mapping#name* [**`ipv6-address`** | **`identifier`**]

mapping-name
A name to identify the static mapping.

ipv6

The IPv6 address that is assigned to a client.

identifier

The character string that identifies a client. It is compared against the ia-na option sent by the client. This value is typically the low-order 4 bytes of the MAC address of the client.

Configuration mode.

```
service {
    dhcpv6-server {
        static-mapping mapping-name {
        ipv6-address ipv6
        identifier identifier
        }
    }
}
```

Use this command to specify the IPv6 address for a client.

Use the `set` form of this command to create the prefix-delegation configuration.

Use the `delete` form of this command to remove the prefix-delegation configuration.

Use the `show` form of this command to view the prefix-delegation configuration.

show dhcpv6 client leases

Displays DHCPv6 information for interfaces that are configured as DHCPv6 clients.

Syntax:

```
show dhcpv6 client leases
```

Operational mode

Use this command to display current DHCPv6 client information for interfaces that are configured as DHCPv6 clients. If an address has been acquired, the command shows the lease parameters associated with that address, including the unique ID, assigned IPv6 address, and time remaining on the lease.

To configure an interface as a DHCPv6 client, refer to the documentation for that interface.

show dhcpv6 relay-agent status

Displays the status of the DHCPv6 relay agent.

Syntax:

```
show dhcpv6 relay-agent status
```

Operational mode

Use this command to display the status of the DHCPv6 relay agent. This status includes an indication of whether the DHCPv6 relay agent is configured. If it is configured, the command indicates whether the DHCPv6 relay agent is running.

show dhcpv6 server leases

Displays the status of all leases assigned by the DHCPv6 server.

Syntax:

```
show dhcpv6 server leases
```

Operational mode

Use this command to display the status of all leases assigned by the DHCPv6 server. For each lease, it shows the unique ID of the client, assigned IPv6 address, and time remaining on the lease.

show dhcpv6 server status

Displays the status of the DHCPv6 server.

Syntax:

```
show dhcpv6 server status
```

Operational mode

Use this command to display the status of the DHCPv6 server. This status includes an indication of whether the DHCPv6 server is configured. If it is configured, the command indicates whether the DHCPv6 server is running. The command notes whether any address ranges are configured. If none are configured, the server can assign only parameters; it cannot assign addresses.

DNS

DNS overview

DNS is an Internet directory service that provides mappings between human-readable domain names and numeric IP addresses. DNS mappings are recorded in resource records that are stored on name servers distributed throughout the Internet. A device needing to access a host across the Internet sends a DNS query to a name server. The name server consults its resource records and returns an answer with the IP address of the specified name.

The DNS system has billions of resource records. If the requested record is not local to the consulted name server, the name server consults another name server, and so on, until the requested information is located and returned.

There are billions of resource records in the DNS system. To keep the data manageable, the records are divided into zones, which contain resource records for a DNS domain or subdomain.

The AT&T Vyatta vRouter supports three main DNS-related features:

- System DNS *(page 102)*
- Dynamic DNS *(page 102)*
- DNS forwarding *(page 102)*

System DNS

In system DNS, you define the list of name servers that the AT&T Vyatta vRouter can use to resolve host names to IP addresses. This list is created by using the `system name-server` command. (The `system name-server` command is described in AT&T Vyatta Network Operating System Basic System Configuration Guide; for your convenience, an example of system DNS is provided in Configuring access to a name server *(page 103)*.)

Dynamic DNS

Originally, DNS mappings were statically specified in “zone files,” which were periodically loaded onto DNS servers. These zone files worked reasonably well at a time when most hosts were configured with static IP addresses. However, since the 1990s, many network endpoints have been assigned IP addresses using dynamic protocols such as DHCP. Until 1997, devices with DHCP-assigned IP addresses essentially could not participate in the DNS system.

In 1997, the Internet Engineering Task Force (IETF) published RFC 2136, *Dynamic Updates in the Domain Name System* , describing the dynamic DNS update protocol. Dynamic DNS (DDNS) provides a mechanism for DNS entries to be established and removed dynamically. Devices using dynamic DNS can notify a domain name server in real time of changes to host name, IP address, or other DNS-related information.

This feature is particularly useful for systems in which a dynamic IP address is provided by the ISP. Whenever the IP address changes, the AT&T Vyatta vRouter updates a DDNS service provider with the change. The DDNS provider is responsible for propagating this change to other DNS servers. The AT&T Vyatta vRouter supports a number of DDNS providers.

DNS forwarding

In many environments that use consumer-level ISP connections, the ISP both assigns the client router with its IP address and notifies the client router of the DNS server to use. In many cases, the IP address of the DNS server itself is assigned through DHCP and changes periodically; the ISP notifies the client router of the change in DNS server IP address through periodic updates. This makes it problematic to statically configure a DNS server IP address on the DHCP server of the client router for its LAN clients.

In cases like these, the AT&T Vyatta vRouter can use DNS forwarding (also called DNS relay) to maintain connectivity between hosts on its network and the DNS server of the ISP.

When DNS forwarding is used, the client router offers its own client-side IP address (which is static) as the DNS server address to the hosts on its network, so that all client DNS requests are made to the client-side address of the client router. When DNS requests are made, the client router forwards them to the ISP DNS server; answers are directed back to the client router and forwarded through to the client hosts. If the ISP changes the address of its DNS server, the client router simply records the new address of the server. The server address remains unchanged from the perspective of the LAN clients.

Another advantage to DNS forwarding is that DNS requests are cached in the AT&T Vyatta vRouter (until either the time-to-live value in the DNS record expires or the cache fills). Subsequent requests for a cached entry are responded to locally, with a corresponding reduction in WAN traffic.

DNS configuration examples

This section presents the following topics:

- Configuring access to a name server *(page 103)*
- Configuring dynamic DNS *(page 103)*
- Configuring DNS forwarding *(page 105)*
- Statically configured entries and DNS forwarding *(page 108)*

Configuring access to a name server

To be able to translate host names (such as www.att.com) to IP addresses (such as 69.59.140.141), the system must be able to access a DNS server.

Configuring access to a DNS server is a function of basic system management, and is described in AT&T Vyatta Network Operating System Basic System Configuration Guide. For your convenience, the configuration example is repeated here.

Table 1 *(page 103)* configures a static IP address for the DNS server at address 12.34.56.100. To configure the AT&T Vyatta vRouter in this way, perform the following steps.

Table 15: Configuring static access to a DNS name server

Step	Command
Specify the IP address of the DNS server.	`vyatta@R1# set system name-server 12.34.56.100`

Configuring dynamic DNS

Figure 1 *(page 105)* shows a typical DDNS scenario. In this scenario:

- The AT&T Vyatta vRouter (R1) is connected to an ISP via dp0p1p1.
- The network domain is company.com.
- The AT&T Vyatta vRouter host name is r1.company.com.
- The web server of the company is located behind the AT&T Vyatta vRouter. Its host name is www.company.com.
- The ISP is providing dynamic IP addresses to its clients through DHCP.
- The IP address of the dp0p1p1 interface in the AT&T Vyatta vRouter changes over time because of the dynamic assignment by the ISP.
- The web server of the company is behind a Network Address Translation (NAT) device on the AT&T Vyatta vRouter, so its IP address (as viewed from the Internet) changes when the ISP assigns a new address to the dp0p1p1 interface.
- Because the web address of the server changes, responses to DNS queries for www.company.com must also change to the new IP address. DDNS resolves this problem.

DDNS allows the AT&T Vyatta vRouter (R1) to update the DNS system with the new IP address information for any local host names (for example, r1.company.com, and www.company.com) whenever the IP address on dp0p1p1 changes. The setup process is as follows:

1. Sign up for DDNS service from one of the supported service providers:

   Choose from:
   - DNS Park: www.dnspark.com
   - DSL Reports: www.dslreports.com
   - DynDNS: www.dyndns.com
   - easyDNS: www.easydns.com
   - namecheap: www.namecheap.com
   - Sitelutions: www.sitelutions.com
   - zoneedit: www.zoneedit.com

   Info:

   The individual providers offer instructions for sign-up.

   Note: Depending on the service provider, host names may need to include the domain name (for example, `www` instead of `www.company.com`).

2. Configure the AT&T Vyatta vRouter (R1 in the example) with service provider information such as the service name, a login ID, and a password so that the system can determine how to log on and send updates to the DDNS service provider.
3. Configure the AT&T Vyatta vRouter with the host names that must be updated in the DNS system when the IP address on dp0p1p1 changes.

Setting up Dynamic DNS

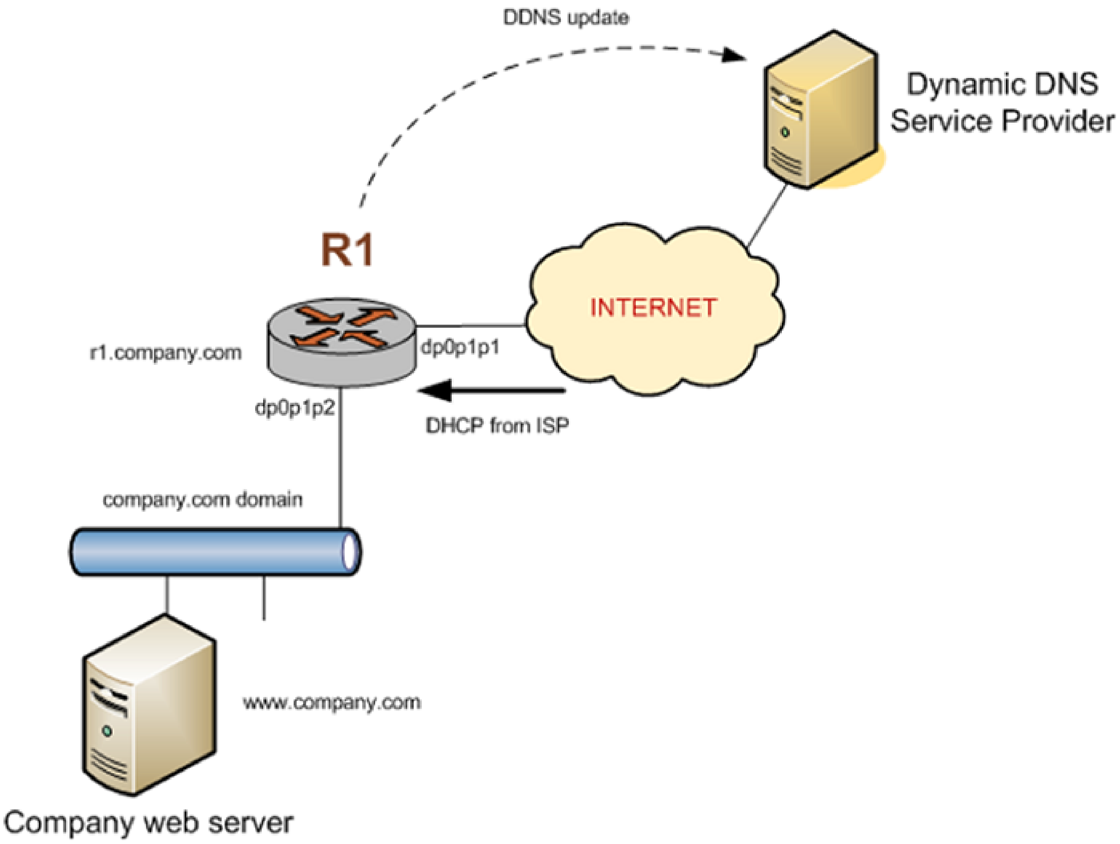


Figure 8: Dynamic DNS

The following example shows how to set up DDNS for DDNS service provider DynDNS. It is assumed for this example that you have already signed up with DynDNS). To configure the AT&T Vyatta vRouter in this way, perform the following steps in configuration mode.

Table 16: Setting up dynamic DNS

Step	Command

Set the service provider.	`vyatta@R1# set service dns dynamic interface`
	` dp0p1p1 service dyndns`

Set the DDNS service provider login ID (for example, vtest).	`vyatta@R1# set service dns dynamic interface`
	` dp0p1p1 service dyndns login vtest`

Set the DDNS service provider password (for example, testpwd).	`vyatta@R1# set service dns dynamic interface`
	` dp0p1p1 service dyndns password testpwd`

Specify R1 as a host name whose DNS entry needs to be updated when the IP address on dp0p1p1 changes.	`vyatta@R1# set service dns dynamic`
	` interface dp0p1p1 service dyndns host-name`
	` r1.company.com`

Specify www as a host name whose DNS entry needs to be updated when the IP address on dp0p1p1 changes.	`vyatta@R1# set service dns dynamic`
	` interface dp0p1p1 service dyndns host-name`
	` www.company.com`

Commit the changes.	`vyatta@R1# commit OK`

Show the dynamic DNS configuration.	`vyatta@R1# show service dns dynamic`
	`interface dp0p1p1 {`
	` service dyndns {`
	`   host-name r1.company.com`
	`   host-name www.company.com`
	`   login vtest`
	`   password testpwd`
	` }`
	`}`

At this point, whenever the IP address on dp0p1p1 changes, the AT&T Vyatta vRouter automatically logs onto the DynDNS service by using the vtest login ID and the testpwd password. It sends an update for the r1.company.com and www.company.com host names specifying the new IP address required to reach those hosts on the company.com domain. External users that query DNS for r1.company.com or www.company.com are subsequently answered with the new address from the DNS system.

> **Note:** Dynamic DNS updates are logged. To see the updates, set up logging by using the system **`syslog global facility`** *facility* **`level`** *level* command, where *facility* is daemon and *level* is notice. See AT&T Vyatta Network Operating System Basic System Configuration Guide for details.

Configuring DNS forwarding

Configuring the AT&T Vyatta vRouter for DNS forwarding has two main steps:

1. Specifying the DNS name servers to which to forward
2. Specifying the interfaces on which to listen for DNS requests

Specifying DNS Name Servers

Name server locations can be obtained in three ways:

- From the system name server list, defined by using the `set system name-server` command
- By DHCP
- By listing additional name servers by using service dns forwarding dhcp <interface> *(page 114)*

By default, the AT&T Vyatta vRouter forwards DNS requests to name servers on the system name server list plus name servers obtained through DHCP. You can override the default behavior by specifying any or all of the following:

- Specifically use system-defined name servers. To do this, use service dns forwarding system *(page 116)*.
- Specifically use name servers received for the interface that is using DHCP client to get an IP. To do this, use service dns forwarding dhcp <interface> *(page 114)*.
- List additional name servers by using service dns forwarding name-server <ipv4> *(page 115)*.

These three options can be used in any combination; however, using any of them eliminates the default DNS forwarding behavior.

When DNS forwarding starts or restarts, it broadcasts a message to all the name servers in the pool and selects the first name server to answer. This name server is used unless it becomes unreachable, in which case the system sends another broadcast message to the remaining name servers in the pool.

Specifying the Listening Interfaces

The listening interfaces are the interfaces to which internal clients forward DNS requests. The DNS forwarding service listens for these requests and forwards them to the name server.

To set the listening interface, use service dns forwarding listen-on <interface> *(page 115)*. You can specify more than one interface by issuing this command multiple times.

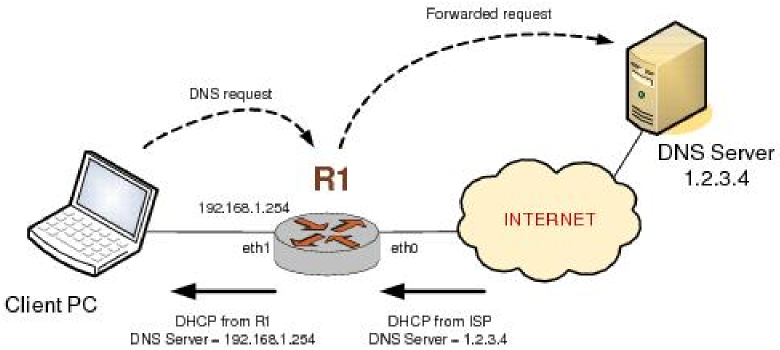
DNS Forwarding Scenario

After these steps are completed, DNS forwarding is set up. At this point, the AT&T Vyatta vRouter DHCP server can be used to distribute the DNS forwarding interface address to DHCP clients. (For information about setting up a DHCP server on the AT&T Vyatta vRouter, see DHCP *(page 20)*.

Figure 1 *(page 107)* shows a typical scenario in which DNS forwarding is deployed. In this scenario:

- The ISP is providing dynamic IP addresses to its customers, including an AT&T Vyatta vRouter (R1) through DHCP.
- The AT&T Vyatta vRouter (R1) is providing DHCP service to clients on its local network.
- Local clients send DNS requests to the AT&T Vyatta vRouter.
- The DNS forwarding service on the AT&T Vyatta vRouter forwards the requests to the the DNS server of the ISP.

Figure 9: Scenario using DNS forwarding



The following example shows how to set up the key parts of the AT&T Vyatta vRouter for the preceding scenario. To configure the AT&T Vyatta vRouter in this way, perform the following steps in configuration mode.

Table 17: Setting up DNS forwarding

Step	Command
Set IP address and prefix on dp0p1p2.	`vyatta@R1# set interfaces dataplane dp0p1p2 address 192.168.1.254/24`
Set dp0p1p1 as a DHCP client.	`vyatta@R1# set interfaces dataplane dp0p1p1 address dhcp`
Set up the DHCP server on R1 by creating the configuration node for dp0p1p2_POOL on subnet 192.168.1.0/24. Specify the start and stop IP addresses for the pool.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_POOL subnet 192.168.1.0/24 start 192.168.1.100 stop 192.168.1.199`
Specify the default router for dp0p1p2_POOL.	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_POOL subnet 192.168.1.0/24 default-router 192.168.1.254`
Create a DNS server list using DNS server information provided by the DHCP server of the ISP (on dp0p1p1).	`vyatta@R1# set service dns forwarding dhcp dp0p1p1`
Listen for DNS requests on dp0p1p2.	`vyatta@R1# set service dns forwarding listen-on dp0p1p2`
Specify a DNS server for dp0p1p2_POOL (in this case, it acts as a DNS forwarder).	`vyatta@R1# set service dhcp-server shared-network-name dp0p1p2_POOL subnet 192.168.1.0/24 dns-server 192.168.1.254`
Commit the changes.	`vyatta@R1# commit`

Step	Command

Show the DNS-related configuration.	vyatta@R1# show service dns
	forwarding {
	     dhcp dp0p1p1
	     listen-on dp0p1p2
	}

Statically configured entries and DNS forwarding

Because of difficulties interworking with network address translation (NAT) on the corporate gateway, it is sometimes difficult to obtain correct IP addresses for hosts on the corporate network. To work around this problem, you can create static entries on a local AT&T Vyatta vRouter by using the `system static-host-mapping` command. Any entries configured in this way are compared with incoming DNS queries before the query is passed to DNS forwarding. If a match is found, the corresponding IP address is returned.

The following table shows how to set up the system to return an IP address of 12.34.56.78 if it receives a DNS query for either vyatta.com or vdut1.

Table 18: Setting up static entries

Step	Command
Create the static host-mapping configuration node.	vyatta@R1# set system static-host-mapping host-name vyatta.com
Provide an alias host name (this step is optional).	vyatta@R1# set system static-host-mapping host-name vyatta.com alias vdut1
Specify the IP address to be returned in response to the DNS query.	vyatta@R1# set system static-host-mapping host-name vyatta.com inet 12.34.56.78
Commit the changes.	vyatta@R1# commit

Show the static host-mapping configuration.	vyatta@R1# show system static-host-mapping
	 host-name vyatta.com{
	    alias vdut1
	    inet 12.34.56.78
	 }

DNS Commands

reset dns forwarding all

Resets all counters related to DNS forwarding and resets the DNS forwarding cache.

Syntax:

`reset dns forwarding all`

Operational mode

Use this command to reset all counters related to DNS forwarding and remove all entries from the DNS forwarding cache.

reset dns forwarding cache

Removes all entries from the DNS forwarding cache.

Syntax:

`reset dns forwarding cache`

Operational mode

Use this command to remove all entries from the DNS forwarding cache.

service dns dynamic interface <interface>

Enables support of dynamic DNS (DDNS) on an interface.

Syntax:

`set service dns dynamic interface` *interface*

Syntax:

`delete service dns dynamic interface` *interface*

Syntax:

`show service dns dynamic interface` *interface*

interface

Multinode. An interface that is to support DDNS.

You can have more than one interface that supports DDNS by creating multiple **`interface`** configuration nodes.

Configuration mode

```
service {
 dns {
  dynamic {
   interface interface {
   }
  }
 }
}
```

By default, this command applies to the default routing table.

Use this command to enable support of DDNS on an interface.

Use the `set` form of this command to enable support of DDNS on an interface.

Use the `delete` form of this command to disable DDNS on an interface and remove all its dynamic DNS configuration.

Use the `show` form of this command to view DDNS configuration.

service dns dynamic interface <interface> service <service>

Specifies a dynamic DNS (DDNS) service provider.

Syntax:

```
set service dns dynamic interface interface service service
```

Syntax:

```
delete service dns dynamic interface interface service service
```

Syntax:

```
show service dns dynamic interface interface service service
```

interface

Multinode. An interface that supports DDNS.

service

Multinode. The name of a DDNS service provider. The name is one of the following: **`dnspark`**, **`dslreports`**, **`dyndns`**, **`easydns`**, **`namecheap`**, **`sitelutions`**, or **`zoneedit`**.

You can specify more than one DDNS provider for each interface by creating multiple **`service`** configuration nodes.

Configuration mode

```
service {
    dns {
        dynamic {
            interface interface {
                service service
            }
         }
    }
}
```

Use this command to specify the organizations that provides DDNS service to the AT&T Vyatta vRouter.

Use the `set` form of this command to specify a DDNS service provider.

Use the `delete` form of this command to remove a DDNS service provider.

Use the `show` form of this command to display information for a DDNS service provider.

service dns dynamic interface <interface> service <service> host-name <hostname>

Specifies the name of a host for which to update the DNS record of the dynamic DNS (DDNS) service provider.

Syntax:

```
set service dns dynamic interface interface service service host-name hostname
```

Syntax:

```
delete service dns dynamic interface interface service service host-name hostname
```

Syntax:

`show service dns dynamic interface` *interface* **`service`** *service* **`host-name`**

interface
: Multinode. An interface that supports DDNS.

service
: Multinode. The name of a DDNS service provider. The name is one of the following: **`dnspark`**, **`dslreports`**, **`dyndns`**, **`easydns`**, **`namecheap`**, **`sitelutions`**, or **`zoneedit`**.

hostname
: The name of a host.

Configuration mode

```
service {
 dns {
  dynamic {
   interface interface {
    service service {
     host-name hostname
    }
   }
  }
 }
}
```

Use this command to specify the name of a host for which to update the DNS record of the DDNS service provider.

Use the `set` form of this command to specify a host name.

Use the `delete` form of this command to remove the host name from the configuration.

Use the `show` form of this command to view host name configuration.

service dns dynamic interface <interface> service <service> login <service-login>

Specifies a login ID to use to log on to a dynamic DNS (DDNS) service provider.

Syntax:
`set service dns dynamic interface` *interface* **`service`** *service* **`login`** *service-login*

Syntax:
`delete service dns dynamic interface` *interface* **`service`** *service* **`login`**

Syntax:
`show service dns dynamic interface` *interface* **`service`** *service* **`login`**

interface
: Multinode. An interface that supports DDNS.

service
: Multinode. The name of a DDNS service provider. The name is one of the following: **`dnspark`**, **`dslreports`**, **`dyndns`**, **`easydns`**, **`namecheap`**, **`sitelutions`**, or **`zoneedit`**.

login
: A login ID.

Configuration mode

```
service {
 dns {
  dynamic {
   interface interface {
    service service {
     login service-login
    }
   }
  }
 }
}
```

Use this command to specify the login ID to use to log on to a DDNS service provider.

Use the `set` form of this command to specify the login ID for a DDNS service provider.

Use the `delete` form of this command to remove the login ID for a DDNS service provider.

Use the `show` form of this command to display the login ID for a DDNS service provider.

service dns dynamic interface <interface> service <service> password <service-password>

Specifies the password to use to log on to a dynamic DNS (DDNS) service provider.

Syntax:
`set service dns dynamic interface` *interface* **`service`** *service* **`password`** *service-password*

Syntax:
`delete service dns dynamic interface` *interface* **`service`** *service* **`password`**

Syntax:
`show service dns dynamic interface` *interface* **`service`** *service* **`password`**

interface
Multinode. An interface that supports DDNS.

service
Multinode. The name of a DDNS service provider. The name is one of the following: **`dnspark`**, **`dslreports`**, **`dyndns`**, **`easydns`**, **`namecheap`**, **`sitelutions`**, or **`zoneedit`**.

password
A password.

Configuration mode

```
service {
 dns {
  dynamic {
   interface interface {
    service service {
     password service-password
    }
   }
  }
 }
}
```

Use this command to specify the password to use to log on to a DDNS service provider.

Use the `set` form of this command to specify the password for a DDNS service provider.

Use the `delete` form of this command to remove the password for a DDNS service provider.

Use the `show` form of this command to display the password for a DDNS service provider.

service dns dynamic interface <interface> service <service> server <addr>

Specifies a server to which to send dynamic DNS (DDNS) updates.

Syntax:

```
set service dns dynamic interface interface service service server addr
```

Syntax:

```
delete service dns dynamic interface interface service service server
```

Syntax:

```
show service dns dynamic interface interface service service server
```

The default server of the DDNS service provider is used.

interface

Multinode. An interface that supports DDNS.

service

Multinode. The name of a DDNS service provider. The name is one of the following: **dnspark**, **dslreports**, **dyndns**, **easydns**, **namecheap**, **sitelutions**, or **zoneedit**.

addr

An IP address or a host name. Only some DDNS service providers require this address or host name.

Configuration mode

```
service {
 dns {
  dynamic {
   interface interface {
    service service {
     server addr
    }
   }
  }
 }
}
```

Use this command to specify the IP address or host name of the DDNS service provider's server that DDNS updates are sent to. This should be set only if the DDNS service provider requires it.

Use the `set` form of this command to specify the server to send DDNS updates to.

Use the `delete` form of this command to use the default DDNS service provider servers.

Use the `show` form of this command to view DDNS service provider server configuration.

service dns forwarding cache-size <size>

Specifies the size of the DNS forwarding service cache.

Syntax:

```
set service dns forwarding cache-size size
```

Syntax:

```
delete service dns forwarding cache-size
```

Syntax:

```
show service dns forwarding cache-size
```

A maximum of 150 DNS entries are stored in the DNS forwarding cache.

size

Optional. The maximum number of DNS entries to be held in the DNS forwarding cache. The number ranges 0 through10000, where 0 means an unlimited number of entries are stored. The default number is150.

Configuration mode

```
service {
 dns {
  forwarding {
   cache-size size
  }
 }
}
```

Use this command to specify the size of the DNS forwarding cache.

Use the `set` form of this command to specify the size of the DNS forwarding cache.

Use the `delete` form of this command to restore the default size of the DNS forwarding cache, which is 150 entries.

Use the `show` form of this command to display the size of the DNS forwarding cache.

service dns forwarding dhcp <interface>

Specifies an interface on which DHCP updates to name server information are received.

Syntax:
`set service dns forwarding dhcp` *interface*

Syntax:
`delete service dns forwarding dhcp` *interface*

Syntax:
`show service dns forwarding dhcp` *interface*

The system forwards DNS requests to all configured name servers and all name servers specified through DHCP.

interface

Multinode. An interface that is to receive name server information updates from a DHCP server.

Configuration mode

```
service {
 dns {
  forwarding {
   dhcp interface
  }
 }
}
```

Use this command to specify an interface that is to act as a DHCP client and receive updates to DNS name server information. The AT&T Vyatta vRouter uses this information to forward DNS requests from its local clients to the name server.

To be configured to listen for updates to name server information, the interface must be configured to obtain its own IP address through DHCP; that is, it must be configured as a DHCP client. Refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide for information about configuring the IP address of a physical interface.

By default, the DNS forwarding service creates a pool of name servers to which it forwards DNS requests; this pool comprises any name servers statically configured for the system (by using the `system name-server` command) and those of which it is notified through DHCP. This command overrides the default behavior: when an interface is specified by using this command, the system attends to DHCP name server information updates arriving on the specified interface.

This command can be combined with service dns forwarding name-server <ipv4> *(page 115)*, service dns forwarding system *(page 116)*, or both commands to provide a larger pool of candidate name servers.

Use the `set` form of this command to specify an interface to be used as the source of updates to the DHCP name server.

Use the `delete` form of this command to restore the default method of receiving updates to the name server, that is, the system forwards DNS requests to all configured name servers and all name servers specified through DHCP.

Use the `show` form of this command to view DNS forwarding DHCP update configuration.

service dns forwarding listen-on <interface>

Specifies an interface on which to listen for client DNS requests.

Syntax:
`set service dns forwarding listen-on` *interface*

Syntax:
`delete service dns forwarding listen-on` *interface*

Syntax:
`show service dns forwarding listen-on` *interface*

interface

Mandatory. Multinode. An interface on which to listen for client DNS requests.

You can specify more than one interface to receive client DNS requests by creating multiple **`listen-on`** configuration nodes.

Configuration mode

```
service {
 dns {
  forwarding {
   listen-on interface
  }
 }
}
```

Use this command to specify interfaces on which to listen for client DNS requests. Only queries received on interfaces specified with this command receive DNS answers. At least one interface must be specified for DNS forwarding to operate.

Use the `set` form of this command to specify an interface on which to listen for client DNS requests.

Use the `delete` form of this command to stop an interface from listening for client DNS requests.

Use the `show` form of this command to display the configuration of client DNS request listening.

service dns forwarding name-server <ipv4>

Specifies a name server to which to forward DNS requests.

Syntax:
`set service dns forwarding name-server` *ipv4*

Syntax:
`delete service dns forwarding name-server` *ipv4*

Syntax:
`show service dns forwarding name-server` *ipv4*

ipv4

Optional. Multinode. The IPv4 address of a name server to which to forward DNS requests.

You can forward DNS requests to more than one name server by creating multiple **`name-server`** configuration nodes.

Configuration mode

```
service {
 dns {
  forwarding {
   name-server ipv4
  }
 }
}
```

Use this command to specify a name server to which client DNS requests are forwarded.

Use of this command is optional. By default, the DNS forwarding service creates a default pool of name servers that comprises those statically configured by using the `system name-server` command plus those of which it was notified using DHCP. This command overrides the defaults: when this command is used, the system forwards DNS requests to the specified name server or servers.

This command can be combined with service dns forwarding dhcp <interface> *(page 114)*, service dns forwarding system *(page 116)*, or both commands to provide a larger pool of candidate name servers.

Use the `set` form of this command to specify a name server to which to forward DNS requests.

Use the `delete` form of this command to remove a name server from the list of name servers to which to forward DNS requests. If the last specified server is removed, the default forwarding behavior is restored.

Use the `show` form of this command to display the name servers to which DNS requests are forwarded.

service dns forwarding system

Specifies DNS forwarding to system-configured name servers.

Syntax:
`set service dns forwarding system`

Syntax:
`delete service dns forwarding system`

Syntax:
`show service dns forwarding`

Configuration mode

```
service {
 dns {
  forwarding {
   system
  }
 }
}
```

Use this command to direct the system to forward DNS requests to name servers that are statically configured by using the `system name-server` command.

By default, the DNS forwarding service forwards DNS requests to a pool of name servers that comprises the statically configured name servers plus those of which it was notified by using DHCP. This command overrides the defaults: when this command is used, DNS requests are forwarded to statically configured name servers.

This command can be combined with service dns forwarding dhcp <interface> *(page 114)*, service dns forwarding name-server <ipv4> *(page 115)*, or both commands to provide a larger pool of candidate name servers.

Use the `set` form of this command to specify DNS forwarding to system-configured name servers.

Use the `delete` form of this command to restore the default DNS forwarding behavior.

Use the `show` form of this command to display the configuration of DNS forwarding.

show dns dynamic status

Displays the update status for all hosts configured for dynamic DNS (DDNS) updates.

Syntax:

`show dns dynamic status`

Operational mode

By default, this command applies to the default routing table.

Use this command to display the update status for all host names that are configured for DDNS updates.

The following example shows how to display the update status for hosts that are configured for DDNS updates.

```
vyatta@R1> show dns dynamic status
show dns dynamic status
interface    : dp0p1p3
ip address   : 1.2.3.4
host-name    : test1.getmyip.com
last update  : Thu Sep 11 19:30:43 2008
update-status: good

interface    : dp0p1p3
ip address   : 1.2.3.5
host-name    : test2.getmyip.com
last update  : Thu Sep 11 19:30:43 2008
update-status: good

interface    : dp0p1p4
ip address   : 1.3.4.5
host-name    : test4
last update  : Thu Sep 11 19:34:16 2008
update-status: good
vyatta@R1>
```

show dns forwarding nameservers

Displays the name servers that are being used for DNS forwarding.

Syntax:

`show dns forwarding nameservers`

Operational mode

Use this command to display the name servers that are being used for DNS forwarding and those that are available but are not being used for DNS forwarding.

The following example shows how to display the name servers that are being used for DNS forwarding.

```
vyatta@R1> show dns forwarding nameservers
-----------------------------------------------
 Nameservers configured for DNS forwarding
-----------------------------------------------
10.0.0.30 available via 'system'
-----------------------------------------------
 Nameservers NOT configured for DNS forwarding
-----------------------------------------------
10.0.0.31 available via 'dhcp dp0p1p4'
vyatta@R1>
```

show dns forwarding statistics

Displays DNS forwarding statistics.

Syntax:

`show dns forwarding statistics`

Operational mode

Use this command to display statistics related to DNS forwarding. The statistics restart each time a change occurs in name servers from any source (DHCP, system, or statically configured), a change in static host mapping (by using the `system static-host-mapping` command), or a change made to the DNS forwarding configuration.

The following example shows how to display DNS forwarding statistics.

```
vyatta@R1> show dns forwarding statistics
----------------
Cache statistics
----------------
Cache size: 150
Queries forwarded: 5
Queries answered locally: 2
Total DNS entries inserted into cache: 23
DNS entries removed from cache before expiry: 0
---------------------
Nameserver statistics
---------------------
Server: 10.0.0.30
Queries sent: 5
Queries retried or failed: 0
vyatta@R1>
```

update dns dynamic interface <interface>

Sends a forced update to a dynamic DNS (DDNS) service provider on a specific interface.

Syntax:

`update dns dynamic interface` *text*

interface

An interface from which to send the forced update.

Operational mode

Use this command to manually initiate a forced update to a DDNS service provider. The forced update provides the DDNS service provider with the status of the specified interface.

Note that this command should be used sparingly because frequent unnecessary updates could cause the host name to be blocked by the DDNS service provider.

Flow Monitoring

Flow Monitoring overview

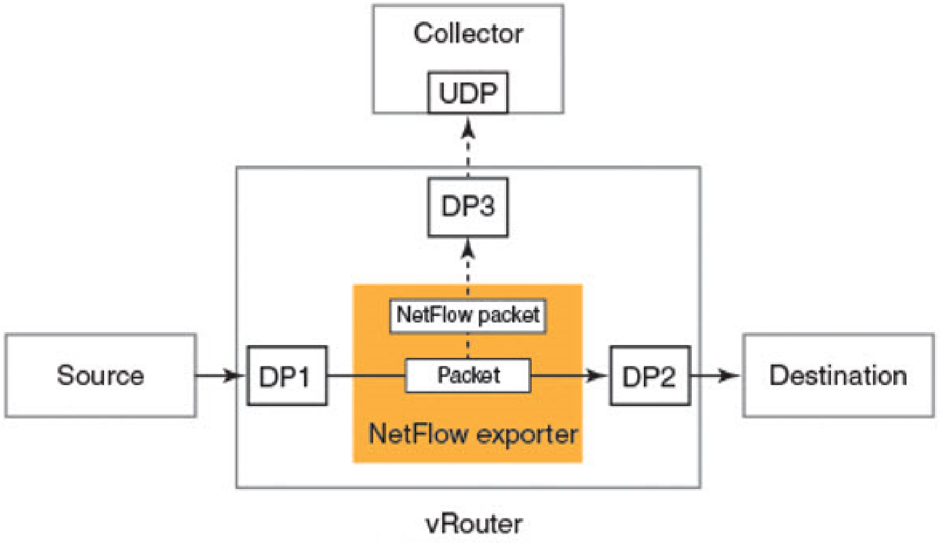
The Flow Monitoring service allows network administrators to collect IP flow information from an AT&T Vyatta vRouter.

AT&T Vyatta vRouters support the NetFlow Version 9 and IP Flow Information Export (IPFIX), which is based on the Internet Engineering Task Force (IETF) standard. This standard defines how IP flow information is formatted and transferred from an exporter (in this case, an AT&T Vyatta vRouter) to a NetFlow collector, a system that collects IPv4 and IPv6 flow information, as shown in the following figure.

The exporter periodically collects information about packets that flow through the router into a flow record. Then, the exporter packs the record into a UDP packet and sends it to the collector. For more information about the NetFlow Version 9, refer to RFC 3954 and for more information about the IPFIX, refer to RFC 7011 export format.

The user can also choose the choose the packet and information fields to aggregate and export.

Figure 10: AT&T Vyatta vRouter flow-monitoring architecture



The NetFlow Version 9 and IPFIX export format is based on templates, which allows for an extensible design of the record format. This means that future enhancements of the Flow Monitoring service do not require changes to the export protocol.

The following tables describe the data that is extracted by the Flow Monitoring service from the sampled packets and exported to the NetFlow collector.

Table 19: Exported NetFlow Version 9 and IPFIX IPv4 data

Field	Field ID
sourceIPv4Address	8
destinationIPv4Address	12
protocolIdentifier	4
ipClassOfService	5
sourceTransportPort	7

Field	Field ID
destinationTransportPort	11
ingressInterface	10
ipNextHopIPv4Address	15
tcpControlBits	6
egressInterface	14
flowDirection	61
packetDeltaCount	2
octetDeltaCount	1
flowStartMilliseconds	152
flowEndMilliseconds	153
sourceIPv4PrefixLength	9
destinationIPv4PrefixLength	13
bgpNextHopIPv4Address	18
bgpSourceAsNumber	16
bgpDestinationAsNumber	17
bgpPrevAdjacentAsNumber	129
bgpNextAdjacentAsNumber	128

Table 20: Exported NetFlow Version 9 and IPFIX IPv6 data

Field	Field ID
sourceIPv6Address	27
destinationIPv6Address	28
protocolIdentifier	4
ipClassOfService	5
sourceTransportPort	7
destinationTransportPort	11
ingressInterface	10
flowDirection	61
ipNextHopIPv6Address	62
tcpControlBits	6
egressInterface	14
packetDeltaCount	2
octetDeltaCount	1
flowStartMilliseconds	152
flowEndMilliseconds	153
sourceIPv6PrefixLength	29
destinationIPv6PrefixLength	30
bgpNextHopIPv6Address	63
bgpSourceAsNumber	16
bgpDestinationAsNumber	17
bgpPrevAdjacentAsNumber	129
bgpNextAdjacentAsNumber	128

Flow Monitoring configuration

The Flow Monitoring service consists of aggregators and exporters that accept flows and pass flows to other connected modules. Aggregator and exporter modules may be configured on an interface. The interface selectors generate defined fixed single-packet flows. For more information about the fields in these flows, refer to Exported NetFlow Version 9 and IPFIX IPv4 data *(page 120)*. Aggregator modules can connect with other aggregator and exporter modules by using a `next` statement. You can configure multiple such connections.

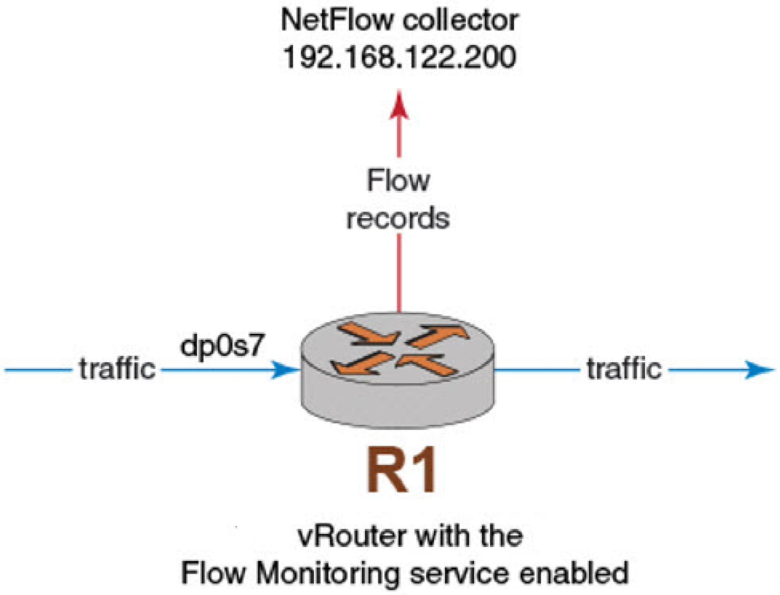
You can define a rule for the aggregator. Each rule is a list of key or nonkey fields. A key field identifies a unique entry in the cache functionality provided by the aggregator module. Nonkey fields are aggregated: for example, if packetDeltaCount is added as a nonkey field, the collected values are aggregated and summed before exporting the values to the flow report. An aggregator must have a rule, and a rule must have at least one key field.

Exporter modules do not pass flows to other modules.

> **Note:** The flexible key feature works only if the key and nonkey fields that are specified in an aggregator rule are present in flows entering that aggregator; otherwise, the flows are ignored. Therefore in the following example, the fields that are specified in the `foo` aggregator are a subset of the fields generated by the interface.

The following procedure shows how to configure the Flow Monitoring service on the R1 AT&T Vyatta vRouter, as shown in the example in the following figure.

Figure 11: AT&T Vyatta vRouter flow-monitoring configuration example



To configure the Flow Monitoring service on R1, perform the following steps in configuration mode.

Table 21: Configuring the flow-monitoring service

Step	Command

Create a flow-monitoring exporter and specify the IP address and UDP port of the NetFlow collector.	`set service flow-monitoring exporter foo udp-collector address 192.168.122.200`
	`vyatta@R1# set service flow-monitoring exporter foo udp-collector port 9995`

Step	Command
Create a packet selector named foo and specify a sampling size of 10 packets. The sampling rate in this case is 1:10, which means that the router randomly selects one packet from every 10 packets that flow through any data plane interface with which the selector is associated.	`vyatta@R1# set service flow-monitoring selector foo randomly out-of 10`

Create a packet aggregator named foo and specify an expiration time of 1,800 seconds for active flows and 15 seconds for inactive flows.	`vyatta@R1# set service flow-monitoring aggregator foo expiration inactive-timeout 15`
	`vyatta@R1# set service flow-monitoring aggregator foo expiration active-timeout 1800`

Connect the aggregator to the exporter.	`vyatta@R1# set service flow-monitoring aggregator foo next exporter foo`

Step	Command

Configure the aggregation rule for the foo aggregator .	`vyatta@R1# set service flow-monitoring aggregator foo key destinationIPv4Address`
	`vyatta@R1# set service flow-monitoring aggregator foo key destinationTransportPort`
	`vyatta@R1# set service flow-monitoring aggregator foo key ipClassOfService`
	`vyatta@R1# set service flow-monitoring aggregator foo key protocolIdentifier`
	`vyatta@R1# set service flow-monitoring aggregator foo key sourceIPv4Address`
	`vyatta@R1# set service flow-monitoring aggregator foo key sourceTransportPort`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key bgpDestinationAsNumber`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key bgpSourceAsNumber`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key destinationIPv4PrefixLength`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key egressInterface`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key flowEndMilliseconds`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key flowStartMilliseconds`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key ipNextHopIPv4Address`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key octetDeltaCount`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key packetDeltaCount`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key sourceIPv4PrefixLength`
	`vyatta@R1# set service flow-monitoring aggregator foo non-key tcpControlBits`

Step	Command

Apply the foo selector and foo aggregator to the dp0s7 data plane interface.	`vyatta@R1# set interfaces dataplane dp0s7 flow-monitoring selector foo`
	`vyatta@R1# set interfaces dataplane dp0s7 flow-monitoring aggregator foo`

Commit the configuration.	`vyatta@R1# commit`
Save the configuration.	`vyatta@R1# save`

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Verify the configuration.</td><td><pre>
vyatta@R1# show interfaces
interfaces {
   dataplane dp0s7 {
     address 10.10.1.1/24
     flow-monitoring {
          aggregator foo
          selector foo
            }
          }
       }

vyatta@R1# show services
service {
  flow-monitoring {
       exporter foo {
         udp-collector {
           address 192.168.122.200
           port 9995
                       }
                    }
            selector foo {
                 randomly {
                   out-of 10
                          }
                       }
             aggregator foo {
                 aggregator statistics: {
                    flows in cache: 18
                    expired flows: 180
                    expiration {
                       active-timeout 1800
                       inactive-timeout 15
                               }
                        next {
                                exporter foo
                             }
                        rule {
                           key
 destinationIPv4Address
                           key
 destinationTransportPort
                           key
 ingressInterface
                           key
 ipClassOfService
                           key
 protocolIdentifier
                           key
 sourceIPv4Address
                           key
 sourceTransportPort
                           non-key
 bgpDestinationAsNumber
                           non-key
 bgpSourceAsNumber
                           non-key
 destinationIPv4PrefixLength
                           non-key
 egressInterface
                           non-key
 flowEndMilliseconds
                           non-key
 flowStartMilliseconds
                           non-key
 ipNextHopIPv4Address
                           non-key
 octetDeltaCount
                           non-key
 packetDeltaCount
                           non-key
</pre></td></tr>
</table>

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Exit the configuration mode.</td><td><pre>vyatta@R1# exit</pre></td></tr>
<tr><td>Show exporter statistics.</td><td><pre>vyatta@R1:~$ show flow-monitoring
dataplane statistics:
    interface dp0s7:
        monitor default:
            packets observed: 884
            samples taken: 88

export daemon statistics:
  /exporter/default:
    /monitor/mon1/ipv4:
      reports received: 88
      reports exported: 88
    /monitor/mon1/ipv6:
      reports received: 0
      reports exported: 0</pre></td></tr>
</table>

Note: If a data plane interface fails to gather samples because of resource constraints, show flow-monitoring *(page 139)* displays the number of lost samples in the samples lost field (under the samples taken field) in the output.

Flow Monitoring Commands

interfaces dataplane <interface> flow-monitoring aggregator <aggregator-name>

Defines flow-monitoring aggregator on a data plane interface.

Syntax:
`set interfaces dataplane` *interface* **`flow-monitoring`** **`aggregator`** *aggregator-name*

Syntax:
`delete interfaces dataplane` *interface* **`flow-monitoring`** **`aggregator`** *aggregator-name*

Syntax:
`show interfaces dataplane` *interface* **`flow-monitoring`** **`aggregator`** *aggregator-name*

None

interface
Data plane interface. For more information about data plane interfaces, refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.

aggregator-name
Name of a flow-aggregator.

Configuration mode

```
interfaces {
    dataplane interface {
        flow-monitoring {
            aggregator aggregator-name
        }
    }
}
```

Use the `set` form of this command to define a flow-monitoring aggregator on a data plane interface.

Use the `delete` form of this command to remove a flow-monitoring aggregator from the data plane interface.

Use the `show` form of this command to display the name of flow-monitoring aggregator for a data plane interface.

interfaces dataplane <interface> flow-monitoring exporter <exporter-name>

Defines flow-monitoring exporter on a data plane interface.

Syntax:
`set interfaces dataplane` *interface* **`flow-monitoring`** **`exporter`** *exporter-name*

Syntax:
`delete interfaces dataplane` *interface* **`flow-monitoring`** **`exporter`** *exporter-name*

Syntax:
`show interfaces dataplane` *interface* **`flow-monitoring`** **`exporter`** *exporter-name*

None

interface

A data plane interface. For more information about data plane interfaces, refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.

exporter-name

Name of a flow exporter.

Configuration mode

```
interfaces {
    dataplane interface {
        flow-monitoring {
            exporter exporter-name
        }
    }
}
```

Use the `set` form of this command to define a flow-monitoring exporter on a data plane interface.

Use the `delete` form of this command to remove a configured flow-monitoring exporter from the data plane interface.

Use the `show` form of this command to display the name of a configured exporter for a data plane interface.

interfaces dataplane <interface> flow-monitoring selector <selector-name>

Associates a packet selector with a data plane interface through which the traffic to be monitored flows.

Syntax:

`set interfaces dataplane` *interface* **`flow-monitoring`** **`selector`** *selector-name*

Syntax:

`delete interfaces dataplane` *interface* **`flow-monitoring`** **`selector`** *selector-name*

Syntax:

`show interfaces dataplane` *interface* **`flow-monitoring`**

interface

A data plane interface. For more information about data plane interfaces, refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.

selector-name

The name of a packet selector.

Configuration mode

```
interfaces {
    dataplane interface {
        flow-monitoring {
            selector selector-name
        }
    }
}
```

Use the `set` form of this command to associate a packet selector with a data plane interface through which the traffic to be monitored flows.

Use the `delete` form of this command to disassociate a packet selector from a data plane interface.

Use the `show` form of this command to display the name of the configured selector for a data plane interface.

service flow-monitoring aggregator <aggregator-name> expiration <active-time-out | inactive-timeout>

Specifies the aggregator expiration of a Netflow collector.

Syntax:
`set service flow-monitoring aggregator` *aggregator-name* **`expiration`** { **`active-timeout`** *timeout-seconds* | **`inactive-timeout`** *timeout-seconds* }

Syntax:
`delete service flow-monitoring exporter` *exporter-name* **`expiration`**

Syntax:
`show service flow-monitoring exporter` *exporter-name*

The default timeout value is 0 seconds. The default values of 0 means that flows are expired immediately.

aggregator-name
Name of the flow-aggregator.

`active-timeout` ***timeout-seconds***
Expiration timeout in seconds for long-lasting flows.

`inactive-timeout` ***timeout-seconds***
Expiration timeout in seconds for idle or inactive flows.

> **Note:** Typically, you can use 15 for the inactive-timeout and 3600 for the active-timeout. Ensure that the active-timeout is greater than the inactive-timeout.

Configuration mode

```
service {
flow-monitoring {
  exporter aggregator-name{
          expiration {
             active-timeout timeout-seconds
            inactive-timeout timeout-seconds
            }
        }
    }
}
```

Use the `set` form of this command to specify the aggregator expiration to a Netflow collector.

Use the `delete` form of this command to remove the aggregator expiration from a NetFlow collector.

Use the `show` form of this command to display the aggregator expiration configuration of a NetFlow collector.

service flow-monitoring aggregator <aggregator-name> hashtable-bits

Specifies the length of hashtable used for aggregation in bits for the Netflow collector.

Syntax:
`set service flow-monitoring aggregator` *aggregator-name* **`hashtable-bits`** *bit-length*

Syntax:
`delete service flow-monitoring aggregator` *aggregator-name* **`hashtable-bits`**

Syntax:

`show service flow-monitoring aggregator` *aggregator-name*

The default hashtable-bits size is 17.

aggregator-name
The name of the flow-aggregator.

bit-length
Length of the hashtable used for aggregation in bits. The number ranges from 1 through 32.

Configuration mode

```
service {
flow-monitoring {
  exporter aggregator-name{
          hashtable-bits {
              bit-length
            }
        }
    }
}
```

Use the `set` form of this command to specify the length of the hashtable used for aggregation

Use the `delete` form of this command to remove the hashtable-bit length.

Use the `show` form of this command to display the aggregator configuration

service flow-monitoring aggregator <aggregator-name> next <aggregator | exporter>

Specifies the next flow-monitoring module for a Netflow collector.

Syntax:
`set service flow-monitoring aggregator` *aggregator-name* **`next`** { **`aggregator`** *aggregator-name* | **`exporter`** *exporter-name* }

Syntax:
`delete service flow-monitoring aggregator` *aggregator-name* **`next`** { **`aggregator`** *aggregator-name* | **`exporter`** *exporter-name* }

Syntax:
`show service flow-monitoring aggregator` *aggregator-name*

None

aggregator-name
Name of a flow-aggregator.

`aggregator` ***aggregator-name***
Name of an aggregator for the next flow-monitoring module.

`exporter` ***exporter-name***
Name of an exporter for the next flow-monitoring module.

Configuration mode

```
service {
flow-monitoring {
  exporter aggregator-name{
          next {
             aggregator aggregator-name
             exporter exporter-name
            }
        }
    }
}
```

Use the `set` form of this command to specify the next flow-monitoring module for a Netflow collector.

Use the `delete` form of this command to remove the next flow-monitoring module of a Netflow collector.

Use the `show` form of this command to display the aggregator of a Netflow collector.

service flow-monitoring aggregator <aggregator-name> rule <key | non-key>

Defines an aggregator rule of a Netflow collector.

Syntax:

```
set service flow-monitoring aggregator aggregator-name rule { key key-value | non-key key-value }
```

Syntax:

```
delete service flow-monitoring aggregator aggregator-name rule { key key-value | non-key tkey-value }
```

Syntax:

```
show service flow-monitoring aggregator aggregator-name
```

None

aggregator-name
Name of the flow-aggregator.

`key` ***key-value***
Specifies a key field on which to aggregate.

`non-key` ***key-value***
Specifies a nonkey field to collect

> **Note:** For more information about key values, refer to Exported NetFlow Version 9 and IPFIX (RFC 7011) tables for IPv4 and IPv6 data in Flow monitoring overview *(page 120)*.

Configuration mode

```
service {
flow-monitoring {
  exporter aggregator-name{
          rule {
             key key-field
             non-key key-field
            }
        }
    }
}
```

Use the `set` form of this command to specify the aggregator rule of a Netflow collector.

Use the `delete` form of this command to remove the aggregator rule from a NetFlow collector.

Use the `show` form of this command to display the aggregator rule of a Netflow collector.

service flow-monitoring exporter <exporter-name> max-record-rate

Specifies the maximum number of flow records that are sent each second to a Netflow collector.

Syntax:

```
set service flow-monitoring exporter exporter-name max-record-rate record-per-second
```

Syntax:

delete service flow-monitoring exporter *exporter-name* **max-record-rate**

Syntax:
show service flow-monitoring exporter *exporter-name*

The default number of records is zero, which means that there is no limit and as many records as are available are sent without delay.

exporter-name
Name of the flow-exporter.

record-per-second
Maximum number of flow records that are sent each second to the Netflow collector. The number ranges from 0 through 4294967295.

Configuration mode

```
service {
flow-monitoring {
  exporter exporter-name{
          max-record-rate {
             record-per-second
            }
        }
    }
}
```

Use the `set` form of this command to specify the maximum number of flow records that are sent each second to a Netflow collector.

Use the `delete` form of this command to remove the number of flow records that are sent each second to a NetFlow collector.

Use the `show` form of this command to display the number of flow records that are sent each second to a NetFlow collector.

service flow-monitoring exporter <exporter-name> protocol-version NFV9 | IPFIX

Specifies the protocol version of the Netflow collector.

Syntax:
set service flow-monitoring exporter *exporter-name* **protocol-version** { **NFV9** | **IPFIX** }

Syntax:
delete service flow-monitoring exporter *exporter-name* **protocol-version** { **NFV9** | **IPFIX** }

Syntax:
show service flow-monitoring exporter *exporter-name*

The default protocol version is NFV9.

exporter-name
Name of the flow-exporter.

IPFIX
Specifies IPFIX flow-monitoring export control.

NFV9
Specifies NFV9 flow-monitoring export control.

Configuration mode

```
service {
flow-monitoring {
  exporter exporter-name{
          protocol-version {
             IPFIX
             NFv9
            }
        }
    }
}
```

Use the `set` form of this command to specify the protocol version of a NetFlow collector.

Use the `delete` form of this command to remove the protocol version from a NetFlow collector.

Use the `show` form of this command to display the configured protocol version of a NetFlow collector.

service flow-monitoring exporter <exporter-name> template-refresh-interval

Specifies the interval for sending the templates periodically.

Syntax:
`set service flow-monitoring exporter` *exporter-name* **`template-refresh-interval`** *interval*

Syntax:
`delete service flow-monitoring exporter` *exporter-name* **`template-refresh-interval`** *interval*

Syntax:
`show service flow-monitoring exporter` *exporter-name*

The default template-refresh-interval is 20 seconds.

exporter-name
The name of the flow-exporter.

interval
Interval time for sending the templates periodically. The time interval ranges from 1 through 3600 seconds

Configuration mode

```
service {
flow-monitoring {
  exporter exporter-name{
          template-refresh-interval {
             interval
            }
        }
    }
}
```

Use the `set` form of this command to specify the interval for sending the templates periodically.

Use the `delete` form of this command to remove the protocol-version of the NetFlow collector.

Use the `show` form of this command to display the configured interval for sending the templates periodically.

service flow-monitoring exporter <exporter-name> udp-collector address <ip-address>

Specifies the IPv4 or IPv6 address of the NetFlow collector.

Syntax:
`set service flow-monitoring` **`exporter`** *exporter-name* **`udp-collector address`** *ip-address*

Syntax:

`delete service flow-monitoring` **`exporter`** *exporter-name* **`udp-collector address`** *ip-address*

Syntax:
`show service flow-monitoring` **`exporter`** *exporter-name* **`udp-collector`**

exporter-name
Specifies the name of the exporter.
ip-address
IPv4 or IPv6 address of the collector.

Configuration mode

```
service {
    flow-monitoring {
        exporter exporter-name {
            udp-collector {
                address ip-address
            }
        }
    }
}
```

Use the `set` form of this command to specify the IPv4 or IPv6 address of the NetFlow collector.

Use the `delete` form of this command to remove the IP address form the NetFlow collector.

Use the `show` form of this command to display the configured IP address of the NetFlow collector.

service flow-monitoring exporter <exporter-name> udp-collector mtu <udp-mtu>

Specifies the UDP MTU size of a Netflow collector.

Syntax:
`set service flow-monitoring exporter` *exporter-name* **`udp-collector mtu`** *udp-mtu*

Syntax:
`delete service flow-monitoring exporter` *exporter-name* **`udp-collector mtu`** *udp-mtu*

Syntax:
`show service flow-monitoring exporter` *exporter-name*

The default size is 1400.

exporter-name
Name of the flow exporter.
udp-mtu
UDP MTU size of the collector. The size ranges from 92 through 65535.

Configuration mode

```
service {
flow-monitoring {
  exporter exporter-name{
          udp-collector {
             mtu udp-mtu
            }
        }
    }
}
```

Use the `set` form of this command to specify the UDP MTU size of a NetFlow collector.

Use the `delete` form of this command to specify the default MTU size, which is 1400.

Use the `show` form of this command to display the UDP MTU size of the NetFlow collector.

service flow-monitoring exporter <exporter-name> udp-collector port <udp-port>

Specifies the UDP port for the NetFlow collector.

Syntax:
`set service flow-monitoring exporter` *exporter-name* **`udp-collector port`** *udp-port*

Syntax:
`delete service flow-monitoring exporter` *exporter-name* **`udp-collector port`** *udp-port*

Syntax:
`show service flow-monitoring exporter` *exporter-name*

None

exporter-name
Specifies the name of the exporter.

udp-port
UDP port number of the collector. The number ranges from 0 through 65535.

Configuration mode

```
service {
    flow-monitoring {
        exporter exporter-name {
            udp-collector {
                port udp-port
            }
        }
    }
}
```

Use the `set` form of this command to specify a UDP port of the NetFlow collector.

Use the `delete` form of this command to delete the UDP port number.

Use the `show` form of this command to display the UDP port for the NetFlow collector.

service flow-monitoring exporter <exporter-name> udp-collector routing-instance <routing-instance>

Specifies the routing instance to use to export flows to a collector.

Syntax:
`set service flow-monitoring` **`exporter`** *exporter-name* **`udp-collector routing-instance`** *routing-instance*

Syntax:
`delete service flow-monitoring` **`exporter`** *exporter-name* **`udp-collector routing-instance`** *routing-instance*

Syntax:
`show service flow-monitoring` **`exporter`** *exporter-name* **`udp-collector`**

exporter-name

Specifies the name of the exporter.

routing-instance

Name of a VRF routing instance.

Configuration mode

```
service {
    flow-monitoring {
        exporter exporter-name {
            udp-collector {
                routing-instance routing-instance
            }
        }
    }
}
```

Use the `set` form of this command to specify the VRF routing instance for exporting flows to a collector.

Use the `delete` form of this command to remove the VRF routing instance for exporting flows to a collector.

Use the `show` form of this command to display the VRF routing instance for exporting flows to a collector.

service flow-monitoring selector <selector-name> direction

Specifies the direction for Netflow monitoring.

Syntax:

`set service flow-monitoring selector` *selector-name* **`direction`** { **`egress`** | **`ingress`** | **`both`** }

Syntax:

`delete service flow-monitoring selector` *selector-name* **`direction`** { **`egress`** | **`ingress`** | **`both`** }

Syntax:

`delete service flow-monitoring selector` *selector-name* **`direction`**

Ingress direction

selector-name

The name of the packet selector.

Configuration mode

```
service {
    flow-monitoring {
        selector selector-name {
            direction {
                ingress
                egress
                both
            }
        }
    }
}
```

Use the `set` form of this command to specify the direction for Netflow monitoring (ingress, egress, or both).

Use the `delete` form of this command to delete the direction setting for Netflow monitoring.

Use the `show` form of this command to display the direction setting for Netflow monitoring.

service flow-monitoring selector <selector-name> randomly out-of <num-of-packets>

Creates a random-packet selector and specifies the size of the packet sample window from which to select a packet.

Syntax:

```
set service flow-monitoring selector selector-name randomly out-of num-of-packets
```

Syntax:

```
delete service flow-monitoring selector selector-name randomly out-of num-of-packets
```

Syntax:

```
show service flow-monitoring selector
```

The default size of the packet sample is 1000.

selector-name
The name of the packet selector.

num-of-packets
The size of the packet sample window from which to select a packet. The size ranges from 10 through 10000.

Configuration mode

```
service {
    flow-monitoring {
        selector selector-name {
            randomly {
                out-of num-of-packets
            }
        }
    }
}
```

This command randomly samples packets. For example, for a 1:100 sampling, the AT&T Vyatta vRouter randomly calculates a number from 1 through 100, say 63. When the sixty-third packet arrives at the data plane interface that is being monitored, the router extracts the NetFlow packet details and sends them to the NetFlow collector. The router ignores the remaining 37 packets of the sampling window (packets 64 through 100), then the router repeats the process of sampling packets.

Use the `set` form of this command to create a random-packet selector and specify the sample size.

Use the `delete` form of this command to delete the random-packet selector.

Use the `show` form of this command to display the details of the configured random-packet selector.

clear flow-monitoring

Clears the flow-monitoring statistics.

Syntax:

```
clear flow-monitoring
```

Operational mode

Use this command to clear the flow-monitoring statistics that have been gathered by the Flow Monitoring service.

show flow-monitoring

Displays the flow-monitoring statistics.

Syntax:

```
show flow-monitoring
```

Operational mode

Use this command to display the flow-monitoring statistics that have been gathered by the Flow Monitoring service.

The following example shows how to display flow-monitoring configuration information and usage statistics.

```
vyatta@vyatta:~$ show flow-monitoring
dataplane statistics:
    interface dp0s3:
        monitor default:
            packets observed:            128352
            samples taken:                 1283
    interface dp0s4:
        monitor default:
            packets observed:              2243
            samples taken:                  224

aggregator statistics:
    aggregator foo:
        flows in cache:                      18
        expired flows:                      180
    aggregator bar:
        flows in cache:                       6
        expired flows:                       46

exporter statistics:
    exporter alice:
        samples exported:                  1097
        flows exported:                     180
        flow packets sent:                  180
    exporter bob:
        samples exported:                   192
        flows exported:                      46
        flow packets sent:                   46
    exporter fred:
        samples exported:                  1289
        flows exported:                     226
        flow packets sent:                  180
```

LLDP

LLDP overview

LLDP is an open standard for network devices to communicate link-layer topology and connection endpoint information on IEEE 802 (Ethernet) LANs and MANs. LLDP is described in the IEEE standards document 802.1AB, *Station and Media Access Control Connectivity Discovery*. It allows a station on the network to advertise information about its capabilities, configuration, and identity to other LLDP-enabled stations on the same physical network. This information is stored in the device as a standard management information base (MIB) as specified in RFC 2922. A network management system can query these MIBs using SNMP to model the topology of the network.

Configuring LLDP

To enable LLDP on an AT&T Vyatta vRouter, you must enable the service by setting its configuration node, as in the following example.

Table 22: Enabling LLDP on a system

Step	Command
Create the LLDP service configuration node.	`vyatta@vyatta# set service lldp`
Commit the configuration.	`vyatta@vyatta# commit`

After the service is enabled, you can record information about the location, management address, and port of the device and the legacy protocols it supports. Additional information, including the configured capabilities and neighbors of the system, is extracted automatically from the system and stored in a MIB.

The following example shows how to configure a civic-based location for the system, as follows:

```
Suite 200 - 1301 Shoreway Road
Belmont, CA, USA
94002-4157
```

Table 23: Configuring a civic-based location for LLDP

Step	Command
Configure the language.	`vyatta@vyatta#set service lldp interface dp0p1p1 location civic-based ca-type 0 ca-value English`
Configure the occupant.	`vyatta@vyatta#set service lldp interface dp0p1p1 location civic-based ca-type 23 ca-value "Vyatta, Inc. Corporate Headquarters"`
Configure the suite number.	`vyatta@vyatta#set service lldp interface dp0p1p1 location civic-based ca-type 26 ca-value “Suite 200”`

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Configure the floor number.</td><td><pre>vyatta@vyatta#set service lldp interface dp0p1p1 location civic-based ca-type 27 ca-value 2nd</pre></td></tr>
<tr><td>Configure the street address.</td><td><pre>vyatta@vyatta#set service lldp interface dp0p1p1 location civic-based ca-type 6 ca-value “1301 Shoreway Road”</pre></td></tr>
<tr><td>Configure the city.</td><td><pre>vyatta@vyatta#set service lldp interface dp0p1p1 location civic-based ca-type 3 ca-value Belmont</pre></td></tr>
<tr><td>Configure the country.</td><td><pre>vyatta@vyatta#set service lldp interface dp0p1p1 location country-code US</pre></td></tr>
<tr><td>Configure the ZIP code.</td><td><pre>vyatta@vyatta#set service lldp interface dp0p1p1 location civic-based ca-type 24 ca-value 94002-4157</pre></td></tr>
<tr><td>Commit the configuration.</td><td><pre>vyatta@vyatta#commit</pre></td></tr>
<tr><td>Show the LLDP configuration.</td><td><pre>vyatta@vyatta#show service lldp

 interface dp0p1p1 {
    location {
        civic-based {
            ca-type 0 {
                ca-value English
            }
            ca-type 3 {
                ca-value Belmont
            }
            ca-type 6 {
                ca-value “1301 Shoreway Road”
            }
            ca-type 23 {
                ca-value "Vyatta, Inc. Corporate Headquarters"
            }
            ca-type 24 {
                ca-value 94002-4157
            }
            ca-type 26 {
                ca-value "Suite 200"
            }
            ca-type 27 {
                ca-value 2nd
            }
            country-code US
        }
    }
 }</pre></td></tr>
</table>

The location can be coordinate-based rather than civic-based, as shown in the following example.

Table 24: Configuring the physical coordinates of a system

Step	Command

Configure the latitude coordinate.	`vyatta@vyatta#set service lldp interface`
	`dp0p1p4 location coordinate-based latitude`
	`37.524449N`

Configure the longitude coordinate.	`vyatta@vyatta#set service lldp interface`
	`dp0p1p4 location coordinate-based longitude`
	`122.267255W`

Commit the configuration.	`vyatta@vyatta#commit`

Show the LLDP configuration for the dp0p1p4 interface.	`vyatta@vyatta#show service lldp interface`
	`dp0p1p4`
	`location {`
	`   coordinate-based {`
	`       latitude 37.524449N`
	`       longitude 122.267255W`
	`   }`
	`}`

Displaying LLDP information

When the system is enabled for LLDP, it can gather and display information about link-layer neighbors, as shown below.

Showing LLDP neighbors

```
vyatta@vyatta:~$show lldp neighbors

Capability Codes: R - Router, B - Bridge, W - Wlan r - Repeater, S - Station
                  D - Docsis, T - Telephone, O - Other

Device ID        Local     Proto  Cap   Platform             Port ID
---------        -----     -----  ---   --------             -------
R1               dp0p160p1 LLDP   RS    Vyatta Router      dp0p160p1
R2               dp0p160p1 LLDP   RS    Vyatta Router      dp0p160p1
HNF-BFD2         dp0p160p1 LLDP   RS    Vyatta Router      dp0p160p1
HNF-BF           dp0p192p1 LLDP   RS    Vyatta Router      dp0p192p1
HNF-BFD2         dp0p224p1 LLDP   RS    Vyatta Router      dp0p224p1
```

The following example shows detailed information on LLDP neighbors.

Showing detailed information on LLDP neighbors

```
vyatta@vyatta:~$show lldp neighbors detail

-------------------------------------------------------------------------------
LLDP neighbors:
-------------------------------------------------------------------------------
Interface:    dp0p160p1, via: LLDP, RID: 1, Time: 0 day, 01:12:52
  Chassis:
         ChassisID:    mac 00:0c:29:e6:b1:7d
         SysName:      R1
         SysDescr:     Vyatta Router running on AT&T 5600 vRouter 3.5 R5
         MgmtIP:       10.37.108.1
         MgmtIP:       3000:1:2::1
         Capability:   Bridge, off
         Capability:   Router, on
         Capability:   Wlan, off
         Capability:   Station, on
  Port:
         PortID:       mac 00:0c:29:e6:b1:7d
         PortDescr:    dp0p160p1
         PMD autoneg:  supported: no, enabled: yes
           MAU oper type: 10GigBaseR - R PCS/PMA, unknown PMD.
  LLDP-MED:
        Device Type:  Network Connectivity Device
        Capability:   Capabilities
        Capability:   Policy
        Capability:   Location
        Capability:   MDI/PSE
        Capability:   MDI/PD
        Capability:   Inventory
        Inventory:
          Hardware Revision: None
          Software Revision: 3.14.48-1-amd64-vyatta
          Firmware Revision: 6.00
          Serial Number: VMware-56 4d dd 2d b2 65 00 db-1
          Manufacturer: VMware, Inc.
          Model:        VMware Virtual Platform
          Asset ID:     No Asset Tag
--------------------------------------------------------------------------------
Interface:    dp0p160p1, via: LLDP, RID: 2, Time: 0 day, 01:12:52
  Chassis:
    ChassisID:    mac 00:0c:29:44:dc:65
    SysName:      R2
    SysDescr:     Vyatta Router running on AT&T 5600 vRouter 3.5 R5
    MgmtIP:       10.37.108.3
    MgmtIP:       3000:2:4::1
    Capability:   Bridge, off
    Capability:   Router, on
    Capability:   Wlan, off
    Capability:   Station, on
  Port:
    PortID:       mac 00:0c:29:44:dc:65
    PortDescr:    dp0p160p1
    PMD autoneg:  supported: no, enabled: yes
      MAU oper type: 10GigBaseR - R PCS/PMA, unknown PMD.
  LLDP-MED:
    Device Type:  Network Connectivity Device
    Capability:   Capabilities
    Capability:   Policy
    Capability:   Location
    Capability:   MDI/PSE
    Capability:   MDI/PD
    Capability:   Inventory
  Inventory:
    Hardware Revision: None
    Software Revision: 3.14.48-1-amd64-vyatta
    Firmware Revision: 6.00
    Serial Number: VMware-56 4d 92 ca 84 63 bc f2-a
    Manufacturer: VMware, Inc.
    Model:        VMware Virtual Platform
    Asset ID:     No Asset Tag
--------------------------------------------------------------------------------
```

LLDP Commands

service lldp

Enables the LLDP service.

Syntax:
`set service lldp`

Syntax:
`delete service lldp`

Syntax:
`show service lldp`

Configuration mode

```
service {
    lldp {}
}
```

Use this command to enable LLDP on the system.

Use the `set` form of this command to enable the LLDP service.

Use the `delete` form of this command to disable the LLDP service.

Use the `show` form of this command to display configuration of the LLDP service.

service lldp interface <interface> location civic-based

Records a civic-based location for an LLDP-enabled device.

Syntax:
`set service lldp interface` *interface* **`location civic-based`** [**`ca-type`** *type* **`ca-value`** *value* | **`country-code`** *code*]

Syntax:
`delete service lldp interface` *interface* **`location civic-based ca-type`** *type* **`ca-value`**

Syntax:
`show service lldp interface` *interface* **`location civic-based ca-type`** *type* **`ca-value`**

interface
An interface to which the configured information applies. The name of any IEEE 802.1-compatible interface or the **`all`** keyword, where **`all`** refers to all 802.1-compatible interfaces.

`ca-type` ***type***
Multinode. Records a civic address type as defined in the ANSI document. The type is one of the following:

- 0 — Language
- 1 — National subdivisions
- 2 — County, parish, district
- 3 — City, township
- 4 — City division, borough, ward

- 5 — Neighborhood, block
- 6 — Street
- 16 — Leading street direction
- 17 — Trailing street suffix
- 18 — Street suffix
- 19 — House number
- 20 — House number suffix
- 21 — Landmark or vanity address
- 22 — Additional location info
- 23 — Name
- 24 — Postal or ZIP code
- 25 — Building
- 26 — Unit
- 27 — Floor
- 28 — Room number
- 29 — Place type
- 128 — Script

You can record multiple civic address components by creating multiple **`ca-type`** and **`ca-value`** pairs, where each pair represents a different component.

`ca-value` ***value***

The value for the specified civic address type.

`country-code` ***code***

A two-letter code, as defined in ISO 3166, representing the country in which the device is located.

Configuration mode

```
service {
    lldp {
        interface interface {
            location {
            civic-based {
                ca-type type {
                    ca-value value
                }
                country-code code
            }
        }
    }
}
```

Use this command to record the components of a civic address identifying the location of the device. A civic address-based location requires a country code and at least one **`ca-type`** and **`ca-value`** pair.

Use the `set` form of this command to specify a component of a civic address.

Use the `delete` form of this command to remove civic address configuration.

Use the `show` form of this command to show civic address component configuration.

service lldp interface <interface> location coordinate-based

Records a coordinate-based location for an LLDP-enabled device.

Syntax:

`set service lldp interface` *interface* **`location`** **`coordinate-based`** [**`altitude`** *altitude* | **`datum`** *datum* | **`latitude`** *latitude* | **`longitude`** *longitude*]

Syntax:

delete service lldp interface *interface* **location** **coordinate-based** [**altitude** | **datum** | **latitude** | **longitude**]

Syntax:
show service lldp interface *interface* **location** **coordinate-based** [**altitude** | **datum** | **latitude** | **longitude**]

No location is configured.

interface

An interface to which the configured information applies. The name of any IEEE 802.1-compatible interface can be specified or the **all** keyword, where **all** refers to all 802.1-compatible interfaces.

altitude ***altitude***

Specifies the altitude, in meters, of the device. The default altitude is 0.

datum ***datum***

Specifies the reference datum for the coordinate system. The datum is **WGS84**, **NAD83**, or **MLLW**. The default datum is **WGS84**.

latitude ***latitude***

Specifies the latitude of the device. The format for the latitude is *deg.minD*. where *deg* is degrees, *min* is minutes to any level of precision, and *D* is N or S, representing North or South, respectively; for example 37.524449N.

longitude ***longitude***

The longitude of the device. The format is *deg.minD* . where *deg* is degrees, *min* is minutes to any level of precision, and *D* is E or W, representing East or West, respectively; for example, 122.267255W.

Configuration mode

```
service {
    lldp {
        interface interface {
            location {
                coordinate-based {
                    altitude altitude
                    datum datum
                    latitude latitude
                    longitude longitude
                }
            }
        }
    }
}
```

Use this command to specify a location for a device based on its coordinates.

When a coordinate-based location is used, both the latitude and longitude must be configured; other values are optional.

Use the `set` form of this command to specify a coordinate-based location.

Use the `delete` form of this command to remove a coordinate-based location and restore any default values.

Use the `show` form of this command to show coordinate-based configuration.

service lldp interface <interface> location elin <phone-num>

Records an emergency line identification number (ELIN) for an LLDP-enabled device.

Syntax:
set service lldp interface *interface* **location** **elin** *phone-num*

Syntax:
delete service lldp interface *interface* **location** **elin**

Syntax:
`show service lldp interface` *interface* **`location elin`**

interface

An interface to which the configured information applies. The name of any IEEE 802.1-compatible interface or the **`all`** keyword, where **`all`** refers to all 802.1-compatible interfaces.

phone-num

An emergency line identification number. The number is a 10-to-25-digit phone number. A phone number with fewer than 10 digits must be padded with zeros; for example, 911 must be represented as 0000000911.

Configuration mode

```
service {
    lldp {
        interface interface {
            location {
                elin phone-num
            }
        }
    }
}
```

Use this command to specify an emergency call service ELIN.

Use the `set` form of this command to record an ELIN.

Use the `delete` form of this command to remove the ELIN.

Use the `show` form of this command to display the ELIN.

service lldp legacy-protocols <protocol>

Specifies which legacy (proprietary) link-layer discovery protocols to support in addition to LLDP.

Syntax:
`set service lldp legacy-protocol` *protocol*

Syntax:
`delete service lldp legacy-protocol` *protocol*

Syntax:
`show service lldp legacy-protocol`

protocol

Multinode. A proprietary link-layer discovery protocol. The protocol is one of the following:

`cdp` —Cisco Discovery Protocol

`edp` —Extreme Discovery Protocol

`fdp` —Foundry Discovery Protocol

`sonmp` —Nortel Discovery Protocol

You can enable support for multiple legacy protocols by creating multiple legacy-protocol configuration nodes.

Configuration mode

```
service {
    lldp {
        legacy-protocol protocol
    }
}
```

Use this command to specify a legacy (proprietary) link-layer discovery protocol to support in addition to LLDP.

If a frame from one of the specified protocols is received on an interface, the system begins to send frames for that protocol on the interface.

Use the `set` form of this command to enable support for a legacy protocol.

Use the `delete` form of this command to disable support for a legacy protocol.

Use the `show` form of this command to show legacy protocol configuration.

service lldp management-address <ipv4>

Records the management address of the system.

Syntax:
`set service lldp management-address` *ipv4*

Syntax:
`delete service lldp management-address`

Syntax:
`show service lldp management-address`

The system automatically determines which address to advertise as the management address.

ipv4
The IP address of the management system. The address must be an IPv4 address.

Configuration mode

```
service {
    lldp {
        management-address ipv4
    }
}
```

Use this command to specify the IP address to be advertised as the management address by LLDP.

Use the `set` form of this command to set the management address.

Use the `delete` form of this command to restore the default behavior, that is, the system automatically determines which address to advertise as the management address.

Use the `show` form of this command to show LLDP management address configuration.

show lldp neighbors

Displays a summary of link layer neighbors that are running LLDP.

Syntax:
`show lldp neighbors [ detail ]`

When used with no option, this command displays a summary of information about link layer neighbors.

`detail`
Optional. Displays detailed information about link layer neighbors.

Operational mode

Use this command to display information about link layer neighbors that are running LLDP.

If support for any legacy link-layer discovery protocol has been enabled (using service lldp legacy-protocols <protocol> *(page 147)*), the system also displays neighbors discovered using that protocol.

The following example shows how to display a summary of link layer neighbors.

```
vyatta@vyatta:~$show lldp neighbors

Capability Codes: R - Router, B - Bridge, W - Wlan r - Repeater, S - Station
                  D - Docsis, T - Telephone, O - Other

Device ID        Local     Proto  Cap   Platform             Port ID
---------        -----     -----  ---   --------             -------
R1               dp0p160p1 LLDP   RS    Vyatta Router      dp0p160p1
R2               dp0p160p1 LLDP   RS    Vyatta Router      dp0p160p1
HNF-BFD2         dp0p160p1 LLDP   RS    Vyatta Router      dp0p160p1
HNF-BF           dp0p192p1 LLDP   RS    Vyatta Router      dp0p192p1
HNF-BFD2         dp0p224p1 LLDP   RS    Vyatta Router      dp0p224p1
```

The following example shows how to display details of link layer neighbors.

```
vyatta@vyatta:~$show lldp neighbors detail
-------------------------------------------------------------------------------
LLDP neighbors:
-------------------------------------------------------------------------------
Interface:    dp0p1p1, via: CDPv1, RID: 3, Time: 0 day, 00:19:34
  Chassis:
    ChassisID:    local medusa
    SysName:      medusa
    SysDescr:     cisco 2511 running on
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-I-L), Version 12.0(14), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2000 by cisco Systems, Inc.
Compiled Tue 31-Oct-00 23:59 by linda
    MgmtIP:       10.1.0.9
    Capability:   Router, on
  Port:
    PortID:       ifname Ethernet0
    PortDescr:    Ethernet0
-------------------------------------------------------------------------------
Interface:    dp0p1p1, via: LLDP, RID: 4, Time: 0 day, 00:19:28
  Chassis:
    ChassisID:    mac 00:1b:21:44:70:44
    SysName:      tethys
    SysDescr:     Vyatta Series 3500 running on 999.larkspurse.04270036
    MgmtIP:       10.1.0.40
    Capability:   Bridge, off
    Capability:   Router, on
    Capability:   Wlan, off
  Port:
    PortID:       mac 00:24:e8:7b:ca:6c
    PortDescr:    dp0p1p1
    PMD autoneg:  supported: yes, enabled: yes
      Adv:          10Base-T, HD: yes, FD: yes
      Adv:          100Base-T, HD: yes, FD: yes
      Adv:          1000Base-T, HD: no, FD: yes
      MAU oper type: 100BaseTXFD - 2 pair category 5 UTP, full duplex mode
  LLDP-MED:
    Device Type:  Network Connectivity Device
    Capability:   Capabilities
    Capability:   Location
    Capability:   Inventory
    LLDP-MED Location Identification: Type: elin
      ECS ELIN:     0000000911
    Inventory:
      Software Revision: 2.6.32-1-586-vyatta
      Firmware Revision: 2.0.11
      Serial Number: JGSM3K1
      Manufacturer: Vyatta
      Model:        Series 3500
-------------------------------------------------------------------------------
Interface:    dp0p1p4, via: LLDP, RID: 6, Time: 0 day, 00:00:03
  Chassis:
    ChassisID:    mac 00:0c:29:8c:53:7c
    SysName:      R1
    SysDescr:     Vyatta Router running on Vyatta Subscription Edition 6.0 2010.03.22
    MgmtIP:       20.0.0.2
    Capability:   Bridge, off
    Capability:   Router, on
    Capability:   Wlan, off
  Port:
    PortID:       ifname dp0p1p1
    PortDescr:    bridge
    PMD autoneg:  supported: yes, enabled: yes
      Adv:          10Base-T, HD: yes, FD: yes
      Adv:          100Base-T, HD: yes, FD: yes
      Adv:          1000Base-T, HD: no, FD: yes
      MAU oper type: 1000BaseTFD - Four-pair Category 5 UTP, full duplex mode
  LLDP-MED:
    Device Type:  Network Connectivity Device
    Capability:   Capabilities
    Capability:   Location
    Capability:   Inventory
    LLDP-MED Location Identification: Type: address
      Country:      US
      Language:     English
      City, township: Belmont
      Street:       1301 Shoreway Road
      Name:         Vyatta, Inc. Corporate Headquarters
      Postal/ZIP code: 94002-4157
      Floor:        2nd
      Room number:  Suite 200
    Inventory:
      Hardware Revision: None
      Software Revision: 2.6.32-1-586-vyatta-virt
      Firmware Revision: 6.00
      Serial Number: VMware-56 4d 6b 88 64 cc 44 27-2
      Manufacturer: VMware, Inc.
      Model:        VMware Virtual Platform
      Asset ID:     No Asset Tag

----------------------------------------------------------------
```

NHRP

NHRP overview

Next Hop Resolution Protocol (NHRP) is a software-addressing service commonly used in nonbroadcast multiaccess (NBMA) networks. In an NBMA network, NHRP provides the mapping between the NBMA next hop and the Layer 3 subnetwork address.

NHRP and NBMA networks

An NBMA network connects multiple hosts, but has no broadcast or multicast capability; data is transmitted only from one device to one other device. For full connectivity in such a network, connections must be meshed in some way.

A full mesh of connections is difficult to scale, so NBMA networks often employ some version of a hub-and-spoke network to reduce the complexity of the network. However, hub-and-spoke networks have challenges of their own.

- The hub becomes a single point of failure for the network.
- All network traffic passes through the hub, which becomes a processing bottleneck.

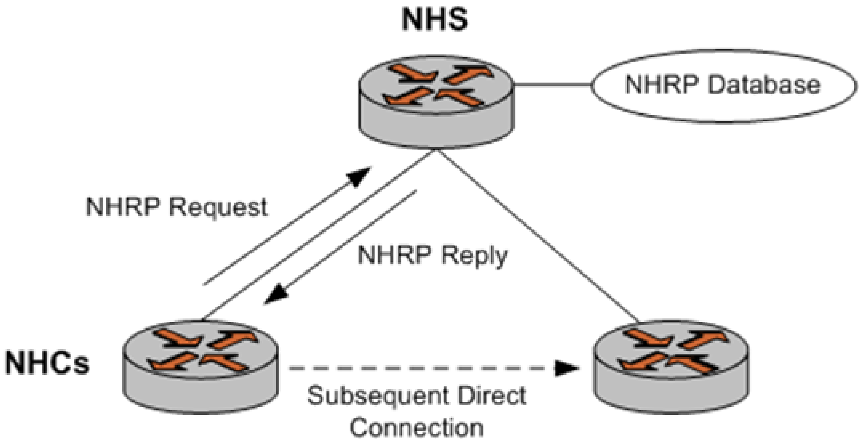
NHRP allows you to reduce the number of paths through the network and reduces the need for static configuration, helping provide the connectivity of a full mesh but greater scalability.

NHSs and NHCs

In an NHRP-enabled network, a router is configured with NHRP as a next-hop server (NHS). The NHS becomes a kind of route server, maintaining an NHRP database mapping the NBMA next hop to IP addresses.

The NHRP database is dynamically built and kept accurate using an ARP-like query-and-reply mechanism. Devices that need to communicate register dynamically with the NHS as a next-hop client (NHC). Having determined the existence of the NHC, the NHS adds it to the NBMA network without configuration. The NHC dynamically determines the locations of the other devices in the network from the NHS. The first communication of an NHC to a given device in the network initially flows through the NHS. During its first communication to a destination device, the NHC queries for the IP address of the destination device from the NHS and determines it. Thereafter, the NHC initiates a dynamic GRE/IPSec tunnel to the destination device and traffic flows directly from the NHC to the device. The tunnel is torn down when traffic is no longer flowing between the two devices.

Figure 12: NHRP



Supported standards

The AT&T Vyatta vRouter implementation of NHRP complies with the RFC 2332: NBMA Next Hop Resolution Protocol (NHRP) standard.

NHRP configuration

NHRP is frequently used with multipoint Generic Routing Encapsulation (mGRE) to create networks of dynamically built point-to-point tunnels. When these tunnels are secured with IP Security (IPsec), the result is a dynamic multipoint virtual private network (DMVPN).

- Configuration examples using NHRP with mGRE are provided in AT&T Vyatta Network Operating System Tunnels Configuration Guide.
- Configuration examples using NHRP in a DMVPN are provided in AT&T Vyatta Network Operating System DMVPN Configuration Guide.

NHRP Commands

interfaces tunnel <tunx> nhrp

Enables NHRP on a tunnel interface.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp`**

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp`**

Syntax:
`show interfaces tunnel` *tunx* **`nhrp`**

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp
    }
}
```

Use this command to enable NHRP on a tunnel interface.

Use the `set` form of this command to enable NHRP on a tunnel interface.

Use the **`delete`** form of this command to remove NHRP from a tunnel interface.

Use the `show` form of this command to display NHRP configuration on a tunnel interface.

interfaces tunnel <tunx> nhrp authentication <secret>

Specifies a password that authenticates NHRP packets.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp authentication`** *secret*

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp authentication`**

Syntax:
`show interfaces tunnel` *tunx* **`nhrp authentication`**

NHRP packets are not authenticated.

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

secret

Mandatory. A plain text password that authenticates packets. The password is a maximum of eight characters.

Configuration mode

```
interfaces {
   tunnel tunx {
      nhrp {
         authentication secret
      }
   }
}
```

Use this command to specify a password that authenticates NHRP packets. The password is embedded in all outgoing NHRP packets. All incoming NHRP packets on the interface are discarded unless the password is present.

Use the `set` form of this command to specify a password that authenticates NHRP packets.

Use the `delete` form of this command to delete an authentication password.

Use the `show` form of this command to display an authentication password.

interfaces tunnel <tunx> nhrp dynamic-map nbma-domain <nbma-domain-name>

Specifies that protocol address-to-nonbroadcast multiaccess (NBMA) address mappings are to be determined dynamically by using the next-hop server whose fully qualified domain name (FQDN) is specified.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp dynamic-map nbma-domain`** *nbma-domain-name*

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp dynamic-map nbma-domain`** *nbma-domain-name*

Syntax:
`show interfaces tunnel` *tunx* **`nhrp dynamic-map nbma-domain`**

tunx
Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

nbma-domain-name
Mandatory. The FQDN of the next-hop server.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            dynamic-map {
                nbma-domain nbma-domain-name
            }
        }
    }
}
```

Use this command to specify that protocol address-to-NBMA address mappings are to be determined dynamically by using the next-hop server whose FQDN is specified.

Use the `set` form of this command to specify the FQDN of the next-hop server.

Use the `delete` form of this command to remove the FQDN of the next-hop server.

Use the `show` form of this command to display the FQDN of the next-hop server.

interfaces tunnel <tunx> nhrp dynamic-map protocol-address <protocol-addr>/<prefix>

Specifies that protocol address-to-NBMA address mappings are to be determined dynamically by using the next-hop server whose protocol address is specified.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp dynamic-map protocol-address`** *protocol-addr* **/** *prefix*

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp dynamic-map protocol-address`** *protocol-addr* **/** *prefix*

Syntax:
`show interfaces tunnel` *tunx* **`nhrp dynamic-map protocol-address`**

tunx
Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

protocol-addr
Mandatory. The protocol address of the next-hop server. The protocol address is an IPv4 address.

prefix
Mandatory. The protocol address prefix of the next-hop server. The protocol address prefix is an IPv4 prefix.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            dynamic-map {
                protocol-address protocol-addr/prefix
            }
        }
    }
}
```

Use this command to specify that protocol address-to-NBMA address mappings are to be determined dynamically using the next-hop server whose NBMA IP address is specified.

Use the `set` form of this command to specify the NBMA IP address of the next-hop server.

Use the `delete` form of this command to remove the NBMA IP address of the next-hop server.

Use the `show` form of this command to display the NBMA IP address of the next-hop server.

interfaces tunnel <tunx> nhrp holding-time <time>

Specifies the hold time for NHRP Registration requests and Resolution replies sent from an interface or a shortcut target.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp holding-time`** *time*

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp holding-time`**

Syntax:
`show interfaces tunnel` *tunx* **`nhrp holding-time`**

The hold time is 7,200 seconds (two hours).

tunx
Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

time
Mandatory. The hold time in seconds. The default time is 7200 seconds (two hours).

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            holding-time time
        }
    }
}
```

Use this command to specify the hold time for NHRP Registration requests and Resolution replies sent from an interface or a shortcut target. The hold time is the amount of time that the system retains its knowledge of protocol address-to-NBMA address mappings. If not refreshed within the hold time, the mappings are discarded and need to be re-established.

Use the `set` form of this command to specify the hold time for NHRP Registration requests and Resolution replies sent from an interface or a shortcut target.

Use the `delete` form of this command to delete the hold time.

Use the `show` form of this command to display the hold time.

interfaces tunnel <tunx> nhrp map <protocol-addr>/<prefix> nbma-address <nbma-addr>

Statically maps a protocol address or protocol address prefix to the NBMA address of an NBMA peer.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr* **/** *prefix* **`nbma-address`** *nbma-addr*

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr* **/** *prefix* **`nbma-address`** *nbma-addr*

Syntax:
`show interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr* **/** *prefix* **`nbma-address`**

tunx
Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

protocol-addr
Mandatory. The protocol address of a destination. The protocol address is an IPv4 address.

prefix
Mandatory. The protocol address prefix of the destination. The protocol address prefix is an IPv4 prefix.

nbma-addr
Mandatory. The IPv4 address of the destination in the NBMA network.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            map protocol-addr/prefix {
                nbma-address nbma-addr
            }
        }
    }
}
```

Use this command to statically map a protocol address or protocol address prefix of a destination to an NBMA address in an NBMA network.

Use the `set` form of this command to map a destination protocol address or protocol address prefix to an NBMA address in an NBMA network.

Use the `delete` form of this command to remove a protocol address-to-NBMA address mapping.

Use the `show` form of this command to display a protocol address-to-NBMA address mapping.

interfaces tunnel <tunx> nhrp map <protocol-addr>/<prefix> register

Specifies that an NHRP Registration Request should be sent to the peer when the NHRP process starts.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr/prefix* **`register`**

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr/prefix* **`register`**

Syntax:
`show interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr/prefix*

Automatic NHRP registration is disabled.

tunx
Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

protocol-addr
Mandatory. The protocol address of a destination. The protocol address is an IPv4 address.

prefix
Mandatory. The protocol address prefix of the destination. The protocol address prefix is an IPv4 prefix.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            map protocol-addr/prefix {
                register
            }
        }
    }
}
```

Use this command to enable automatic sending of an NHRP Registration request when the NHRP process starts.

Use the `set` form of this command to enable automatic sending of an NHRP Registration request when the NHRP process starts up.

Use the `delete` form of this command to reset automatic NHRP registration to its default configuration.

Use the `show` form of this command to display NHRP address-mapping configuration.

interfaces tunnel <tunx> nhrp map <protocol-addr>/<prefix> register-no-unique

Enables NHRP Registration requests and replies to be sent without the unique flag being set.

Syntax:

`set interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr* **/** *prefix* **`register-no-unique`**

Syntax:

`delete interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr* **/** *prefix* **`register-no-unique`**

Syntax:

`show interfaces tunnel` *tunx* **`nhrp map`** *protocol-addr* **/** *prefix*

The unique flag is set in NHRP Registration requests and replies.

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

protocol-addr

Mandatory. The protocol address of a destination. The protocol address is an IPv4 address.

prefix

Mandatory. The protocol address prefix of the destination. The protocol address prefix is an IPv4 prefix.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            map protocol-addr/prefix {
                register-no-unique
            }
        }
    }
}
```

Use this command to enable NHRP Registration requests and replies to be sent without the unique flag being set. When the unique flag is set in an NHRP Registration request, a next-hop server rejects the request if the IP address to NBMA address mapping has changed and the request is received before the entry in the next-hop server times out. This rejection is typically the case when the client receives a new IP address, for example, in a DHCP environment.

Use the `set` form of this command to enable NHRP Registration requests and replies to be sent without the unique flag being set.

Use the `delete` form of this command to enable the default setting of the unique flag in NHRP Registration requests and replies.

Use the `show` form of this command to display NHRP address-mapping configuration.

interfaces tunnel <tunx> nhrp multicast parameters

Specifies how NHRP should soft switch multicast traffic.

Syntax:

`set interfaces tunnel` *tunx* **`nhrp multicast parameters`** { **`dynamic`** | **`nhs`** }

Syntax:

`delete interfaces tunnel` *tunx* **`nhrp multicast parameters`**

Syntax:
`show interfaces tunnel` *tunx* **`nhrp multicast parameters`**

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

`dynamic`

Forwards all multicast packets to all directly connected peers.

`nhs`

Repeats all multicast packets to each statically configured next hop.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            multicast {
                parameters {dynamic | nhs}
            }
        }
    }
}
```

Use this command to specify how NHRP should soft switch multicast traffic. Only one of the two options, **`dynamic`** or **`nhs`**, can be specified.

Note: Take care to avoid multicast repetition when multiple next-hop servers are present.

Use the `set` form of this command to specify how NHRP should soft switch multicast traffic.

Use the `delete` form of this command to remove the multicast parameters configuration.

Use the `show` form of this command to display the multicast parameters configuration.

interfaces tunnel <tunx> nhrp multicast protocol-address <protocol-addr>

Instructs NHRP to forward multicast traffic to a specific protocol address.

Syntax:
`set interfaces tunnel` *tunx* **`nhrp multicast protocol-address`** *protocol-addr*

Syntax:
`delete interfaces tunnel` *tunx* **`nhrp multicast protocol-address`** *protocol-addr*

Syntax:
`show interfaces tunnel` *tunx* **`nhrp multicast protocol-address`**

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

protocol-addr

Mandatory. The protocol address of a destination. The protocol address is an IPv4 address.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            multicast {
                protocol-address protocol-addr
            }
        }
    }
}
```

Use this command to instruct NHRP to forward multicast traffic to the specified protocol address.

> **Note:** Take care to avoid multicast repetition when multiple next-hop servers are present.

Use the `set` form of this command to instruct NHRP to forward multicast traffic to a specific protocol address.

Use the `delete` form of this command to remove the multicast protocol-address configuration.

Use the `show` form of this command to display the multicast protocol-address configuration.

interfaces tunnel <tunx> nhrp redirect

Instructs the sender of a forwarding packet to create a direct connection with the destination.

Syntax:

```
set interfaces tunnel tunx nhrp redirect
```

Syntax:

```
delete interfaces tunnel tunx nhrp redirect
```

Syntax:

```
show interfaces tunnel tunx nhrp redirect
```

Forwarding packets are forwarded normally.

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            redirect
        }
    }
}
```

Use this command to instruct the sender of a forwarding packet to create a direct connection with the destination. This connection is achieved by sending NHRP Traffic Indication packets back to the sender of the forwarding packet.

Use the `set` form of this command to instruct the sender of a forwarding packet to create a direct connection with the destination.

Use the `delete` form of this command to remove the redirect configuration.

Use the `show` form of this command to display the redirect configuration.

interfaces tunnel <tunx> nhrp shortcut

Enables the creation of a shortcut route.

Syntax:

```
set interfaces tunnel tunx nhrp shortcut
```

Syntax:

```
delete interfaces tunnel tunx nhrp shortcut
```

Syntax:

```
show interfaces tunnel tunx nhrp shortcut
```

Shortcut routes are not created.

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            shortcut
        }
    }
}
```

Use this command to enable the creation of a shortcut route; the resolution and establishment of a shortcut route is triggered when an NHRP Traffic Indication packet is received.

> **Note:** You still need to use routing protocol or have static routes to the hub node in your NBMA network. NHRP does not advertise routes; creates a shortcut route only for an already routable subnet.

Use the `set` form of this command to enable the creation of a shortcut route.

Use the `delete` form of this command to remove the shortcut configuration.

Use the `show` form of this command to display the shortcut configuration.

interfaces tunnel <tunx> nhrp shortcut-destination

Instructs NHRP to create a shortcut route to a subnet located on the interface.

Syntax:

```
set interfaces tunnel tunx nhrp shortcut-destination
```

Syntax:

```
delete interfaces tunnel tunx nhrp shortcut-destination
```

Syntax:

```
show interfaces tunnel tunx nhrp shortcut-destination
```

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            shortcut-destination
        }
    }
}
```

Use this command to instruct NHRP to create a shortcut route to a subnet located on the interface.

Use the `set` form of this command to instruct NHRP to create a shortcut route to a subnet located on the interface.

Use the `delete` form of this command to remove the shortcut-destination configuration.

Use the `show` form of this command to display the shortcut-destination configuration.

interfaces tunnel <tunx> nhrp shortcut-target <protocol-addr>/<prefix>

Defines an off-NBMA network prefix for which the Generic Routing Encapsulation (GRE) interface acts as a gateway.

Syntax:

```
set interfaces tunnel tunx nhrp shortcut-target protocol-addr / prefix
```

Syntax:

```
delete interfaces tunnel tunx nhrp shortcut-target protocol-addr / prefix
```

Syntax:

```
show interfaces tunnel tunx nhrp shortcut-target
```

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

protocol-addr

Mandatory. The protocol address of a destination. The protocol address is an IPv4 address.

prefix

Mandatory. The protocol address prefix of the destination. The protocol address prefix is an IPv4 prefix.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            shortcut-target protocol-addr/prefix
        }
    }
}
```

Use this command to define an off-NBMA network prefix for which the GRE interface acts as a gateway.

Use the `set` form of this command to define an off-NBMA network prefix for which the GRE interface acts as a gateway.

Use the `delete` form of this command to remove the shortcut-target configuration.

Use the `show` form of this command to display the shortcut-target configuration.

interfaces tunnel <tunx> nhrp shortcut-target holding-time <time>

Specifies the hold time for Resolution Requests and Resolution Responses.

Syntax:

```
set interfaces tunnel tunx nhrp shortcut-target holding-time time
```

Syntax:

```
delete interfaces tunnel tunx nhrp shortcut-target holding-time
```

Syntax:

```
show interfaces tunnel tunx nhrp shortcut-target holding-time
```

tunx

Mandatory. The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

time

Mandatory. The hold time in seconds.

Configuration mode

```
interfaces {
    tunnel tunx {
        nhrp {
            shortcut-target holding-time time
        }
    }
}
```

Use this command to specify the hold time for Resolution Requests and Resolution Responses.

Use the `set` form of this command to specify the hold time for Resolution Requests and Resolution Responses.

Use the `delete` form of this command to remove the hold time configuration.

Use the `show` form of this command to display the hold time configuration.

reset ip nhrp flush tunnel

Removes all non-permanent entries.

Syntax:

```
reset ip nhrp flush tunnel [ tunx ]
```

Non-permanent entries for all tunnel interfaces are removed.

tunx

The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

Operational mode

Use this command to remove all non-permanent entries for the specified tunnel interface. If no tunnel interface is specified, all non-permanent entries for all tunnel interfaces are removed.

The following example shows how to remove all non-permanent entries for the tun0 tunnel.

```
vyatta@vyatta:~$ reset ip nhrp flush tunnel tun0
vyatta@vyatta:~$
```

reset ip nhrp purge tunnel

Removes and reregisters all NHRP entries.

Syntax:

```
reset ip nhrp purge tunnel [ tunx ]
```

NHRP entries for all tunnel interfaces are removed.

tunx

The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

Operational mode

Use this command to remove and reregister all NHRP entries for a specific tunnel interface. If no tunnel interface is specified, all NHRP entries for all tunnel interfaces are removed and reregistered.

The following example shows how to remove and reregister all NHRP entries for the tun0 tunnel.

```
vyatta@vyatta:~$ reset ip nhrp purge tunnel tun0
vyatta@vyatta:~$
```

show ip nhrp tunnel

Displays NHRP information about a specific tunnel.

Syntax:

`show ip nhrp tunnel` [*tunx*]

NHRP information for all tunnel interfaces is displayed.

tunx

The identifier for a tunnel interface. The identifier ranges from tun0 through tun*x*, where *x* is a non-negative integer.

Operational mode

Use this command to display NHRP information about a specific tunnel. If no tunnel interface is specified, NHRP information about all tunnel interfaces is displayed.

The following example shows how to display NHRP information about the tun0 tunnel.

```
vyatta@vyatta:~$ show ip nhrp tunnel tun0
Interface: tun0
Type: static
Protocol-Address: 10.0.0.100/24
NBMA-Address: 172.18.0.1
vyatta@vyatta:~$
```

Path Performance Monitoring

Path Performance Monitoring overview

The Path Performance Monitoring feature introduces the Path Monitor service, which monitors the quality of network paths on a periodic basis. Path Monitor lets you configure one or more Path Monitor hosts. Each host determines the target to be monitored (the path endpoint) and the types of probes it supports. Path Monitor also allows the configuration of policies that define the required quality of a network path. A Path Monitor instance defines the association between a host and policy set. The instance runs in a given context (source interface), which determines the path being monitored.

A policy engine determines whether the result samples of a Path Monitor comply with the requirements of each associated policy, on a one-to-one basis. You can use the policy compliance results to influence PBR in the forwarding plane.

Path Monitor uses ping as the probe (type) to measure path performance.

Components of a Path Performance Monitoring solution

For path monitoring to work, you must configure the following components.

Component	Description
Path Monitor host	You must configure at least one Path Monitor host. A Path Monitor host is a collection of parameters that define the following: • The IP address or domain name of the target host (a Path Monitor monitors the path to the target host) • The monitor type (currently, only ping is supported on the vRouter) Optionally, you can specify a DSCP value to tag the monitor packets, which helps ensure that the ping traffic uses the same QoS as the traffic on the monitored link. If you do not specify a DSCP value, the ping traffic might encounter different treatment en-route, which results in unrepresentative or meaningless measurements.
Path Monitor policy	You must configure at least one Path Monitor policy. A Path Monitor policy is a collection of parameters that define whether the monitoring results comply with the policy.
Path Monitor	You must configure at least one Path Monitor instance. A Path Monitor instance is an association between a Path Monitor host, a context (a sample interval and a source interface), and one or more Path Monitor policies. Compliance is considered on a one-to-one basis between a Path Monitor instance and each of its associated policies. A Path Monitor instance pings a Path Monitor host and measures the round-trip time (RTT). If the RTT is within the defined threshold, as defined by the associated Path Monitor policy, the path complies with the policy.

Component	Description
Routing instances	AT&T recommends that you configure routing instances to isolate the traffic for each path. The routes within these instances can be either statically or dynamically learned.
PBR policy	To route traffic based upon path performance, you must also configure at least one PBR policy. This is not required if you only want to passively monitor the paths.

You can use Path Monitor instances, hosts, and policies, and PBR policies as building blocks to create varied and simple or complex routing decision flows.

Path Monitor policy compliance states and parameters

A Path Monitor policy defines three compliance states:

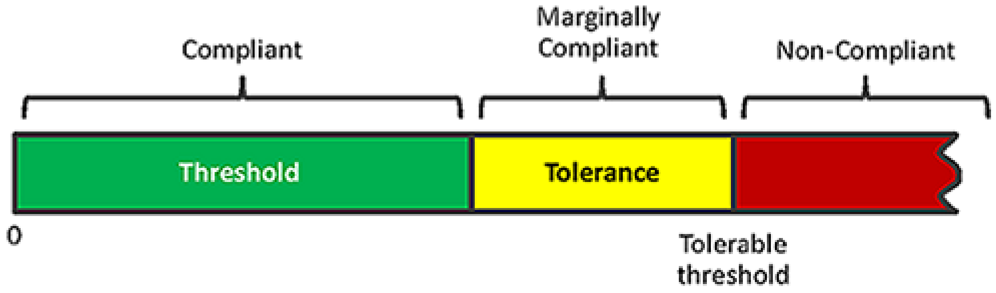
- Compliant
- Marginally Compliant
- Non-Compliant

To determine compliance, a Path Monitor pings the associated Path Monitor host and calculates the average RTT value in milliseconds for the ping packets. Then, the Path Monitor checks whether the RTT is within the values defined by the following parameters.

Parameter	Description
Threshold	Defines the upper limit for RTT at or below which the RTT complies with a Path Monitor policy. The threshold ranges from 5 through 10000 milliseconds (ms).
Tolerance	Introduces a dampening mechanism that is biased toward staying in compliance. The sum of threshold and tolerance defines the upper limit for RTT beyond which the RTT does not comply with a Path Monitor policy. The tolerance ranges from 5 (default) through 1000 ms.
Robustness	Introduces an additional dampening mechanism to prevent excessive policy state flapping. The robustness parameter is disabled by default (when disabled, robustness is 1). The robustness parameter determines the number of consecutive Path Monitor result samples with the same compliance result that are required to cause a policy compliance transition between the Compliant and Non-Compliant states. If the current state is Compliant and the robustness mechanism is triggered, the compliance state changes to Marginally Compliant. The robustness parameter allows a tradeoff between stability in bursty traffic conditions and speed of detection. The value of the robustness parameter ranges from 1 (default) through 10.

The following diagram shows the relationship between the Path Monitor compliance states and the threshold and tolerance parameters.

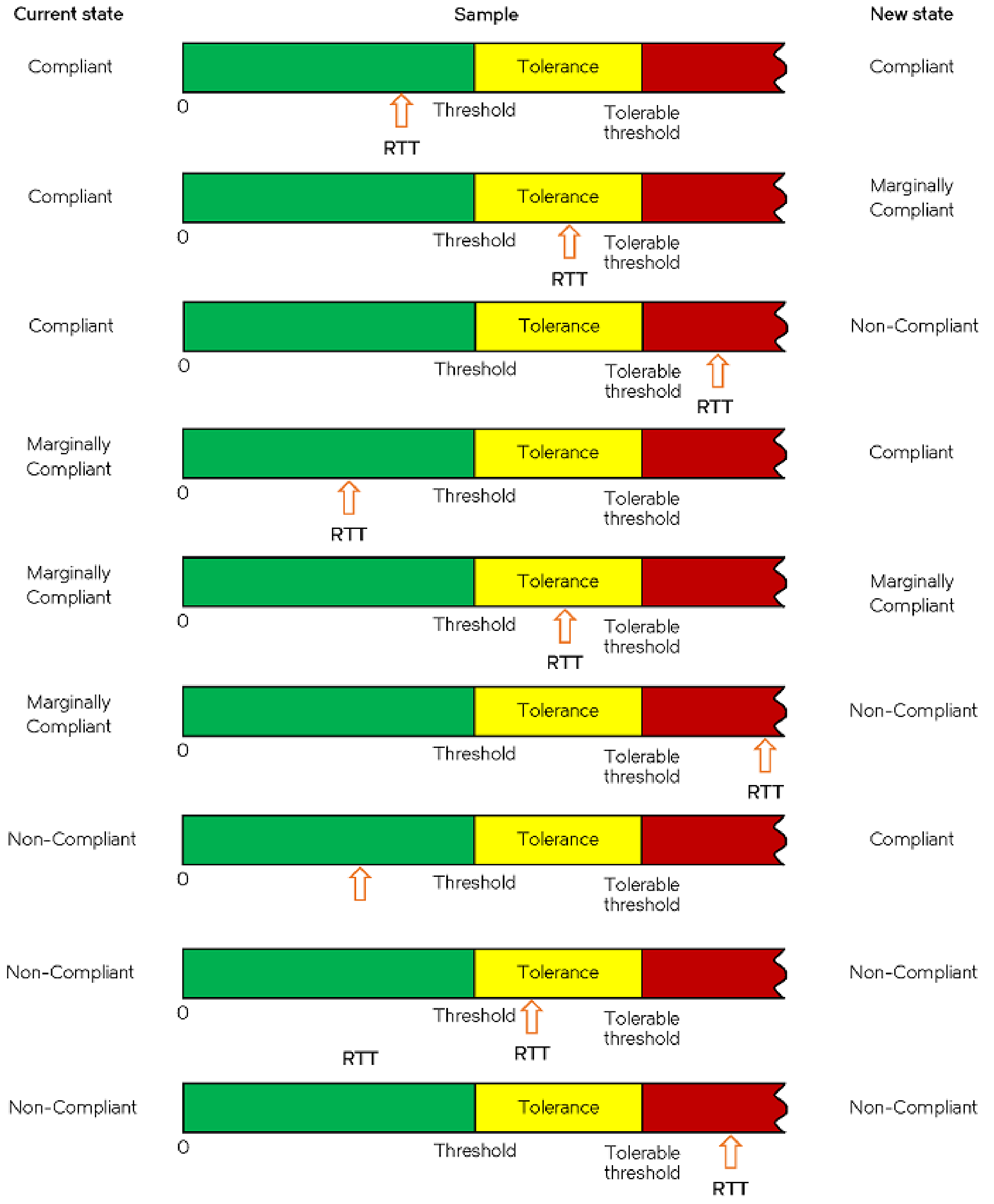
Figure 13: Path Monitor compliance states



When robustness is disabled (robustness = 1), the following table and figure show how the Path Monitor policy engine determines the new compliance state of a Path Monitor based on the current state of the Path Monitor and the RTT value.

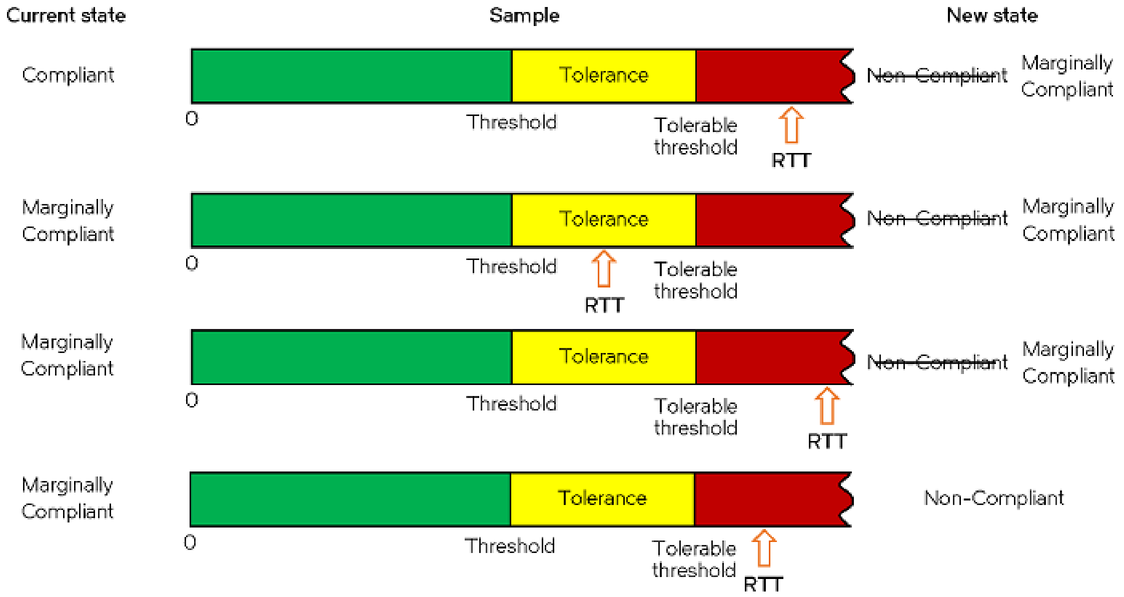
Current state		New state
Compliant, Marginally Compliant, or Non-Compliant	RTT <= Threshold	Compliant
Compliant, Marginally Compliant	Threshold < RTT <= (Threshold + Tolerance)	Marginally Compliant
Compliant, Marginally Compliant	RTT > (Threshold + Tolerance)	Non-Compliant
Non-Compliant	RTT > Threshold	Non-Compliant

Figure 14: How new Path Monitor compliance states are determined with a robustness of 1



The following figure shows how the Path Monitor policy engine determines the new compliance state for a Path Monitor when you set robustness to 2. In this figure, the new states for the third and fourth samples are different, even though RTT is greater than the tolerable threshold in both samples. The reason for the different states is that it takes two consecutive Non-Compliant RTT values for the new state to change from Marginally Compliant to Non-Compliant.

Figure 15: How new Path Monitor compliance states are determined with a robustness of 2



Path Monitor parameters

When configuring a Path Monitor, you can configure the following parameters.

Parameter	Description
Interface	You must configure an interface through which traffic flows to the Path Monitor host and bind the Path Monitor to the interface. You can bind the interface to any routing instance.
Host	You must specify the target Path Monitor host to be monitored.
Interval	You can configure how often a Path Monitor generates results. The interval ranges from 5 through 120 seconds. The default is 5 seconds.
Policy	You must associate one or more policies with a Path Monitor. For each policy, the Path Monitor determines the compliance state of the policy each time the Path Monitor samples results. For example, if the interval is 10 seconds, the Path Monitor samples results every 10 seconds.
Initial state	You can specify the initial compliance state to be used for all policies that are associated with a Path Monitor. By default, the initial state is Non-Compliant.

Monitoring behavior

The Path Monitor follows this process to monitor paths:

1. A Path Monitor starts a ping session toward the target.
2. The session sends 5 packets and collects the average RTT result.
3. The Path Monitor compares the result with the requirements of each policy associated with the Path Monitor, in turn, and determines the current compliance state of each policy.

4. The Path Monitor pushes the compliance state of each policy to PBR, which influences local routing decisions.
5. The Path Monitor waits until the interval period has elapsed since the last session started, then repeats the same process with a new session.

If the Path Monitor detects more than 25 percent packet loss in ping packets, the Path Monitor discards the RTT result and considers the collected sample non-compliant with all policies. Each configured Path Monitor operates independently of other Path Monitors.

Logging

Path Monitor logs a message when the results of a monitor go into or out of compliance with the requirements of a policy. For example:

```
monitord[30173]: [Monitor 's4'][Policy 'cloud'] now compliant
monitord[30173]: [Monitor 's4'][Policy 'cloud'] no longer compliant
```

If the compliance is marginal (the results are in the tolerance range or the robustness count has been triggered), Path Monitor logs a message. For example:

```
monitord[30173]: [Monitor 's3'][Policy 'cloud'][Requirement 'round trip time'] Compliance is
 marginal (result above threshold by 11.856 and robustness count is 1)
```

If Path Monitor cannot determine any results (that is, cannot send any packets) or experiences packet loss over 25 percent, which is not enough to accurately determine compliance, Path Monitor logs a message. For example:

```
monitord[30173]: Monitor s2 failed
```

This message indicates that the results sample is considered to be out of compliance with all policies associated with that Path Monitor.

If updates to the policy-compliance state cannot be programmed in the data plane because of a data plane issue, Path Monitor logs a message. For example:

```
monitord[30690]: Failure running 'pathmonitor s3,cloud noncompliant' on dataplane:
                 Command '['/opt/vyatta/bin/vplsh', '-l', '-c', 'pathmonitor s3,cloud
 noncompliant']' returned non-zero exit status 1
monitord[30690]: [Monitor 's3'][Policy 'cloud'] Failed to implement policy compliance state 'False'
```

If Path Monitor cannot process the monitoring results, it logs a message. For example:

```
monitord[30173]: [Monitor 's4'][Policy 'cloud'] Receiving monitor results faster than they can be
 processed
```

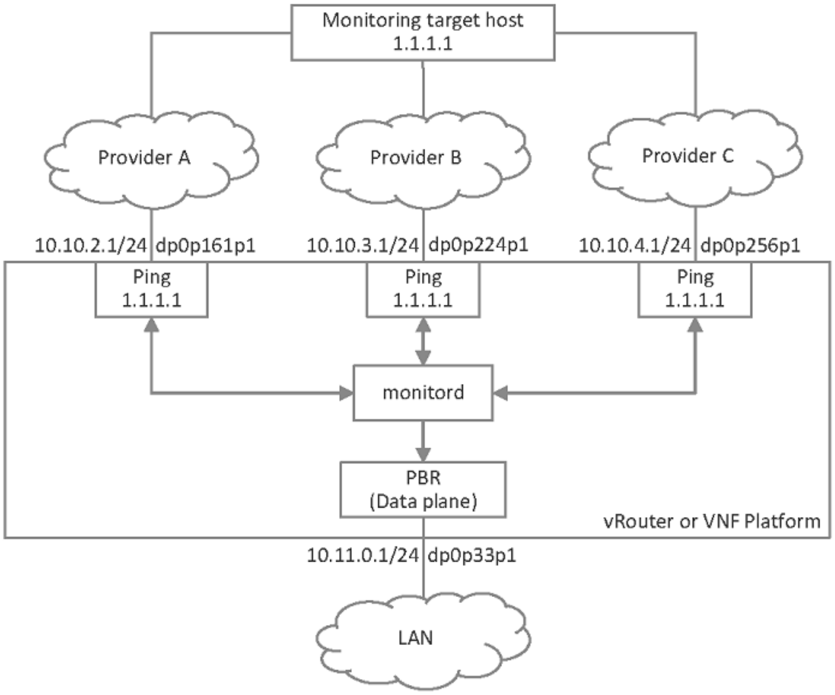
Configuring optimized application routing based on path performance monitoring

The configuration example in this section supports the following topology diagram. In this example, three Path Monitor instances monitor the network paths for three provider networks and switches traffic between providers based on path quality. Based on the Path Monitor configuration in this example, the vRouter routes the VoIP traffic over the best path.

The following assumptions apply to this example.

- dp0p161p1 is a fibre Internet link.
- dp0p224p1 is a lower-quality backup fibre Internet link.
- dp0p256p1 is a backup LTE Internet link.
- dp0p33p1 is the ingress interface for all traffic destined to the Internet.
- The provider for non-VoIP traffic is chosen by using ECMP.

Figure 16: Path Monitor topology diagram



Configuring the interfaces

Before configuring the Path Monitor settings, configure the interfaces for the LAN network and the three provider networks (Provider A, Provider B, and Provider C). To configure the interfaces for these networks, perform the following steps.

Table 25: Configuring the interfaces

Step	Command

Configure the interface for the LAN network, the source of the VoIP traffic.	`vyatta@vyatta# set interfaces dataplane dp0p33p1 address '10.11.0.1/24'`
	`vyatta@vyatta# set interfaces dataplane dp0p33p1 description 'LAN'`

Configure the interface for the Provider A network.	`vyatta@vyatta# set interfaces dataplane dp0p161p1 address '10.10.2.1/24'`
	`vyatta@vyatta# set interfaces dataplane dp0p161p1 description 'Provider A'`

Configure the interface for the Provider B network.	`vyatta@vyatta# set interfaces dataplane dp0p224p1 address '10.10.3.1/24'`
	`vyatta@vyatta# set interfaces dataplane dp0p224p1 description 'Provider B'`

Configure the interface for the Provider C network.	`vyatta@vyatta# set interfaces dataplane dp0p256p1 address '10.10.4.1/24'`
	`vyatta@vyatta# set interfaces dataplane dp0p256p1 description 'Provider C'`

Step	Command

Commit and view the configuration.	`vyatta@vyatta# commit`
	`[edit]`
	`vyatta@vyatta# show interfaces dataplane`
	` dataplane dp0p33p1 {`
	`        address 10.11.0.1/24`
	`        description LAN`
	` }`
	` dataplane dp0p161p1 {`
	`        address 10.10.2.1/24`
	`        description "Provider A"`
	` }`
	` dataplane dp0p224p1 {`
	`        address 10.10.3.1/24`
	`        description "Provider B"`
	` }`
	` dataplane dp0p256p1 {`
	`        address 10.10.4.1/24`
	`        description "Provider C"`
	` }`

Configuring Path Monitor

To configure the Path Monitor settings for this example, perform the following steps.

Table 26: Configuring Path Monitor

Step	Command

Create a Path Monitor host named dc and configure the parameters for ping sessions to this host.	`vyatta@vyatta# set service path-monitor host dc target 1.1.1.1`
	`vyatta@vyatta# set service path-monitor host dc type ping`

Create a policy named voip and set the round-trip-time threshold to 5 ms and the tolerance to 10 ms.
When the average measured round-trip time falls within, or falls below, the 5 ms threshold, the path being monitored complies with the policy. When the measured round-trip time is greater than 15 ms (threshold + tolerance), the path no longer complies with the policy.
If the path complies with a Path Monitor policy and the Path Monitor measures a round-trip time greater than 5 ms and less than 15 ms, the monitor reports the state as Marginally Compliant.
	`vyatta@vyatta# set service path-monitor policy voip requires type ping round-trip-time threshold 5`
	`vyatta@vyatta# set service path-monitor policy voip requires type ping round-trip-time tolerance 10`

Create the following associations:
• A policy association between provider-a, a ping Path Monitor, and the voip policy
• A ping association between provider-a and the dc host
• A ping association between provider-a and the dp0p161p1 interface
Note: A path monitor must have exactly one host and one or more SLA policies.
	`vyatta@vyatta# set service path-monitor monitor provider-a policy voip`
	`vyatta@vyatta# set service path-monitor monitor provider-a type ping host dc`
	`vyatta@vyatta# set service path-monitor monitor provider-a type ping interface dp0p161p1`

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Create similar associations for provider-b.</td><td>

```
vyatta@vyatta# set service path-monitor
 monitor provider-b policy voip
vyatta@vyatta# set service path-monitor
 monitor provider-b type ping host dc
vyatta@vyatta# set service path-monitor
 monitor provider-b type ping interface
 dp0p224p1
```

</td></tr>
<tr><td>Create similar associations for provider-c.</td><td>

```
vyatta@vyatta# set service path-monitor
 monitor provider-c policy voip
vyatta@vyatta# set service path-monitor
 monitor provider-c type ping host dc
vyatta@vyatta# set service path-monitor
 monitor provider-c type ping interface
 dp0p256p1
```

</td></tr>
<tr><td>Commit and view the configuration.</td><td>

```
vyatta@vyatta# commit
[edit]
vyatta@vyatta# show service path-monitor
 monitor
 monitor provider-a {
        policy voip
        type {
                ping {
                        host dc
                        interface dp0p161p1
                }
        }
 }
 monitor provider-b {
        policy voip
        type {
                ping {
                        host dc
                        interface dp0p224p1
                }
        }
 }
 monitor provider-c {
        policy voip
        type {
                ping {
                        host dc
                        interface dp0p256p1
                }
        }
 }
[edit]
```

</td></tr>
</table>

Configuring PBR policies

Before creating PBR policies, configure routing instances for the three providers, as shown in the following example.

Table 27: Configuring routing instances

Step	Command

Configure the routing instance for Provider A. Add a route for return LAN traffic and a default route so that all egress traffic goes through the provider network.	`vyatta@vyatta# set routing routing-instance provider-a interface dp0p161p1`
	`vyatta@vyatta# set routing routing-instance provider-a protocols static interface-route 10.11.0.0/24 next-hop-routing-instance default next-hop-interface dp0p33p1`
	`vyatta@vyatta# set routing routing-instance provider-a protocols static route 0.0.0.0/0 next-hop 10.10.2.2`

Configure the routing instance for Provider B. Add a route for return LAN traffic and a default route so that all egress traffic goes through the provider network.	`vyatta@vyatta# set routing routing-instance provider-b interface dp0p224p1`
	`vyatta@vyatta# set routing routing-instance provider-b protocols static interface-route 10.11.0.0/24 next-hop-routing-instance default next-hop-interface dp0p33p1`
	`vyatta@vyatta# set routing routing-instance provider-b protocols static route 0.0.0.0/0 next-hop 10.10.3.2`

Configure the routing instance for Provider C. Add a route for return LAN traffic and a default route so that all egress traffic goes through the provider network.	`vyatta@vyatta# set routing routing-instance provider-c interface dp0p256p1`
	`vyatta@vyatta# set routing routing-instance provider-c protocols static interface-route 10.11.0.0/24 next-hop-routing-instance default next-hop-interface dp0p33p1`
	`vyatta@vyatta# set routing routing-instance provider-c protocols static route 0.0.0.0/0 next-hop 10.10.4.2`

Step	Command
Commit and view the configuration.	

```
vyatta@vyatta# commit
[edit]
vyatta@vyatta# show
 routing-instance provider-a {
        interface dp0p161p1
        protocols {
                static {
                        interface-route
 10.11.0.0/24 {
                                next-hop-
routing-instance default {
                                       next-
hop-interface dp0p33p1
                                }
                        }
                        route 0.0.0.0/0 {
                                next-hop
 10.10.2.2
                        }
                }
        }
 }
 routing-instance provider-b {
        interface dp0p224p1
        protocols {
                static {
                        interface-route
 10.11.0.0/24 {
                                next-hop-
routing-instance default {
                                       next-
hop-interface dp0p33p1
                                }
                        }
                        route 0.0.0.0/0 {
                                next-hop
 10.10.3.2
                        }
                }
        }
 }
 routing-instance provider-c {
        interface dp0p256p1
        protocols {
                static {
                        interface-route
 10.11.0.0/24 {
                                next-hop-
routing-instance default {
                                       next-
hop-interface dp0p33p1
                                }
                        }
                        route 0.0.0.0/0 {
                                next-hop
 10.10.4.2
                        }
                }
        }
 }
[edit]
```

After configuring the routing instances for the provider networks, create PBR policies, as shown in the following example. In this example, the PBR policy named voip is applied to all interfaces that receive egress application traffic.

> **Note:** Multiple PBR policies may be applied to the dp0p33p1 interface, if necessary. Likewise, the voip policy may be applied to multiple interfaces, if necessary.

Table 28: Configuring PBR settings

Step	Command

Create a PBR rule to route traffic through Provider A, if it complies with the voip policy. In this example, Provider A is the preferred provider for IPv4 VoIP traffic, so the example starts by first checking the compliance of Provider A. If the provider-a Path Monitor determines that Provider A complies with the voip policy, the vRouter accepts all IPv4 VoIP traffic and routes it through the provider-a routing instance.	`vyatta@vyatta# set policy route pbr voip rule 10 action accept`
	`vyatta@vyatta# set policy route pbr voip rule 10 address-family ipv4`
	`vyatta@vyatta# set policy route pbr voip rule 10 application type voip`
	`vyatta@vyatta# set policy route pbr voip rule 10 path-monitor monitor provider-a policy voip`
	`vyatta@vyatta# set policy route pbr voip rule 10 routing-instance provider-a`

Create a PBR rule to route traffic through Provider B, if it complies with the voip policy. If Provider A does not comply with the voip policy, the provider-b Path Monitor checks the compliance of Provider B with the policy. If provider-b determines that Provider B complies with the voip policy, the vRouter accepts all IPv4 VoIP traffic and routes it through the provider-b routing instance.	`vyatta@vyatta# set policy route pbr voip rule 20 action accept`
	`vyatta@vyatta# set policy route pbr voip rule 20 address-family ipv4`
	`vyatta@vyatta# set policy route pbr voip rule 20 application type voip`
	`vyatta@vyatta# set policy route pbr voip rule 20 path-monitor monitor provider-b policy voip`
	`vyatta@vyatta# set policy route pbr voip rule 20 routing-instance provider-b`

Create a PBR rule to route traffic through Provider C. Provider C is the final backup link for VoIP traffic. If both Provider A and Provider B do not comply with their respective policies, all VoIP traffic goes through Provider C and there is no need to check policy compliance for Provider C. In this example, there is really no need for the provider-c Path Monitor, but it is configured for completeness. This step is very important because it ensures that VoIP traffic does not fall through to the default routing table. In PBR, if none of the rules match, the vRouter uses the default table of the routing instance for the ingress interface to route traffic.
Note: Alternatively, you could check the policy compliance of Provider C and, in this example, if the default routing instance table is used, ECMP chooses the path.
	`vyatta@vyatta# set policy route pbr voip rule 30 action accept`
	`vyatta@vyatta# set policy route pbr voip rule 30 address-family ipv4`
	`vyatta@vyatta# set policy route pbr voip rule 30 application type voip`
	`vyatta@vyatta# set policy route pbr voip rule 30 routing-instance provider-c`

Step	Command
Commit and view the configuration.	(see below)
Apply the voip routing policy to the inbound traffic on the dp0p33p1 data plane interface.	(see below)
Commit and view the configuration.	(see below)

Commit and view the configuration.

```
vyatta@vyatta# commit
[edit]
vyatta@vyatta# show policy route pbr voip
 pbr voip {
        rule 10 {
                action accept
                address-family ipv4
                application {
                        type voip
                }
                path-monitor {
                        monitor provider-a {
                                policy voip
                        }
                }
                routing-instance provider-a
        }
        rule 20 {
                action accept
                address-family ipv4
                application {
                        type voip
                }
                path-monitor {
                        monitor provider-b {
                                policy voip
                        }
                }
                routing-instance provider-b
        }
        rule 30 {
                action accept
                address-family ipv4
                application {
                        type voip
                }
                routing-instance provider-c
        }
 }
[edit]
```

Apply the voip routing policy to the inbound traffic on the dp0p33p1 data plane interface.

```
vyatta@vyatta# set interfaces dataplane
 dp0p33p1 policy route pbr voip
```

Commit and view the configuration.

```
vyatta@vyatta# commit
[edit]
vyatta@vyatta# show interfaces dataplane
 dp0p33p1 policy
 policy {
        route {
                pbr voip
        }
 }
[edit]
```

Configuring default routes for non-VoIP traffic

Configure default routes for handling non-VoIP traffic, as shown in the following example.

Table 29: Configuring default routes for non-VoIP traffic

Step	Command
Configure a default route for the Provider A network.	`vyatta@vyatta# set protocols static route 0.0.0.0/0 next-hop-routing-instance provider-a next-hop 10.10.2.2`
Configure a default route for the Provider B network.	`vyatta@vyatta# set protocols static route 0.0.0.0/0 next-hop-routing-instance provider-b next-hop 10.10.3.2`
Configure a default route for the Provider C network.	`vyatta@vyatta# set protocols static route 0.0.0.0/0 next-hop-routing-instance provider-c next-hop 10.10.4.2`

Commit and view the configuration.	`vyatta@vyatta# commit`
	`[edit]`
	`vyatta@vyatta# show protocols static route 0.0.0.0/0 next-hop-routing-instance`
	`next-hop-routing-instance provider-a {`
	`        next-hop 10.10.2.2`
	`}`
	`next-hop-routing-instance provider-b {`
	`        next-hop 10.10.3.2`
	`}`
	`next-hop-routing-instance provider-c {`
	`        next-hop 10.10.4.2`
	`}`
	`[edit]`

Configuring NAT settings

Configuring NAT settings is optional, but to ensure that any return traffic goes through the same provider network, configure source NAT masquerading on all traffic entering the provider networks, as shown in the following example.

Table 30: Configuring NAT settings

Step	Command

Configure source NAT masquerading on all traffic entering the Provider A network.	`vyatta@vyatta# set service nat source rule 1 description 'Provider A'`
	`vyatta@vyatta# set service nat source rule 1 outbound-interface dp0p161p1`
	`vyatta@vyatta# set service nat source rule 1 translation address masquerade`

Configure source NAT masquerading on all traffic entering the Provider B network.	`vyatta@vyatta# set service nat source rule 20 description 'Provider B'`
	`vyatta@vyatta# set service nat source rule 20 outbound-interface dp0p224p1`
	`vyatta@vyatta# set service nat source rule 20 translation address masquerade`

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Configure source NAT masquerading on all traffic entering the Provider C network.</td><td>

```
vyatta@vyatta# set service nat source rule 30
 outbound-interface dp0p256p1
vyatta@vyatta# set service nat source rule 30
 description 'Provider C'
vyatta@vyatta# set service nat source rule 30
 translation address masquerade
```

</td></tr>
<tr><td>Commit and view the configuration.</td><td>

```
vyatta@vyatta# commit
[edit]
vyatta@vyatta# show service nat {
        source {
                rule 1 {
                        description "Provider
 A"
                        outbound-interface
 dp0p161p1
                        translation {
                                address
 masquerade
                        }
                }
                rule 20 {
                        description "Provider
 B"
                        outbound-interface
 dp0p224p1
                        translation {
                                address
 masquerade
                        }
                }
                rule 30 {
                        description "Provider
 C"
                        outbound-interface
 dp0p256p1
                        translation {
                                address
 masquerade
                        }
                }
        }
 }
[edit]
```

</td></tr>
</table>

Path Monitor Commands

monitor path-monitor

Displays the logs, or toggles debugging logs, for Path Monitor.

Syntax:

`monitor path-monitor [ **enable** | **disable** ]`

`enable`

Enables additional debugging logs.

`disable`

Reverts to the default logging of only errors, warnings, and changes to the policy compliance status.

Operational mode

Use this command to enable additional debugging logs or revert to the default logging of only errors, warnings, and changes to the policy compliance status.

The following example shows how to enable additional debugging logs for Path Monitor.

```
vyatta@vyatta:~$ monitor path-monitor enable
```

service path-monitor host <host-name>

Creates a Path Monitor host.

Syntax:

`set service path-monitor host` *host-name*

Syntax:

`delete service path-monitor host` *host-name*

Syntax:

`show service path-monitor host`

host-name

The alphanumeric name of the host. The maximum number of characters in the name is 30.

Configuration mode

```
service {
        path-monitor {
                host host-name
        }
}
```

Use the `set` form of this command to create a Path Monitor host.

Use the `delete` form of this command to remove a Path Monitor host.

Use the `show` form of this command to display the configured Path Monitor hosts.

service path-monitor host <host-name> target <target-name-or-address>

Sets the DNS name or IP address of the target host acting as the endpoint of the paths to be monitored.

Syntax:
`set service path-monitor host` *host-name* **`target`** *target-name-or-address*

Syntax:
`delete service path-monitor host` *host-name* **`target`** *target-name-or-address*

Syntax:
`show service path-monitor host` *host-name* **`target`**

host-name
The alphanumeric name of the host. The maximum number of characters in the name is 30.

target-name-or-address
The DNS name or IPv4 or IPv6 address of the target host.

Configuration mode

```
service {
       path-monitor {
              host host-name {
                     target target-name-or-address
              }
       }
}
```

Use the `set` form of this command to set the DNS name or IP address of the target host.

Use the `delete` form of this command to delete the DNS name or IP address of the target host.

Use the `show` form of this command to display the DNS name or IP address of the target host.

service path-monitor host <host-name> type ping

Instructs the Path Monitor to use the IP ping method for path monitoring.

Syntax:
`set service path-monitor host` *host-name* **`type ping`** [**`dscp`** *dscp-name-or-value*]

Syntax:
`delete service path-monitor host` *host-name* **`type ping`** [**`dscp`** *dscp-name-or-value*]

Syntax:
`show service path-monitor host` *host-name* **`type ping`** [**`dscp`**]

host-name
The alphanumeric name of the host. The maximum number of characters in the name is 30.

`dscp` ***dscp-name-or-value***
Specifies the IP DSCP name (for example, af11, cs1, or default) or an absolute value to be applied during ping operations. The default value is **`default`**.

Configuration mode

```
service {
       path-monitor {
              host host-name {
                     type {
                            ping {
                                   dscp dscp-name-or-value
                            }
                     }
              }
       }
}
```

Use the `set` form of this command to set IP ping as the method for path monitoring.

Use the `delete` form of this command to remove IP ping as the method for path monitoring.

Use the `show` form of this command to view the configured method for path monitoring.

service path-monitor monitor <monitor-name> initial-state <state>

Determines how PBR handles matching traffic before the first Path Monitor measurements are available.

Syntax:
`set service path-monitor monitor` *monitor-name* **`initial-state`** *state*

Syntax:
`delete service path-monitor monitor` *monitor-name* **`initial-state`** *state*

Syntax:
`show service path-monitor monitor` *monitor-name* **`initial-state`**

monitor-name

The alphanumeric name of a Path Monitor instance. The maximum number of characters in the name is 30.

state

The name of the compliance state. You can specify either of the following states:

`compliant`: All associated Path Monitor policies are compliant.

`non-compliant`: All associated Path Monitor policies are non-compliant.

Configuration mode

```
service {
      path-monitor {
            monitor monitor-name {
                  initial-state state
            }
      }
}
```

The initial state applies to all Path Monitor policies that are associated with the Path Monitor instance that this command specifies. Only those routing PBR policy rules that include this instance become compliant or non-compliant according to the initial-state setting in this command. PBR rules that do not include Path Monitor instances are not affected.

Use the `set` form of this command to configure how PBR handles matching traffic before the first Path Monitor measurements are available.

Use the `delete` form of this command to remove the initial state configuration for a Path Monitor.

Use the `show` form of this command to display the initial state configuration for a Path Monitor.

service path-monitor monitor <monitor-name> policy <policy-name>

Specifies the policy to be used by a Path Monitor.

Syntax:
`set service path-monitor monitor` *monitor-name* **`policy`** *policy-name*

Syntax:
`delete service path-monitor monitor` *monitor-name* **`policy`** *policy-name*

Syntax:
`show service path-monitor monitor` *monitor-name* **`policy`**

monitor-name
The alphanumeric name of a Path Monitor instance.

policy-name
The name of a Path Monitor policy. This policy must have already been defined by the **service path-monitor policy** command.

Configuration mode

```
service {
      path-monitor {
            monitor monitor-name {
                  policy policy-name
            }
      }
}
```

Use the `set` form of this command to specify the policy to be used by a Path Monitor. You can specify multiple policies by running this command multiple times.

Use the `delete` form of this command to remove a Path Monitor policy from the list of policies used by a Path Monitor.

Use the `show` form of this command to display the list of policies used by a Path Monitor.

service path-monitor monitor <monitor-name> type ping host <host-name>

Specifies the host to be pinged by a Path Monitor.

Syntax:
`set service path-monitor monitor` *monitor-name* **`type ping host`** *host-name*

Syntax:
`delete service path-monitor monitor` *monitor-name* **`type ping host`** *host-name*

Syntax:
`show service path-monitor monitor` *monitor-name* **`type ping host`**

monitor-name
The alphanumeric name of a Path Monitor instance.

host-name
The alphanumeric name of the host. This host must have already been defined by the **`service path-monitor host`** command.

Configuration mode

```
service {
      path-monitor {
            monitor monitor-name {
                  type {
                        ping {
                              host host-name
                        }
                  }
            }
      }
}
```

Use the `set` form of this command to specify the host to be pinged by a Path Monitor.

Use the `delete` form of this command to stop the pinging of the host.

Use the `show` form of this command to display the hosts to be pinged.

service path-monitor monitor <monitor-name> type ping interface <interface-name>

Specifies the interface to be used as the source for monitoring path activity.

Syntax:
`set service path-monitor monitor` *monitor-name* **`type ping interface`** *interface-name*

Syntax:
`delete service path-monitor monitor` *monitor-name* **`type ping interface`** *interface-name*

Syntax:
`show service path-monitor monitor` *monitor-name* **`type ping interface`**

monitor-name
The alphanumeric name of a Path Monitor instance.

interface-name
The name of the source interface for monitoring activity.

Configuration mode

```
service {
      path-monitor {
            monitor monitor-name {
                  type {
                        ping {
                              interface interface-name
                        }
                  }
            }
      }
}
```

Use the `set` form of this command to specify the source interface for monitoring activity.

Use the `delete` form of this command to delete the source interface for monitoring activity.

Use the `show` form of this command to display the source interface for monitoring activity.

service path-monitor monitor <monitor-name> type ping interval <ping-interval>

Sets the ping interval, that is, the number of seconds between successive path-monitoring operations.

Syntax:

```
set service path-monitor monitor monitor-name type ping interval ping-interval
```

Syntax:

```
delete service path-monitor monitor monitor-name type ping interval ping-interval
```

Syntax:

```
show service path-monitor monitor monitor-name type ping interval
```

monitor-name

The name of a Path Monitor instance.

ping-interval

The number of seconds between successive path-monitoring operations. The ping interval ranges from 5 (default) through 120 seconds.

Configuration mode

```
service {
      path-monitor {
            monitor monitor-name {
                  type {
                        ping {
                              interval ping-interval
                        }
                  }
            }
      }
}
```

Use the `set` form of this command to set the ping interval for monitoring path activity.

Use the `delete` form of this command to delete the ping interval.

Use the `show` form of this command to display the ping interval.

service path-monitor policy <policy-name>

Creates a Path Monitor policy.

Syntax:

```
set service path-monitor policy policy-name
```

Syntax:

```
delete service path-monitor policy policy-name
```

Syntax:

```
show service path-monitor policy
```

policy-name

The alphanumeric name of a Path Monitor policy. The maximum number of characters in the name is 30.

Configuration mode

```
service {
      path-monitor {
            policy policy-name
      }
}
```

Use the `set` form of this command to create a Path Monitor policy.

Use the `delete` form of this command to delete a Path Monitor policy.

Use the `show` form of this command to display the configured Path Monitor policies.

service path-monitor policy <policy-name> requires type ping round-trip-time robustness <r-count>

Sets the number of consecutive measurement samples that a Path Monitor must collect before indicating a change in the state of a Path Monitor policy (in or out of compliance).

Syntax:
`set service path-monitor policy` *policy-name* **`requires type ping round-trip-time robustness`** *r-count*

Syntax:
`delete service path-monitor policy` *policy-name* **`requires type ping round-trip-time robustness`** *r-count*

Syntax:
`show service path-monitor policy` *policy-name* **`requires type ping round-trip-time robustness`**

policy-name
The alphanumeric name of a Path Monitor policy.

r-count
The number of consecutive measurement samples that must be met before indicating a change in the state of a policy (in or out of compliance). The number ranges from 1 (default) through 10.

Configuration mode

```
service {
     path-monitor {
          policy policy-name {
               requires {
                    type {
                         ping {
                              round-trip-time {
                                   robustness r-count
                              }
                         }
                    }
               }
          }
     }
}
```

Use the `set` form of this command to set the number of consecutive measurement samples that a Path Monitor must collect before indicating a change in the state of a Path Monitor policy.

Use the `delete` form of this command to delete the number of consecutive measurement samples.

Use the `show` form of this command to display the number of consecutive measurement samples.

service path-monitor policy <policy-name> requires type ping round-trip-time threshold <rtt-threshold>

Sets the threshold for the round-trip time.

Syntax:
`set service path-monitor policy` *policy-name* **`requires type ping round-trip-time threshold`** *rtt-threshold*

Syntax:
`delete service path-monitor policy` *policy-name* **`requires type ping round-trip-time threshold`** *rtt-threshold*

Syntax:
`show service path-monitor policy` *policy-name* **`requires type ping round-trip-time threshold`**

policy-name
The alphanumeric name of the policy.

rtt-threshold
The threshold for the round-trip time, in milliseconds (ms). If the RTT is less than or equal to *rtt-threshold* , the RTT complies with this policy. The threshold for the round-trip time ranges from 5 through 10000 ms.

Configuration mode

```
service {
     path-monitor {
          policy policy-name {
               requires {
                    type {
                         ping {
                              round-trip-time {
                                   threshold rtt-threshold
                              }
                         }
                    }
               }
          }
     }
}
```

Use the `set` form of this command to set the round-trip-time threshold.

Use the `delete` form of this command to remove the round-trip-time threshold.

Use the `show` form of this command to display the round-trip-time threshold.

service path-monitor policy <policy-name> requires type ping round-trip-time tolerance <rtt-delta>

Sets the threshold delta for the round-trip time that is measured by a Path Monitor.

Syntax:
`set service path-monitor policy` *policy-name* **`requires type ping round-trip-time tolerance`** *rtt-delta*

Syntax:
`delete service path-monitor policy` *policy-name* **`requires type ping round-trip-time tolerance`** *rtt-delta*

Syntax:
`show service path-monitor policy` *policy-name* **`requires type ping round-trip-time tolerance`**

policy-name
The alphanumeric name of a Path Monitor policy.

rtt-delta
The threshold delta for the measured round-trip time by a Path Monitor. The threshold delta is the degree to which the measured round-trip time can exceed a defined threshold before Path Monitor

declares a policy non-compliant. The round-trip time tolerance ranges from 5 (default) through 1000 milliseconds.

Configuration mode

```
service {
     path-monitor {
          policy policy-name {
               requires {
                    type {
                         ping {
                              round-trip-time {
                                   tolerance rtt-delta
                              }
                         }
                    }
               }
          }
     }
}
```

Use the `set` form of this command to set the threshold delta for the measured round-trip time by a Path Monitor.

Use the `delete` form of this command to remove the threshold delta.

Use the `show` form of this command to display show the configured threshold delta.

show service path-monitor monitor <monitor-name>

Displays a summary of the policy compliance status for a Path Monitor instance.

Syntax:
`show service path-monitor monitor` *monitor-name*

monitor-name
The name of a Path Monitor instance.

Operational mode

Use this command to display a summary of the policy compliance status for a Path Monitor instance.

The following example shows how to display the policy compliance status for a Path Monitor instance named cust1-fibre-voice.

```
vyatta@vyatta:~$ show service path-monitor monitor cust1-fibre-voice
Monitor                        Policy                         Status
-------                        ------                         ------
cust1-fibre-voice              cust1-voice                    Compliant
cust1-fibre-voice              cust1-voice-low                Compliant
vyatta@vyatta:~$
```

show service path-monitor summary

Displays a summary of the policy compliance status for all configured Path Monitor instances.

Syntax:
`show service path-monitor summary`

Operational mode

Use this command to display a summary of the policy compliance status for all configured Path Monitor instances.

The following examples show how to display a summary of the policy compliance status for three configured Path Monitors. The output in the following examples shows different compliance status values for the cust1-lte-voice monitor.

```
vyatta@vyatta:~$ show service path-monitor summary
Monitor                        Policy                         Status
-------                        ------                         ------
cust1-fibre-voice              cust1-voice                    Compliant
cust1-fibre-voice              cust1-voice-low                Compliant
cust1-lte-voice                cust1-voice-low                Compliant
vyatta@vyatta:~$
```

```
vyatta@vyatta:~$ show service path-monitor summary
Monitor                        Policy                         Status
-------                        ------                         ------
cust1-fibre-voice              cust1-voice                    Compliant
cust1-fibre-voice              cust1-voice-low                Compliant
cust1-lte-voice                cust1-voice-low                Marginally Compliant
vyatta@vyatta:~$
```

```
vyatta@vyatta:~$ show service path-monitor summary
Monitor                        Policy                         Status
-------                        ------                         ------
cust1-fibre-voice              cust1-voice                    Compliant
cust1-fibre-voice              cust1-voice-low                Compliant
cust1-lte-voice                cust1-voice-low                Non-Compliant
vyatta@vyatta:~$
```

Port Mirroring

Overview

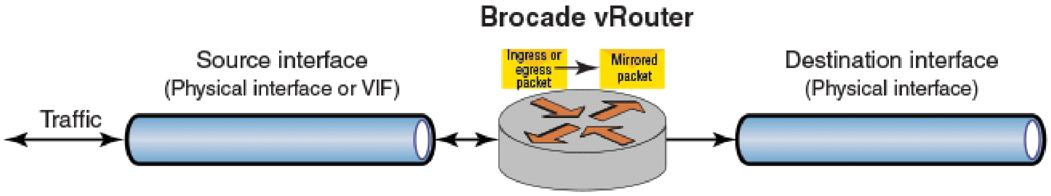
Switch Port Analyzer (SPAN), Remote SPAN (RSPAN), and Encapsulated RSPAN (ERSPAN) enable you to monitor and troubleshoot network traffic.

SPAN overview

SPAN mirrors traffic on one or more source interfaces on an AT&T Vyatta vRouter to a destination interface on the same router. Source interfaces can be both physical and VLAN interfaces. Both ingress and egress traffic on source interfaces can be mirrored. For SPAN, the destination interface is a physical port to which a network monitoring tool is connected for capturing and analyzing the traffic.

The following figure shows a SPAN port mirroring session.

Figure 17: SPAN port mirroring session



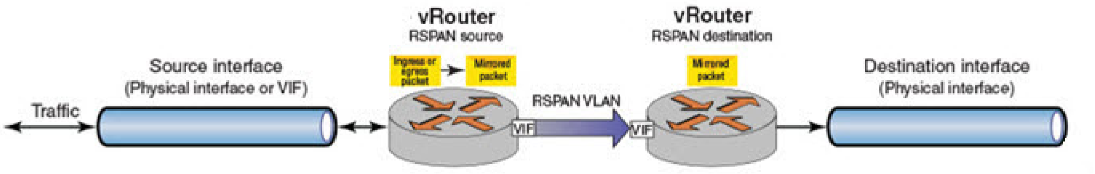
RSPAN overview

RSPAN mirrors traffic to a destination interface on a remote AT&T Vyatta vRouter. RSPAN mirroring has source and destination AT&T Vyatta vRouters. On the source router, traffic from source interfaces is mirrored to a VLAN interface (VIF). This RSPAN VLAN is dedicated to transporting mirrored traffic to the remote router. RSPAN on the destination router receives mirrored traffic on this VLAN interface and forwards it to a destination interface on this router.

Source interfaces can be both physical and VLAN interfaces. Both ingress and egress traffic on source interfaces can be mirrored. For an RSPAN destination session, the destination interface is a physical port to which a network monitoring tool is connected for capturing and analyzing the traffic.

The following figure shows an RSPAN port mirroring session on source and destination AT&T Vyatta vRouters.

Figure 18: RSPAN port mirroring session



> **Note:** When a packet with a VLAN ID arrives at an RSPAN source port, RSPAN creates a mirror copy of the packet and overwrites the VLAN ID in the mirrored packet with the RSPAN VLAN ID, which results in an untagged mirror copy of the original packet.

ERSPAN overview

ERSPAN mirrors traffic to a destination interface on a remote AT&T Vyatta vRouter. Similar to RSPAN, ERSPAN mirroring has source and destination AT&T Vyatta vRouters. However, it uses an ERSPAN tunnel to transmit mirrored packets from the source vRouter to the destination vRouter. The tunnel is configured with local and remote IP addresses and is dedicated to transporting mirrored traffic to the remote vRouter.

On the source vRouter, traffic from source interfaces is mirrored and prepended by Type II or Type III headers. These headers are defined in the following IETF draft proposal.

https://tools.ietf.org/html/draft-foschiano-erspan-00

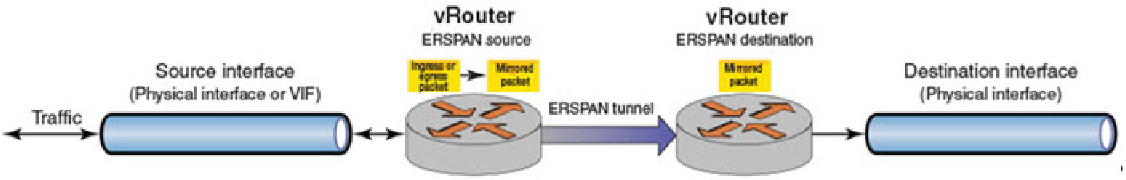
Then, the ERSPAN on the source vRouter sends the traffic to the tunnel. ERSPAN on the destination router receives the mirrored traffic from the tunnel and forwards it to a destination interface on this router for traffic analysis.

Source interfaces are physical interfaces. Both ingress and egress traffic on source interfaces can be mirrored. For an ERSPAN destination session, the destination interface is a physical port to which a network monitoring tool is connected for capturing and analyzing the traffic.

The following figure shows an ERSPAN port mirroring session on source and destination AT&T Vyatta vRouters.

Note: VRF support is not currently available for ERSPAN.

Figure 19: ERSPAN port mirroring session



Port monitor filters for port mirroring

You can configure rule-based filters at the port monitor session level to restrict the volume of ingress or egress IPv4 traffic for port mirroring. Filters configured for a port monitor session apply to the traffic from all the source interfaces for the session. The traffic is filtered before it is mirrored and sent out over the destination interface. You can apply filters to SPAN, RSPAN-source, and ERSPAN-source sessions.

If the source interface of a portmonitor session is a physical interface, all packets on the interface are mirrored. If filters are configured on the portmonitor session for a physical interface, the filters are applied to all traffic received on the interface.

Port monitor filters are constructed using firewall rules. The following limitations apply:

- Only stateless IPv4 packet filters are supported.
- Rules can match source IP address, destination IP address, source port, destination port, IP protocol, or DSCP.

To create and apply port monitor filters:

1. Create firewall rules to use as the port monitor filtering rules. See *AT&T Vyatta Network Operating System Firewall Configuration Guide* for information on creating firewall rules.
2. Apply each filter to the ingress (in) or egress (out) traffic for a port monitoring session.

Configuration requirements and limits

The AT&T Vyatta vRouter has the following configuration requirements and limits for SPAN, RSPAN, and ERSPAN.

- A total of eight monitoring sessions is supported.
- By default, a port monitoring session is enabled. When a source interface is configured, packets from the interface are mirrored only when the session type and the destination interface for the session are also configured. For ERSPAN, the indentifier (ID) and header also must be specified.
- When a session is configured for port monitoring, the type value must be specified. The value is mandatory.
- The SPAN type supports the following interfaces:

  - Source interface, physical or VLAN
  - Destination interface, physical only

- The RSPAN-source type supports the following interfaces:
  - Source interface, physical or VLAN
  - Destination interface, VLAN only

- The RSPAN-destination type supports the following interfaces:
  - Source interface, VLAN only
  - Destination interface, physical only

- The ERSPAN-source type supports the following interfaces:
  - Source interface, physical or VLAN
  - Destination interface, ERSPAN tunnel only

- The ERSPAN-destination type supports the following interfaces:
  - Source interface, ERSPAN tunnel only
  - Destination interface, physical only

- A source interface and a destination interface cannot be same.
- A total of eight distinct source interfaces is allowed for all sessions.
- The default directions for monitoring on a source interface are both receive (RX) and transmit (TX). The direction can also be configured as RX only or TX only.
- Source interfaces cannot be shared between sessions.
- The source VIF for the RSPAN-destination type must be part of a bridge group.
- A destination interface that is configured for monitoring transmits mirrored packets only. Do not use this interface for other traffic.
- The destination interface must be configured in the system, cannot be disabled, and cannot have address, IPv4, and IPv6 attributes.
- A physical destination interface cannot have QoS configured on it.
- Only one destination interface for each session is allowed and cannot be shared between different sessions.
- The destination interface for any session cannot be part of a bridge group.
- An RSPAN-destination or ERSPAN-destination type can have only one source interface for a port monitoring session.
- The source interface of an RSPAN-destination or ERSPAN-destination type cannot have its direction set.
- For port monitoring sessions, you are not allowed to configure a physical interface and VIF from the same physical interface together. For example, both dp0s7 and dp0s7.700 are not allowed.
- You cannot change the type of the session after you commit the configuration.
- You cannot change the ERSPAN identifier and header properties of an ERSPAN-source or ERSPAN-destination session type after you commit the configuration.

SPAN port mirroring configuration

SPAN port mirroring requires the configuration of an AT&T Vyatta vRouter as provided in the following example.

Table 31: Configuring SPAN port mirroring

Step	Command

Configure a SPAN session.	`vyatta@vyatta# set service portmonitor`
	` session 1 type span`

Configure the source interface for the session. By default, the monitoring of the traffic direction is both ingress and egress.	`vyatta@vyatta# set service portmonitor`
	` session 1 source dp0s3`

Configure the destination interface for the session.	`vyatta@vyatta# set service portmonitor`
	` session 1 destination dp0s4`

Commit the configuration.	`vyatta@vyatta# commit`
Save the configuration. Port monitoring for the session is enabled if the **`type`**, **`source`** and **`destination`** parameters are configured properly.	`vyatta@vyatta# save`

Display the port mirroring configuration.	`vyatta@vyatta:~$ show portmonitor session`
	`Session:                      1`
	`    Type:                     span`
	`    State:                    enabled`
	`    Source interfaces:`
	`      Name:                   dp0s3`
	`        Direction:            both`
	`    Destination interface:    dp0s4`

RSPAN port mirroring configuration

RSPAN port mirroring requires the configuration of source and destination AT&T Vyatta vRouters.

Note:

For traffic to flow from RSPAN source and destination vRouters running on Citrix XenServer systems, XenServer must use Linux bridge instead of Open vSwitch. Also, aging on the bridge corresponding to the vRouter network must be set to 0. This setting makes the entries in the forwarding table permanent.

1. To check whether XenServer is using Open vSwitch or Linux bridge, use the `cat` command to view the network.conf file.

   ```
   # cat /etc/xensource/network.conf
   ```

2. If Open vSwitch is in use, use the following command to switch to Linux bridge.

   ```
   # xe-switch-network-backend bridge
   ```

3. Use the `xe network-list` command to find the bridge that corresponds to your network, similar to the following example.

   ```
   # xe network-list
   uuid ( RO)                : ...
             name-label ( RW): ...
       name-description ( RW):
                 bridge ( RO): xapi10
   ```

4. Use the `brctl setageing` command to set the aging for the bridge to 0.

   ```
   # brctl setageing xapi10 0
   ```

To change back to Open vSwitch, enter the following command.

```
# xe-switch-network-backend openvswitch
```

RSPAN-source port mirroring

The following example provides the configuration of RSPAN-source port mirroring on an AT&T Vyatta vRouter.

Table 32: Configuring RSPAN-source port mirroring

Step	Command
Configure an RSPAN-source session.	`vyatta@vyatta# set service portmonitor session 2 type rspan-source`
Configure the source physical interface for the session. By default, the monitoring of the traffic direction is both ingress and egress.	`vyatta@vyatta# set service portmonitor session 2 source dp0s5`
Configure the destination VLAN interface for the session.	`vyatta@vyatta# set service portmonitor session 2 destination dp0s7.700`
Commit the configuration.	`vyatta@vyatta# commit`
Save the configuration. Port monitoring for the session is enabled if the **`type`**, **`source`** and **`destination`** parameters are configured properly.	`vyatta@vyatta# save`

Display the port mirroring configuration.	`vyatta@vyatta:~$ show portmonitor session`
	`Session:                          2`
	`    Type:                         rspan-source`
	`    State:                        enabled`
	`    Source interfaces:`
	`      Name:                       dp0s5`
	`        Direction:                both`
	`    Destination interface:        dp0s7.700`

RSPAN-destination port mirroring

The following example provides the configuration of RSPAN-destination port mirroring on an AT&T Vyatta vRouter.

Table 33: Configuring RSPAN-destination port mirroring

Step	Command
Configure an RSPAN-destination session.	`vyatta@vyatta# set service portmonitor session 3 type rspan-destination`

Step	Command

Configure the source VLAN interface for the session. You cannot specify a direction for an RSPAN-destination session. The source VIF for the RSPAN-destination type must be part of a bridge group.	`vyatta@vyatta# set service portmonitor`
	` session 3 source dp0s7.700`

Configure the destination physical interface for the session.	`vyatta@vyatta# set service portmonitor`
	` session 3 destination dp0s6`

Commit the configuration.	`vyatta@vyatta# commit`
Save the configuration. Port monitoring for the session is enabled if the **`type`**, **`source`** and **`destination`** parameters are configured properly.	`vyatta@vyatta# save`

Display the port mirroring configuration.	`vyatta@vyatta:~$ show portmonitor session`
	`Session:                      3`
	`    Type:                     rspan-`
	`destination`
	`    State:                    enabled`
	`    Source interfaces:`
	`      Name:                   dp0s7.700`
	`        Direction:            both`
	`    Destination interface:    dp0s6`

ERSPAN port mirroring configurations

ERSPAN port mirroring requires the configuration of source and destination AT&T Vyatta vRouters.

ERSPAN-source port mirroring configuration

The following example provides the configuration of ERSPAN-source port mirroring on an AT&T Vyatta vRouter.

Table 34: Configuring ERSPAN-source port mirroring

Step	Command

Configure an ERSPAN GRE tunnel.
The local and remote IP addresses that are configured for the ERSPAN tunnel must be fully routable addresses for each side to ping the other address.
	`vyatta@vyatta# set interfaces dataplane`
	` dp0s11 address 15.1.1.1/24`
	`vyatta@vyatta# set interfaces erspan erspan0`
	` local-ip 15.1.1.1`
	`vyatta@vyatta# set interfaces erspan erspan0`
	` remote-ip 15.1.1.2`

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Display the tunnel configuration.</td><td>

```
vyatta@vyatta#show interfaces
 interfaces {
     dataplane dp0s11 {
         address 15.1.1.1/24
     }
     erspan erspan0 {
         local-ip 15.1.1.1
         remote-ip 15.1.1.2
     }
 }
```

</td></tr>
<tr><td>Configure an ERSPAN-source session.</td><td>

```
vyatta@vyatta# set service portmonitor
 session 22 type erspan-source
```

</td></tr>
<tr><td>Configure the source interface for the session.</td><td>

```
vyatta@vyatta# set service portmonitor
 session 22 source dp0s4.100
```

</td></tr>
<tr><td>Configure the destination ERSPAN tunnel for the session.</td><td>

```
vyatta@vyatta# set service portmonitor
 session 22 destination erspan0
```

</td></tr>
<tr><td>Configure the ERSPAN identifier.</td><td>

```
vyatta@vyatta# set service portmonitor
 session 22 erspan identifier 200
```

</td></tr>
<tr><td>Configure the ERSPAN header type.</td><td>

```
vyatta@vyatta# set service portmonitor
 session 22 erspan header type-II
```

</td></tr>
<tr><td>Commit the configuration.</td><td>

```
vyatta@vyatta# commit
```

</td></tr>
<tr><td>Save the configuration.
Port monitoring for the session is enabled if the <b>type</b>, <b>source</b>, <b>destination</b>, ERSPAN identifier, and ERSPAN header type parameters are configured properly.</td><td>

```
vyatta@vyatta# save
```

</td></tr>
<tr><td>Display the ERSPAN source configuration.</td><td>

```
vyatta@vyatta# show service portmonitor
 portmonitor {
     session 22 {
         destination erpsan0
         erspan {
             header type-II
             identifier 200
         }
         source dp0s4.100
         type erspan-source
     }
 }
```

</td></tr>
</table>

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Display the ERSPAN source information.</td><td><pre>
vyatta@vyatta:~$ show portmonitor session
Session: 22
    Type: erspan-source
    State: enabled
    erspan Identifier: 200
    erspan Header: type-II
    Source interfaces:
      Name: dp0s4.100
    Destination interface: erspan0
</pre></td></tr>
</table>

ERSPAN-destination port mirroring configuration

The following example provides the configuration of ERSPAN-destination port mirroring on an AT&T Vyatta vRouter.

Table 35: Configuring ERSPAN-destination port mirroring

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Configure an ERSPAN tunnel.
The local and remote IP addresses that are configured for the ERSPAN tunnel must be fully routable addresses for each side to ping the other address.</td><td><pre>
vyatta@vyatta# set interfaces dataplane
 dp0s11 address 15.1.1.2/24
vyatta@vyatta# set interfaces erspan erspan0
 local-ip 15.1.1.2
vyatta@vyatta# set interfaces erspan erspan0
 remote-ip 15.1.1.1
</pre></td></tr>
<tr><td>Display the tunnel configuration.</td><td><pre>
vyatta@vyatta#show interfaces
 interfaces {
     dataplane dp0s11 {
         address 15.1.1.2/24
     }
     erspan erspan0 {
         local-ip 15.1.1.2
         remote-ip 15.1.1.1
     }
 }
</pre></td></tr>
<tr><td>Configure an ERSPAN-destination session.</td><td><pre>
vyatta@vyatta# set service portmonitor
 session 22 type erspan-destination
</pre></td></tr>
<tr><td>Configure the ERSPAN source tunnel for the session.</td><td><pre>
vyatta@vyatta# set service portmonitor
 session 22 source erspan0
</pre></td></tr>
<tr><td>Configure the destination interface for the session.</td><td><pre>
vyatta@vyatta# set service portmonitor
 session 22 destination dp0s12
</pre></td></tr>
<tr><td>Commit the configuration.</td><td><pre>
vyatta@vyatta# commit
</pre></td></tr>
<tr><td>Configure the identifier.</td><td><pre>
vyatta@vyatta# set service portmonitor
 session 22 erspan identifier 200
</pre></td></tr>
</table>

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Configure the header type.</td><td>

```
vyatta@vyatta# set service portmonitor
 session 22 erspan header type-II
```

</td></tr>
<tr><td>Save the configuration.
Port monitoring for the session is enabled if the <b>type</b>, <b>source</b>, <b>destination</b>, ERSPAN identifier, and ERSPAN header type parameters are configured properly.</td><td>

```
vyatta@vyatta# save
```

</td></tr>
<tr><td>Display the ERSPAN destination configuration.</td><td>

```
vyatta@vyatta# show service portmonitor
 portmonitor {
     session 22 {
         destination dp0s12
         erspan {
             header type-II
             identifier 200
         }
         source erspan0
         type erspan-source
     }
 }
```

</td></tr>
<tr><td>Display the ERSPAN destination session information.</td><td>

```
vyatta@vyatta:~$ show portmonitor session
Session:                      22
    Type:                     erspan-
destination
    State:                    enabled
    erspan Identifier:        200
    erspan Header:            type-II
    Source interfaces:
      Name:                   erspan0
    Destination interface:    dp0s12
```

</td></tr>
</table>

Port monitor filter configuration

The following example shows how to create port monitor filters and apply them to a port monitor session.

Table 36: Configuring port monitor filters for port mirroring

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Define rule 10 for the firewall ruleset dev with the action to accept packets</td><td>

```
vyatta@vyatta# set security firewall name dev
 rule 10 action accept
```

</td></tr>
<tr><td>Identify the source subnet for rule 10</td><td>

```
vyatta@vyatta# set security firewall name dev
 rule 10 source address 50.3.1.0/24
```

</td></tr>
<tr><td>Identify the destination subnet for rule 10</td><td>

```
vyatta@vyatta# set security firewall name dev
 rule 10 destination address 60.2.0.0/24
```

</td></tr>
<tr><td>Identify the protocol (UDP) that rule 10 applies to</td><td>

```
vyatta@vyatta# set security firewall name dev
 rule 10 protocol udp
```

</td></tr>
</table>

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>Define rule 20 for the firewall ruleset dev with the action to drop packets</td><td>

```
vyatta@vyatta# set security firewall name dev
 rule 20 action drop
```

</td></tr>
<tr><td>Define rule 30 for the firewall ruleset dev2 with the action to accept packets</td><td>

```
vyatta@vyatta# set security firewall name
 dev2 rule 30 action accept
```

</td></tr>
<tr><td>Identify the source subnet for rule 30</td><td>

```
vyatta@vyatta# set security firewall name
 dev2 rule 30 source address 10.1.0.1
```

</td></tr>
<tr><td>Define rule 40 for the firewall ruleset dev2 with the action to drop packets</td><td>

```
vyatta@vyatta# set security firewall name
 dev2 rule 40 action drop
```

</td></tr>
<tr><td>Commit the configuration</td><td>

```
vyatta@vyatta# commit
```

</td></tr>
<tr><td>Show the configuration</td><td>

```
vyatta@vyatta# show security firewall
 firewall {
        name dev {
                rule 10 {
                        action accept
                        destination {
                                address
 60.2.0.0/24
                        }
                        protocol udp
                        source {
                                address
 50.3.1.0/24
                        }
                }
                rule 20 {
                        action drop
                }
        }
        name dev2 {
                rule 30 {
                        action accept
                        source {
                                address
 10.1.0.1
                        }
                }
                rule 40 {
                        action drop
                }
        }
 }
```

</td></tr>
<tr><td>Specify that port monitoring session 2 will use the dev ruleset to filter ingress traffic. Allowed session types are SPAN, RSPAN-source, and ERSPAN-source</td><td>

```
vyatta@vyatta# set service portmonitor
 session 2 filter in dev
```

</td></tr>
<tr><td>Specify that port monitoring session 2 will use the dev2 ruleset to filter egress traffic. Allowed session types are SPAN, RSPAN-source, and ERSPAN-source</td><td>

```
vyatta@vyatta# set service portmonitor
 session 2 filter out dev2
```

</td></tr>
</table>

Step	Command
Commit the configuration	`vyatta@vyatta# commit`

Show the configuration	`vyatta@vyatta# show service portmonitor`
	` session 2 filter`
	` filter {`
	`        in dev`
	`        out dev2`
	` }`

Port Monitoring Commands

interfaces erspan erspan<tunnel-number> ip tos <value>

Specifies the value to write into the Type of Service (ToS) byte of the IP header of an ERSPAN tunnel packet.

Syntax:
`set interfaces erspan` **`erspan`** *tunnel-number* **`ip tos`** *value*

Syntax:
`delete interfaces erspan` **`erspan`** *tunnel-number* **`ip tos`** `[` *value* `]`

Syntax:
`show interfaces erspan` **`erspan`** *tunnel-number* **`ip tos`**

The default value is inherit.

`erspan` ***tunnel-number***

The identifier of a tunnel interface with an integer for the *tunnel-number* variable.

value

Specifies the ToS value to write into the IP header of a tunnel packet. For the value, enter one of the following:

number—The ToS value to write into the header of the tunnel packet (the carrier packet). Enter a value from 0x00 to 0xFF. The 0x00 value means a tunnel packet copies the ToS value from the packet being encapsulated (the passenger packet).

default—The Default Class (00000) for best-effort traffic.

`af` *number*—The Assured Forwarding Class for assurance of delivery as defined in RFC 2597. Depending on the forwarding class and the drop precedence, the class can be one of the following values: **`af11`** through **`af13`**, **`af21`** through **`af23`**, **`af31`** through **`af33`**, or **`af41`** through **`af43`**.

`cs` *number*—Class Selector for network devices that use the Precedence field in the IPv4 header. The number ranges from 1 to 7 and indicates the precedence, for example cs1.

`ef`—Expedited Forwarding, per-hop behavior.

`inherit`—Inherit from original IP header.

`va`—Voice Admit, Capacity-Admitted Traffic.

Configuration mode

```
interfaces {
    erspan erspantunnel-number {
        ip {
            tos value
        }
    }
}
```

Use this command to specify the value to write into the 8-bit ToS byte of the IP header for a packet that traverses a tunnel interface. The ToS byte of the IP header of a packet specifies the forwarding behavior to be applied to the packet.

Use the `set` form of this command to specify the ToS value in the IP header.

Use the `delete` form of this command to reset the ToS value to its default of inherit.

Use the `show` form of this command to display the ToS value.

interfaces erspan erspan<tunnel-number> ip ttl <value>

Sets the time-to-live (TTL) value in the IP header of a tunnel packet.

Syntax:
set interfaces erspan **erspan** *tunnel-number* **ip ttl** *value*

Syntax:
delete interfaces erspan **erspan** *tunnel-number* **ip ttl** [*value*]

Syntax:
show interfaces erspan **erspan** *tunnel-number* **ip ttl**

The default value is 255.

erspan ***tunnel-number***
The identifier of a tunnel interface with an integer for the *tunnel-number* variable.

value
The value for the TTL field in the IP header of a tunnel packet (the carrier packet). Enter an integer from 0 through 255. When the TTL value is set to 0, a tunnel packet copies the TTL value from the packet being encapsulated (the passenger packet).

Configuration mode

```
interfaces {
    erspan erspantunnel-number {
        ip {
            ttl value
        }
    }
}
```

The TTL field of the IP header of a packet limits the lifetime of an IP packet and prevents indefinite packet looping.

Use the `set` form of this command to set the TTL value in the TTL field of the IP header for a packet that traverses a tunnel interface.

Use the `delete` form of this command to reset the TTL value to the default setting of 255.

Use the `show` form of this command to display the current TTL value in the IP header of a tunnel packet.

interfaces erspan erspan<tunnel-number> local-ip <address>

Specifies the IPv4 address for the local endpoint of an ERSPAN tunnel.

Syntax:
set interfaces erspan **erspan** *tunnel-number* **local-ip** *address*

Syntax:
delete interfaces erspan **erspan** *tunnel-number* **local-ip** [*address*]

Syntax:
show interfaces erspan **erspan** *tunnel-number* **local-ip**

erspan ***tunnel-number***
The identifier of a tunnel interface with an integer for the *tunnel-number* variable.

address

An IPv4 address to use as the tunnel endpoint on the local vRouter. The IP address must already be configured for the interface.

Configuration mode

```
interfaces {
    erspan erspantunnel-number {
        local-ip address
    }
}
```

The tunnel does not function when both the local and remote endpoints are not configured.

Use the `set` form of this command to specify the IP address to use as the local endpoint of a tunnel.

Use the `delete` form of this command to delete the local endpoint of a tunnel.

Use the `show` form of this command to display the IP address for the local endpoint of a tunnel.

interfaces erspan erspan <tunnel-number> mtu <bytes>

Sets the maximum transfer unit (MTU) size for an ERSPAN tunnel interface.

Syntax:
`set interfaces erspan` **`erspan`** *tunnel-number* **`mtu`** *bytes*

Syntax:
`delete interfaces erspan` **`erspan`** *tunnel-number* **`mtu`** [*bytes*]

Syntax:
`show interfaces erspan` erspan **`mtu`**

The default MTU size of 1500.

erspan *tunnel-number*

The identifier of a tunnel interface with an integer for the *tunnel-number* variable.

bytes

The MTU size in bytes for the ERSPAN tunnel interface. Enter a value from 68 through 8024. The default size is 1500.

Configuration mode

```
interfaces {
    erspan erspantunnel-number {
        mtu bytes
    }
}
```

Use this command to set the size of the maximum transfer unit (MTU) for encapsulated packets that traverse a tunnel.

The ERSPAN specification does not support packet fragmentation. Packets larger than the MTU are dropped rather than truncated. To avoid dropping packets, configure a large MTU value.

Use the `set` form of this command to set the MTU size for encapsulated packets traversing the tunnel.

Use the `delete` form of this command to reset the MTU size to its default setting of 1500 bytes.

Use the `show` form of this command to display the MTU size for encapsulated packets.

interfaces erspan erspan<tunnel-number> remote-ip <address>

Sets the IPv4 address for the remote endpoint of an ERSPAN tunnel.

Syntax:

```
set interfaces erspan erspan tunnel-number remote-ip address
```

Syntax:

```
delete interfaces erspan erspan tunnel-number remote-ip [ address ]
```

Syntax:

```
show interfaces erspan erspan tunnel-number remote-ip
```

erspan ***tunnel-number***

The identifier of a tunnel interface with an integer for the *tunnel-number* variable.

address

An IPv4 address to use as the tunnel endpoint on the remote vRouter. The IP address must already be configured for the interface.

Configuration mode

```
interfaces {
    erspan erspantunnel-number {
        remote-ip address
    }
}
```

Note that the tunnel cannot be established when both the local and remote endpoints are not configured.

Use the `set` form of this command to set the IP address to use as the remote endpoint of a tunnel.

Use the `delete` form of this command to delete the remote endpoint of a tunnel.

Use the `show` form of this command to display the IP address for the remote endpoint of a tunnel.

service portmonitor session <id> description <string>

Specifies a description for the port-monitoring session.

Syntax:

```
set service portmonitor session id description string
```

Syntax:

```
delete service portmonitor session id description
```

Syntax:

```
show service portmonitor session id description
```

id

The number of the port-monitoring session.

string

A brief description of the session. If the description contains spaces, it must be enclosed in double quotation marks.

Configuration mode

```
service {
    portmonitor {
        session id {
            description string
        }
    }
}
```

Providing a description for a session can help you to quickly determine its purpose when viewing the configuration.

Use the `set` form of the command to specify a description for the port-monitoring session.

Use the `delete` form of the command to delete the description for the port-monitoring session.

Use the `show` form of the command to show the description for the port-monitoring session.

service portmonitor session <id> destination <interface>

Specifies the destination interface or tunnel for a port monitoring session.

Syntax:

`set service portmonitor session` *id* **`destination`** { *interface-name* | *interface-name* **`.`** *vid* | **`erspan`** *tunnel-number* }

Syntax:

`delete service portmonitor session` *id* **`destination`**

Syntax:

`show service portmonitor session` *id* **`destination`** [*interface-name* | *interface-name* **`.`** *vid* | **`erspan`** *tunnel-number*]

id

The number of a port monitoring session.

interface-name

The name of a data plane interface. Following are the supported formats of the interface name:

- **`dp`** *x* **`p`** *y* **`p`** *z*—The name of a data plane interface, where
  - **`dp`** *x* specifies the data plane identifier (ID). Currently, only dp0 is supported.
  - **`p`** *y* specifies a physical or virtual PCI slot index
  - **`p`** *z* specifies a port index (for example, p1).

  For example, dp0p1p2, dp0p160p1, and dp0p192p1.
- **`dp`** *x* **`em`** *y*—The name of a data plane interface on a LAN-on-motherboard (LOM) device that does not have a PCI slot, where **`em`** *y* specifies an embedded network interface number (typically, a small number). For example, dp0em3.
- **`dp`** *x* **`s`** *y*—The name of a data plane interface on a device that is installed on a virtual PCI slot, where *x* **`s`** *y* specifies an embedded network interface number (typically, a small number). For example, dp0s2. Currently, this format applies only when using the KVM platform.
- **`dp`** *x* **`P`** *n* **`p`** *y* **`p`** *z*—The name of a data plane interface on a device that is installed on a secondary PCI bus, where **`P`** *n* specifies the bus number. You can use this format to name data plane interfaces on large physical devices with multiple PCI buses. For these devices, it is possible to have network interface cards installed on different buses with these cards having the same slot ID. The value of *n* must be an integer greater than 0. For example, dp0P1p162p1 and dp0P2p162p1.

interface-name* `.` *vid

For an RSPAN-source session only, specifies the data plane interface and VLAN number.

`erspan` ***tunnel-number***

For ERSPAN-source only, specifies the ERSPAN tunnel number.

Configuration mode

```
service {
    portmonitor {
        session id {
            destination interface-name|interface-name.vid|erspantunnel-number
        }
    }
}
```

The destination interface for an ERSPAN-source session is an ERSPAN tunnel interface.

The destination interface is dedicated to monitoring and transmits only mirrored packets.

The destination interface for a SPAN, an RSPAN-destination, or an ERSPAN-destination session is a physical interface.

The destination interface for an RSPAN-source session is a data plane interface with a VLAN number.

Only one destination interface can be configured for SPAN, RSPAN-source, RSPAN-destination, ERSPAN-source, and ERSPAN-destination in a monitoring session.

An interface cannot be both the source and destination of a monitoring session.

A destination interfaces cannot be a member of more than one monitoring session.

Use the `set` form of the command to specify the destination interface for a port monitoring session.

Use the `delete` form of the command to remove the destination interface for a port monitoring session.

Use the `show` form of the command to show the destination interface for a port monitoring session.

service portmonitor session <id> disable

Disables the port-monitoring session.

Syntax:
`set service portmonitor session` *id* **`disable`**

Syntax:
`delete service portmonitor session` *id* **`disable`**

Syntax:
`show service portmonitor session` *id*

id

The number of the port-monitoring session.

By default, the session is enabled if the session type, source interface, and destination interface are configured correctly.

Configuration mode

```
service {
    portmonitor {
        session id {
            disable
        }
    }
}
```

Use the `set` form of the command to disable the port-monitoring session.

Use the `delete` form of the command to re-enable the description for the port-monitoring session.

Use the `show` form of the command to show whether the port-monitoring session is disabled.

service portmonitor session <id> erspan header

Defines the ERSPAN header type for a port monitoring session.

Syntax:
`set service portmonitor session` *id* **`erspan header`** { **`type-II`** | **`type-III`** }

Syntax:
`delete service portmonitor session` *id* **`erspan header`**

Syntax:
`show service portmonitor session` *id* **`erspan`** [**`header`**]

id
: The number of a session. The maximum number of sessions on the router is eight. Enter an integer from 1 to 31.

`type-II`
: Defines the ERSPAN Type II encapsulation header type that is added to the original frame.

`type-III`
: Defines the ERSPAN Type III encapsulation header type that is added to the original frame. Compared to a TYPE II header, this header is larger and more flexible to support additional fields, including time stamps.

Configuration mode

```
service {
    portmonitor {
        session id {
            erspan {
                header type-II | type-III
            }
        }
    }
}
```

On the source vRouter, traffic from source interfaces is mirrored and prepended by Type II or Type III headers. These headers are defined in the IETF draft proposal.

https://tools.ietf.org/html/draft-foschiano-erspan-00

Port mirroring is enabled when the session type and the destination interface for the session are configured. For ERSPAN sessions, the ERSPAN header type and identifier must also be configured.

Use the `set` form of the command to define the ERSPAN header type.

Use the `delete` form of the command to delete the ERSPAN header type.

Use the `show` form of the command to show the ERSPAN header type for a port monitoring session.

service portmonitor session <id> erspan identifier

Defines the ERSPAN identifier for a port monitoring session.

Syntax:
`set service portmonitor session` *id* **`erspan identifier`** *erspan-id*

Syntax:
`delete service portmonitor session` *id* **`erspan identifier`** *erspan-id*

Syntax:

```
show service portmonitor session id erspan identifier erspan-id
```

id

The number of a session. The maximum number of sessions on the router is eight. Enter an integer from 1 to 31.

erspan-id

The ERSPAN identifier. Enter an integer from 1 to 1023.

Configuration mode

```
service {
    portmonitor {
        session id {
            erspan {
                identifier erspan-id
            }
        }
    }
}
```

Port mirroring is enabled when the session type and the destination interface for the session are configured. For ERSPAN sessions, the ERSPAN header type and identifier must also be configured.

Use the `set` form of the command to define the ERSPAN identifier.

Use the `delete` form of the command to delete the ERSPAN identifier.

Use the `show` form of the command to show the ERSPAN identifier for a port monitoring session.

service portmonitor session <id> filter

Specifies a filter for a port monitoring session.

Syntax:

```
set service portmonitor session id filter { in | out } filter-name
```

Syntax:

```
delete service portmonitor session id filter { in | out } filter-name
```

Syntax:

```
show service portmonitor session id filter
```

id

The number of a session. The maximum number of sessions on the router is eight. Enter an integer from 1 to 31.

`filter-name`

The name of the firewall instance that defines the portmonitor filter rules.

Configuration mode

```
service {
    portmonitor {
       session id {
       filter in string
       filter out string
       }
    }
}
```

Each filter mirrors selected traffic based on the specified rules. "in" filters apply to ingress traffic, and "out" filters apply to egress traffic.

Use the `set` form of the command to specify an ingress or egress filter for the port monitoring session.

Use the `delete` form of the command to delete an ingress or egress filter for the port monitoring session.

Use the `show` form of the command to show the filters defined for the port monitoring session.

The following limitations apply to port monitor filter rules:

- Only stateless IPv4 packet filters are supported.
- Rules can match source IP address, destination IP address, source port, destination port, IP protocol, or DSCP.

service portmonitor session <id> source <interface>

Specifies the source interface for a port monitoring session.

Syntax:

```
set service portmonitor session id source { interface-name | interface-name . vid | erspan tunnel-number
}
```

Syntax:

```
delete service portmonitor session id source { interface-name | interface-name . vid | erspan tunnel-
number }
```

Syntax:

```
show service portmonitor session id source
```

id

The number of a port monitoring session.

interface-name

The name of a data plane interface. Following are the supported formats of the interface name:

- **dp***x* **p***y* **p***z*—The name of a data plane interface, where
  - **dp***x* specifies the data plane identifier (ID). Currently, only dp0 is supported.
  - **p***y* specifies a physical or virtual PCI slot index
  - **p***z* specifies a port index (for example, p1).

  For example, dp0p1p2, dp0p160p1, and dp0p192p1.
- **dp***x* **em***y*—The name of a data plane interface on a LAN-on-motherboard (LOM) device that does not have a PCI slot, where **em***y* specifies an embedded network interface number (typically, a small number). For example, dp0em3.
- **dp***x* **s***y*—The name of a data plane interface on a device that is installed on a virtual PCI slot, where *x* **s***y* specifies an embedded network interface number (typically, a small number). For example, dp0s2. Currently, this format applies only when using the KVM platform.
- **dp***x* **P***n* **p***y* **p***z*—The name of a data plane interface on a device that is installed on a secondary PCI bus, where **P***n* specifies the bus number. You can use this format to name data plane interfaces on large physical devices with multiple PCI buses. For these devices, it is possible to have network interface cards installed on different buses with these cards having the same slot ID. The value of *n* must be an integer greater than 0. For example, dp0P1p162p1 and dp0P2p162p1.

interface-name* . *vid

Specifies the data plane interface and VLAN number.

erspan *tunnel-number*

For ERSPAN-destination only, specifies the ERSPAN tunnel number.

Configuration mode

```
service {
    portmonitor {
        session id {
            source interface-name|interface-name.vid|erspantunnel-number
        }
    }
}
```

For a SPAN or an RSPAN-source session, you can configure more than one source interface for which ingress or egress mirroring is enabled. The source interface can be a physical or VLAN interface.

For an RSPAN-destination session, you can configure one source VLAN interface from which to receive the mirrored packets. The source VIF for the RSPAN-destination type must be part of a bridge group.

The source interface for an ERSPAN-destination is an ERSPAN GRE tunnel.

An interface cannot be both the source and destination of a monitoring session.

A source interface cannot be a member of more than one monitoring session.

Use the `set` form of the command to specify the source interface for a port monitoring session.

Use the `delete` form of the command to remove the source interface for a port monitoring session.

Use the `show` form of the command to show the source interface for a port monitoring session.

service portmonitor session <id> source <interface> direction <direction>

Specifies the direction of a port monitoring for a physical source interface of a SPAN, or an RSPAN-source session.

Syntax:
`set service portmonitor session` *id* **`source`** { *interface-name* | *interface-name* **`.`** *vid* } **`direction`** { **`both`** | **`rx`** | **`tx`** }

Syntax:
`delete service portmonitor session` *id* **`source`** { *interface-name* | *interface-name* **`.`** *vid* } **`direction`**

Syntax:
`show service portmonitor session` *id* **`source`** { *interface-name* | *interface-name* **`.`** *vid* } **`direction`**

id

The number of a port monitoring session.

interface-name

The name of a data plane interface. Following are the supported formats of the interface name:

- **dp** *x* **p** *y* **p** *z*—The name of a data plane interface, where
  - **dp** *x* specifies the data plane identifier (ID). Currently, only dp0 is supported.
  - **p** *y* specifies a physical or virtual PCI slot index
  - **p** *z* specifies a port index (for example, p1).

  For example, dp0p1p2, dp0p160p1, and dp0p192p1.
- **dp** *x* **em** *y*—The name of a data plane interface on a LAN-on-motherboard (LOM) device that does not have a PCI slot, where **em** *y* specifies an embedded network interface number (typically, a small number). For example, dp0em3.
- **dp** *x* **s** *y*—The name of a data plane interface on a device that is installed on a virtual PCI slot, where *x* **s** *y* specifies an embedded network interface number (typically, a small number). For example, dp0s2. Currently, this format applies only when using the KVM platform.
- **dp** *x* **P** *n* **p** *y* **p** *z*—The name of a data plane interface on a device that is installed on a secondary PCI bus, where **P** *n* specifies the bus number. You can use this format to name data plane interfaces on large physical devices with multiple PCI buses. For these devices, it is possible to have network

interface cards installed on different buses with these cards having the same slot ID. The value of *n* must be an integer greater than 0. For example, dp0P1p162p1 and dp0P2p162p1.

interface-name . vid
Specifies the data plane interface and VLAN number.

`both`
Specifies that the interface mirrors and monitors the session in both ingress and egress directions.

`rx`
Specifies that the interface mirrors packets and monitors the session in the ingress direction.

`tx`
Specifies that the interface mirrors packets and monitors the session in the egress direction.

The default directions are both ingress and egress.

Configuration mode

```
service {
    portmonitor {
        session id {
            source interface-name|interface-name.vid{
                direction both | rx | tx
            }
        }
    }
}
```

You cannot configure this command for an RSPAN-destination session.

Use the `set` form of the command to specify the direction that the physical source interface monitors a port monitoring session.

Use the `delete` form of the command to remove the direction that the physical source interface monitors a port monitoring session.

Use the `show` form of the command to show the direction that the physical source interface monitors a port monitoring session.

service portmonitor session <id> type <type>

Defines the identifier and type for a port monitoring session.

Syntax:
`set service portmonitor session` *id* **`type`** { **`span`** | **`rspan-source`** | **`rspan-destination`** | **`erspan-source`** | **`erspan-destination`** }

Syntax:
`delete service portmonitor session` *id* [**`type`** [**`span`** | **`rspan-source`** | **`rspan-destination`** | **`erspan-source`** | **`erspan-destination`**]]

Syntax:
`show service portmonitor session` *id* **`type`**

id
The number of a session. The maximum number of sessions on the router is eight. Enter an integer from 1 to 31.

`span`
Specifies a SPAN session type.

`rspan-source`
Specifies an RSPAN-source session type.

`rspan-destination`
Specifies an RSPAN-destination session type.

`erspan-source`

Specifies an ERSPAN-source session type.

`erspan-destination`

Specifies an ERSPAN-destination session type.

Configuration mode

```
service {
    portmonitor {
        session id {
            type span|rspan-source|rspan-destination|erspan-source|erspan-destination
        }
    }
}
```

Port mirroring is enabled when the session type and the destination interface for the session are configured. For ERSPAN sessions, the ERSPAN header type and identifier must also be configured.

Use the `set` form of the command to define a port monitoring session.

Use the `delete` form of the command to delete a port monitoring session.

Use the `show` form of the command to show the identifier and type for a port monitoring session.

show portmonitor filter

Displays information on the filters defined for port monitoring sessions.

Syntax:

`show portmonitor filter` [*interface*]

interface

The device interface.

Operational mode

Use this command to display information on the filters defined for port monitoring sessions.

The following example displays information for all of the defined port monitor filters.

```
vyatta@vyatta:~$ show portmonitor filter
---------------------------------
Rulesets Information: Portmonitor
--------------------------------
--------------------------------------------------------------------------------
Portmonitor Inbound Filter Rules "dev":
Active on (dp0s4, in)
rule    action  proto           packets         bytes
----    ------  -----           -------         -----
2       allow   any             0               0
  condition - to 31.1.1.1
20      drop    any             2054            107173
  condition - all
Portmonitor Inbound Filter Rules "dev":
Active on (dp0s5, in)
rule    action  proto           packets         bytes
----    ------  -----           -------         -----
2       allow   any             0               0
  condition - to 31.1.1.1
20      drop    any             2126            110807
  condition - all
---------------------------------
Rulesets Information: Portmonitor
--------------------------------
--------------------------------------------------------------------------------
Portmonitor Outbound Filter Rules "dev2":
```

```
Active on (dp0s4, out)
rule    action  proto           packets         bytes
----    ------  -----           -------         -----
2       allow   any             0               0
  condition - from 31.1.1.1
20      drop    any             0               0
  condition - all
Portmonitor Outbound Filter Rules "dev2":
Active on (dp0s5, out)
rule    action  proto           packets         bytes
----    ------  -----           -------         -----
2       allow   any             0               0
  condition - from 31.1.1.1
20      drop    any             0               0
  condition - all
```

The following example displays information for the filters defined on a specific interface (dp0s5).

```
vyatta@vyatta:~$ show portmonitor filter dp0s5
---------------------------------
Rulesets Information: Portmonitor
---------------------------------
--------------------------------------------------------------------------------
Portmonitor Inbound Filter Rules "dev":
Active on (dp0s5, in)
rule    action  proto           packets         bytes
----    ------  -----           -------         -----
2       allow   any             0               0
  condition - to 31.1.1.1
20      drop    any             2198            114551
  condition - all
---------------------------------
Rulesets Information: Portmonitor
---------------------------------
--------------------------------------------------------------------------------
Portmonitor Outbound Filter Rules "dev2":
Active on (dp0s5, out)
rule    action  proto           packets         bytes
----    ------  -----           -------         -----
2       allow   any             0               0
  condition - from 31.1.1.1
20      drop    any             0               0
  condition - all
```

show portmonitor session

Displays configuration information for a port monitoring session.

Syntax:

`show portmonitor session` [*id*]

id

The number of a port monitoring session.

Operational mode

Use this command to display configuration information for a port monitoring session.

The following example shows configuration information for an ERSPAN-destination session with ingress and egress port monitoring filters.

```
vyatta@vyatta:~$ show portmonitor session
  Session:                    2
    Type:                     erspan-source
    State:                    enabled
    erspan Identifier:        20
    erspan Header:            type-II
    Source interfaces:
      Name:                   dp0s5
        Direction:            both
    Destination interface:    erspan2
    Filters:
      Name:                   portin
        Type:                 in
      Name:                   portout
        Type:                 out
```

Related commands

The following table lists related commands that are documented elsewhere.

Related commands documented elsewhere	
`set interfaces dataplane` *interface-name* `[` **`address`** `]`	Defines the IP address that is referenced in the ERSPAN port mirroring configuration. (Refer to AT&T Vyatta Network Operating System LAN Interfaces Configuration Guide.)

sFlow

sFlow overview

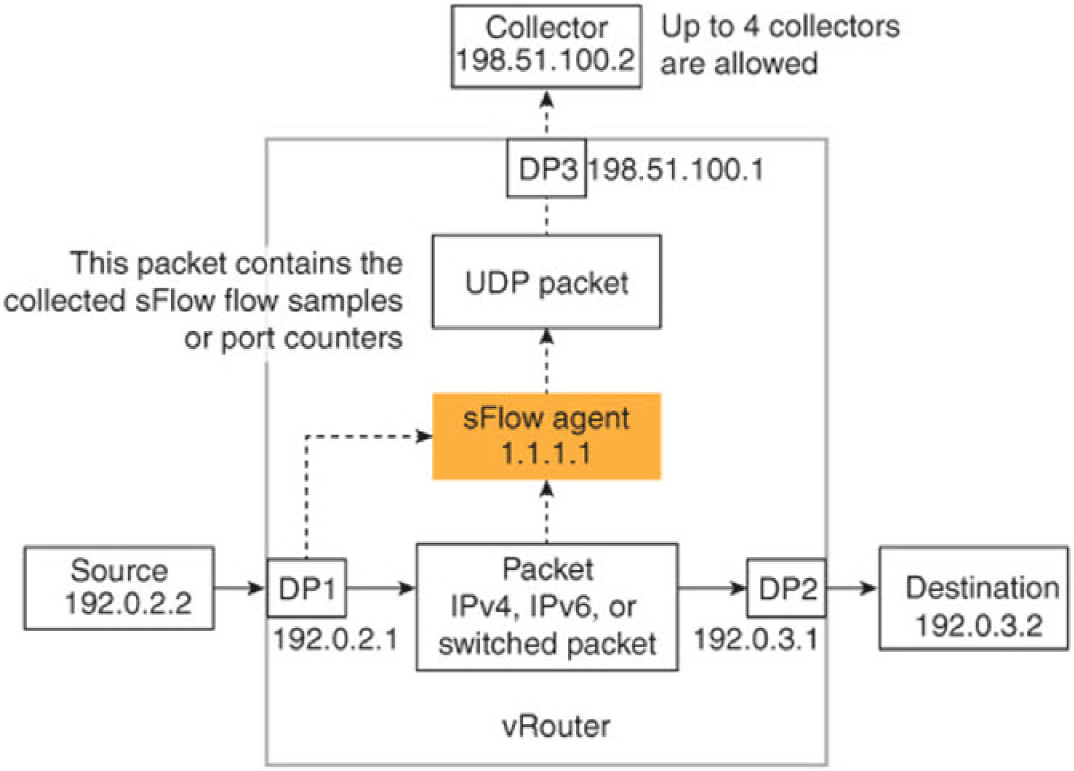
The AT&T Vyatta vRouter supports sFlow (v5), a high-performance sFlow agent for monitoring traffic in a data network. sFlow, which stands for “sampled flow,” reports packet-flow and port-counter samples from a data plane interface to receiving collectors on a timely basis.

> **Note:** In an AT&T Vyatta vRouter, sFlow supports enough functionality and flexibility such that packet-forwarding performance does not drop significantly, yet still delivers enough processing power to provide a significant advantage when processing packets through the data plane pipeline.

You can configure an AT&T Vyatta vRouter to perform the following tasks:

- Sample packet flows.
- Collect packet headers from sampled packets to gather inbound traffic information on these packets.
- Collect byte and packet port-counter statistics (counter samples).
- Compose flow sample messages from the collected information.
- Relay messages to an external device known as a collector. A AT&T Vyatta vRouter supports up to four collectors.

Figure 20: sFlow configuration example



> **Note:** The AT&T Vyatta vRouter supports sFlow (v5), which replaces the version outlined in RFC 3176.

Physical interfaces

sFlow is physical-interface-based, which means that you can enable and disable sFlow on only a physical interface. However, all other interface types (for example, VIF and tunnel) are indirectly enabled because all are associated with physical interfaces.

Packet-flow and port-counter sampling

The AT&T Vyatta vRouter acts as an sFlow agent, which collects inbound packet samples from a data plane interface at the configured sampling rate. The AT&T Vyatta vRouter also collects sample statistics from ports at the configured polling interval. The AT&T Vyatta vRouter sends the collected information to the specified collectors.

Packet-flow sampling

The AT&T Vyatta vRouter, acting as an sFlow agent, samples dropped and nondropped packets. Taking a sample involves extracting information from the packet, including trajectory information (for example, source and destination interface and next hop). For dropped packets, the AT&T Vyatta vRouter adds the reason for dropping the packet to the extracted information.

Counter-flow sampling

By default, counter-flow sampling is enabled on the AT&T Vyatta vRouter. The AT&T Vyatta vRouter, acting as an sFlow agent, sends port statistics with counter sample data in the datagram stream that results from packet-flow sampling.

Communication with collectors

The AT&T Vyatta vRouter creates one or more samples for each UDP packet. When enough samples are collected to fill a UDP datagram (up to 1,400 bytes), the AT&T Vyatta vRouter sends the UDP datagram to the collectors.

Supported sFlow types

The AT&T Vyatta vRouter supports the following sFlow data types:

Table 37: sFlow data types

Type	Description
Sampled header	Includes the protocol type, length, and packet header bytes. Packet header information for each packet must be reported using the sampled header format.
Sampled IPv4	Includes the source and destination IPv4 addresses, IPv4 packet length, protocol type, and so on.
Sampled IPv6	Includes the source and destination IPv6 addresses, IPv6 packet length, protocol type, and so on.
Sampled NAT data	Includes the source and destination IP addresses that are translated by NAT. If an address was not translated, it is equal to the address reported for the sampled header.packet length, and protocol type.

Type	Description
Counter data type	The interface port counters. Only the generic counter is supported.

Extended data types provide additional information about the sampled packet.

Table 38: Extended data types

Type	Description
Extended router	Includes the nexthop IP address and the source and destination prefixes.

Configuring sFlow

The following procedure shows how to configure sFlow for the sample AT&T Vyatta vRouter configuration. Figure 1 *(page 215)* shows this configuration.

Table 39: Configuring sFlow

Step	Command
Specify the address of the agent. By default, this collector listens for sFlow data on the 6343 UDP port. **Note:** A AT&T Vyatta vRouter supports up to four collectors.	`vyatta@sflow1# set service sflow agent-address 1.1.1.1`
Set the polling interval to three seconds. Every three seconds, the sFlow agent that is running on the AT&T Vyatta vRouter collects port counter data.	`vyatta@sflow1# set service sflow polling-interval 3`
Configure the sFlow agent to send the UDP datagrams that contain the collected sFlow information to the default port (6343) of the sFlow collector server at 198.51.100.2.	`vyatta@sflow1# set service sflow server-address 198.51.100.2 server-port 6343`
Specify 512 as the number of packets from which a sample is taken by the sFlow agent. In other words, for every 512 packets that flow through the interface, the sFlow agent selects one packet for analysis.	`vyatta@sflow1# set service sflow sampling-rate 512`
Commit the configuration.	`vyatta@R1# commit`
Save the configuration.	`vyatta@sflow1# save`

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>

Display the sFlow configuration.

The output shows that sFlow was configured, but no statistics were collected. This is because sFlow has not yet been enabled on an interface.

</td><td>

```
vyatta@sflow1# run show sflow
{
    "sFlow information": {
        "sFlow version": 5,
        "sFlow services": "enabled",
        "sFlow agent IP address": "1.1.1.1",
        "Collector destinations configured":
 1,
        "Collectors": [{
                "IP address": "198.51.100.2",
                "UDP port number": 6343
            }
        ],
        "Polling interval": 3,
        "Configured default sampling rate":
 512,
        "Actual default sampling rate": 512,
        "sFlow max-header size": 128,
        "UDP packets sent": 0,
        "Flow samples collected": 0,
        "sFlow interfaces": [],
        "Total sFlow interfaces": 0
    }
}
```

</td></tr>
<tr><td>

Enable sFlow on the dp0p192p1 interface (Figure 1 *(page 215)* indicates it as DP1).

</td><td>

```
vyatta@sflow1# set interfaces dataplane
 dp0p192p1 sflow
```

</td></tr>
<tr><td>

Commit the configuration.

</td><td>

```
vyatta@R1# commit
```

</td></tr>
<tr><td>

Save the configuration.

The sFlow agent can now start collecting packet samples and port-counter statistics.

</td><td>

```
vyatta@sflow1# save
```

</td></tr>
</table>

<table>
<tr><th>Step</th><th>Command</th></tr>
<tr><td>

Display the sFlow configuration.

The output shows that sFlow is enabled for the dp0p192p1 interface and shows that one sample packet was collected.

</td><td>

```
vyatta@sflow1# run show sflow
{
    "sFlow information": {
        "sFlow version": 5,
        "sFlow services": "enabled",
        "sFlow agent IP address": "1.1.1.1",
        "Collector destinations configured":
 1,
        "Collectors": [{
                "IP address": "198.51.100.2",
                "UDP port number": 6343
            }
        ],
        "Polling interval": 3,
        "Configured default sampling rate":
 512,
        "Actual default sampling rate": 512,
        "sFlow max-header size": 128,
        "UDP packets sent": 0,
        "Flow samples collected": 0,
        "sFlow interfaces": [{
                "name": "dp0p192p1"
            }
        ],
        "Total sFlow interfaces": 1
    }
}
```

</td></tr>
</table>

sFlow Commands

interfaces dataplane <dp-port> sflow

Specifies an interface for which to record inbound sFlow packet statistics and port counters.

Syntax:
`set interfaces dataplane` *dp-port* **`sflow`**

Syntax:
`delete interfaces dataplane` *dp-port* **`sflow`**

dp-port
The name of a data plane interface.

Configuration mode

```
interfaces {
    dataplane dp-port {
        sflow
    }
}
```

You can enable multiple interfaces by issuing this command multiple times, once for each interface.

Use the `set` form of this command to enable sFlow on an interface.

Use the `delete` form of this command to disable sFlow on an interface.

service sflow agent-address <IPv4-or-IPv6>

Specifies the IPv4 or IPv6 address (of an sFlow agent) to be included in the sFlow packets that are sent to the collectors.

Syntax:
`set service sflow agent-address` *IPv4-or-IPv6*

Syntax:
`delete service sflow agent-address` *IPv4-or-IPv6*

By default, the AT&T Vyatta vRouter provides an IP address that identifies the agent.

IPv4-or-IPv6
The IPv4 or IPv6 address of an sFlow agent.

Note: The address does not have to be active.

Configuration mode

```
service {
    sflow {
        agent-address address
    }
}
```

Use the `set` form of this command to specify the IP address of an sFlow agent.

Use the `delete` form of this command to delete the IP address of an sFlow agent.

service sflow server-address <IPv4-or-IPv6> server-port <port>

Specifies the IPv4 or IPv6 IP address of an sFlow collector and a port to which to send the UDP datagrams that contain the collected sFlow information.

Syntax:
`set service sflow server-address` *IPv4-or-IPv6* **`server-port`** *port*

Syntax:
`delete service sflow server-address` *IPv4-or-IPv6*

The default port is 6343.

IPv4-or-IPv6
The IP address of the sFlow collector server.

port
A port of an sFlow collector server to which to send the UDP datagrams that contain the collected sFlow information.

Configuration mode

```
interfaces {
    sflow {
        server-address IPv4-or-IPv6 {
            server-port port
        }
    }
}
```

You can specify multiple sFlow collectors by entering this command multiple times. The maximum number of sFlow collectors allowed is four.

Use the `set` form of this command to specify an sFlow collector and a port to which to send the collected sFlow data.

Use the `delete` form of this command to remove an sFlow collector from the list of collectors to which to send sFlow data.

service sflow sampling-rate <sampling-rate>

Specifies the rate at which packets are sampled.

Syntax:
`set service sflow sampling-rate` *sampling-rate*

Syntax:
`delete service sflow sampling-rate` *sampling-rate*

The default sampling rate is 2048 packets.

sampling-rate
The number of packets from which to pick one packet for analysis. The number ranges from 512 through 65535. For example, if a number of 512, it means that the sampling rate is 1 in 512. In other words, the AT&T Vyatta vRouter selects a packet every other 512 packets.

Configuration mode

```
service {
    sflow {
        sampling-rate sampling-rate
    }
}
```

Use the `set` form of this command to specify the sampling rate.

Use the `delete` form of this command to restore the default sampling rate, which is 2,048 packets.

service sflow polling-interval <polling-rate>

Specifies how often sFlow port-counter statistics are collected.

Syntax:
`set service sflow polling-interval` *polling-rate*

Syntax:
`delete service sflow polling-interval` *polling-rate*

The default polling rate is 20 seconds.

polling-rate

A polling rate in seconds. The rate ranges from 0 through 65535. A rate of 0 means that counter polling is disabled.

Configuration mode

```
service {
    sflow {
        polling-interval polling-rate
    }
}
```

Use the `set` form of this command to specify a polling rate.

Use the `delete` form of this command to restore the default polling rate, which is 20 seconds.

show sflow

Displays sFlow configuration information and sFlow usage statistics.

Syntax:
`show sflow`

Operational mode

The following example shows how to display sFlow configuration information and usage statistics.

```
vyatta@sflow1# run sh sflow
{
    "sFlow information": {
        "sFlow version": 5,
        "sFlow services": "enabled",
        "sFlow agent IP address": "1.1.1.1",
        "Collector destinations configured": 1,
        "Collectors": [{
                "IP address": "50.1.1.2",
                "UDP port number": 6343
            }
        ],
        "Polling interval": 3,
        "Configured default sampling rate": 512,
        "Actual default sampling rate": 512,
        "sFlow max-header size": 128,
        "UDP packets sent": 0,
        "Flow samples collected": 0,
        "sFlow interfaces": [],
        "Total sFlow interfaces": 0
    }
    }
```

clear sflow

Clears the collected sFlow statistics.

Syntax:

`clear sflow`

Operational mode

The following example shows how to clear the collected sFlow statistics (the number of UDP packets sent and sFlow flow samples collected).

```
vyatta@vyatta# run clear sflow
[edit]
vyatta@vyatta# run show sflow
{
    "sFlow information": {
        "sFlow version": 5,
        "sFlow services": "enabled",
        "sFlow agent IP address": "1.1.1.1",
        "Collector destinations configured": 1,
        "Collectors": [{
                "IP address": "198.51.100.2",
                "UDP port number": 6343
            }
        ],
        "Polling interval": 3,
        "Configured default sampling rate": 512,
        "Actual default sampling rate": 512,
        "sFlow max-header size": 128,
        "UDP packets sent": 0,
        "Flow samples collected": 0,
        "sFlow interfaces": [{
                "name": "dp0s160"
            }
        ],
        "Total sFlow interfaces": 1
    }
}
```

TWAMP

TWAMP overview

A Two-Way Active Measurement Protocol (TWAMP) server on an AT&T Vyatta vRouter measures round-trip IP performance between any two devices in a network that supports the standard. The TWAMP server implementation is based on the specifications outlined in RFC 5357.

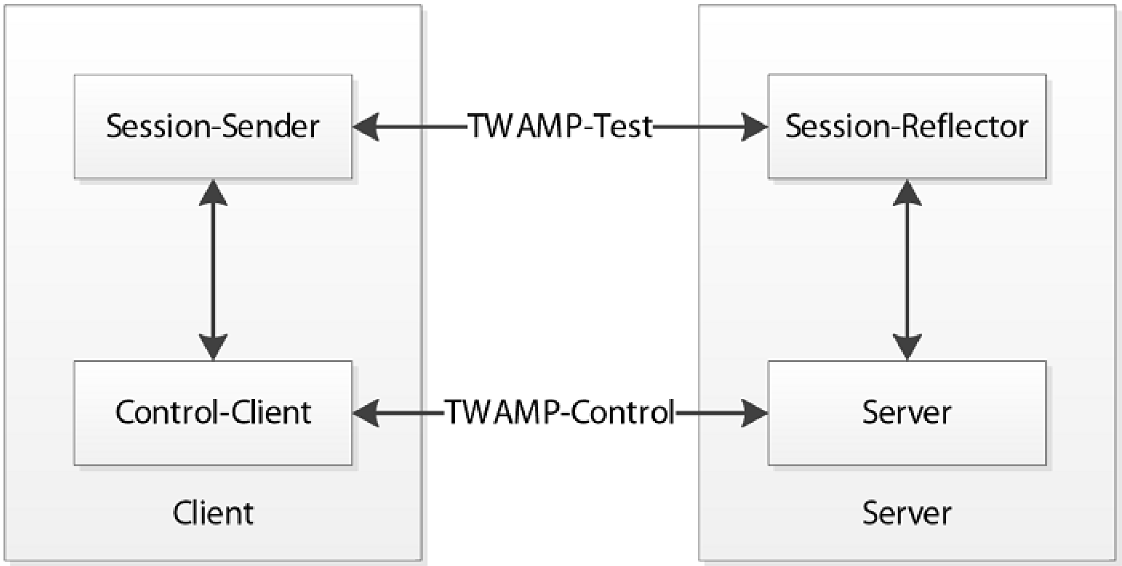
The architecture of the AT&T Vyatta vRouter TWAMP server solution defines the following logical components, as shown in the following figure.

- Session-Reflector—Creates and sends measurement packets when it receives a TWAMP-test packet.
- Server—Manages multiple TWAMP sessions.

The following components are part of the TWAMP client:

- Session-Sender—Creates and sends TWAMP-test packets to Session-Reflector.
- Control-Client—Sends requests to the TWAMP server to measure IP performance.

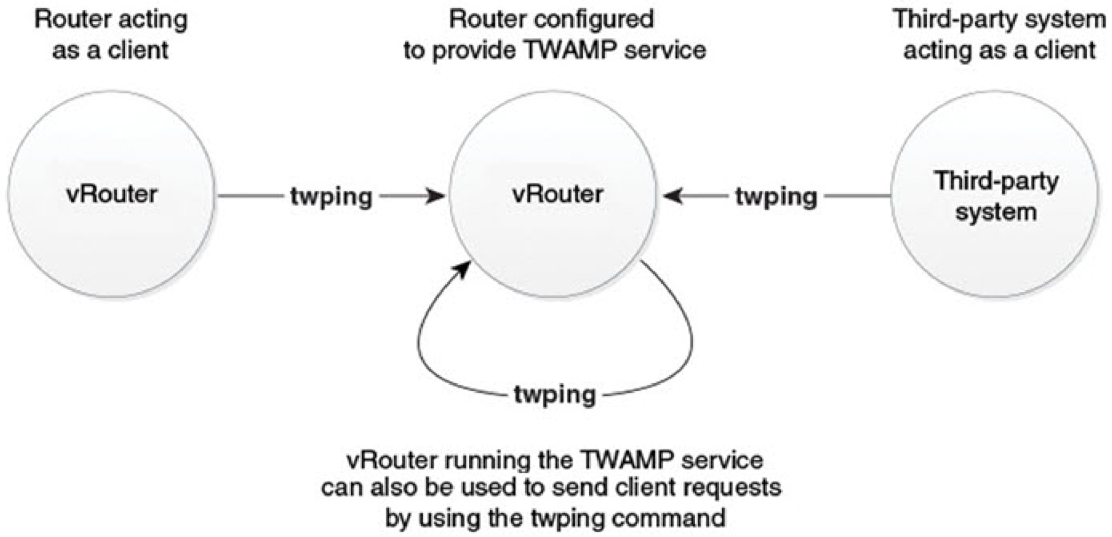
Figure 21: TWAMP architecture



A AT&T Vyatta vRouter includes the `twping` command in the /opt/vyatta/bin/twping directory, which you can use to send client requests to a TWAMP sever.

A AT&T Vyatta vRouter can be a TWAMP server and client at the same time. However, the client can be another AT&T Vyatta vRouter or a third-party system, as shown in the following figure.

Figure 22: TWAMP server-client interaction


Router acting as a client
Router configured to provide TWAMP service
Third-party system acting as a client
vRouter
twping
vRouter
twping
Third-party system
twping
vRouter running the TWAMP service can also be used to send client requests by using the twping command

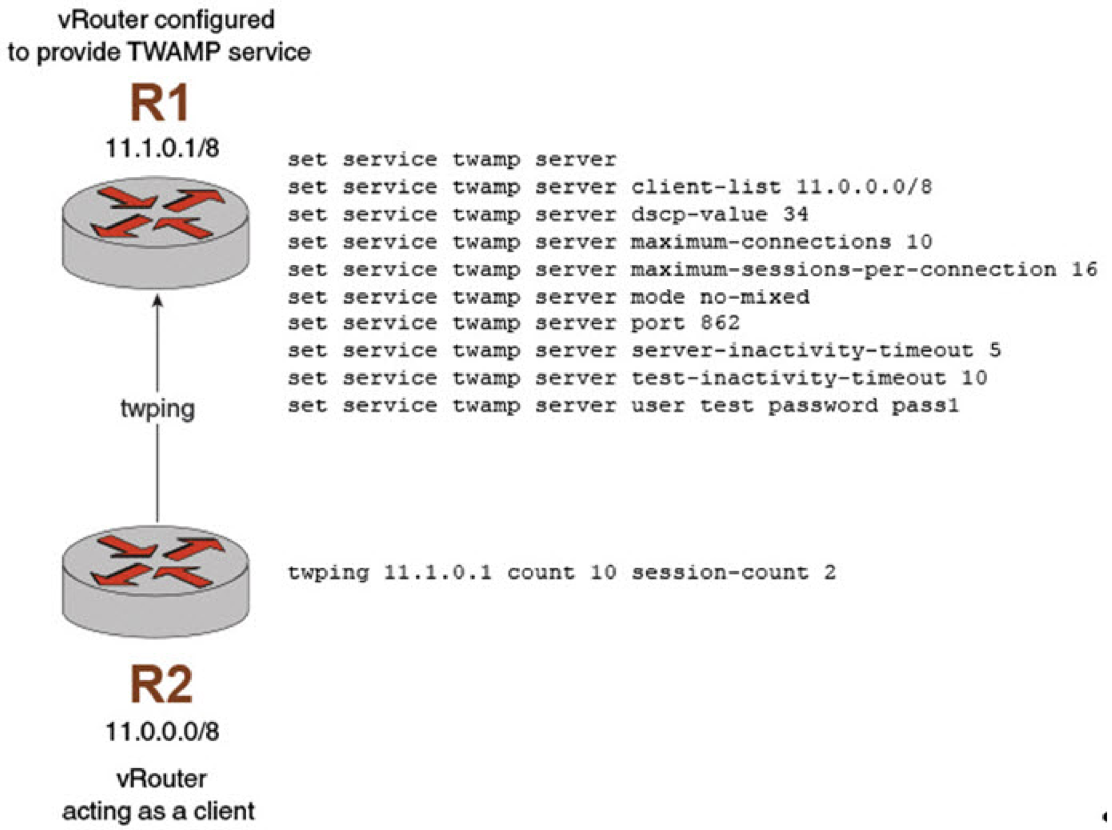
TWAMP configuration

This section includes the following examples:

- Configuring the TWAMP server *(page 226)*
- Using twping to measure IP performance *(page 228)*

Figure 1 *(page 226)* shows the TWAMP client/server topology used in the following examples.

Figure 23: TWAMP server configuration example



```
set service twamp server
set service twamp server client-list 11.0.0.0/8
set service twamp server dscp-value 34
set service twamp server maximum-connections 10
set service twamp server maximum-sessions-per-connection 16
set service twamp server mode no-mixed
set service twamp server port 862
set service twamp server server-inactivity-timeout 5
set service twamp server test-inactivity-timeout 10
set service twamp server user test password pass1
```

```
twping 11.1.0.1 count 10 session-count 2
```

Configuring the TWAMP server

To use the TWAMP server on an AT&T Vyatta vRouter, you must first configure the TWAMP service.

To configure the TWAMP service, perform the following steps in configuration mode.

Table 40: Configuring the TWAMP service

Step	Command
Start the TWAMP server. By default, the server accepts any connection request from any client.	`vyatta@R1# set service twamp server`
Add the client with the 11.0.0.0/8 IP address to the list of clients that can connect to the TWAMP server in the unauthenticated mode.	`vyatta@R1# set service twamp server client-list 11.0.0.0/8`
Specify 34 as the base-10 value of the DSCP byte in the IP header of control packets sent from the server.	`vyatta@R1# set service twamp server dscp-value 34`
Specify 10 as the maximum number of control sessions for each TWAMP server.	`vyatta@R1# set service twamp server maximum-connections 10`

Step	Command
Limit the number of maximum number of test sessions for each control session to 16.	`vyatta@R1# set service twamp server maximum-sessions-per-connection 16`
Disable the mixed client authentication mode.	`vyatta@R1# set service twamp server mode no-mixed`
Specify 862 as the TCP port used for control sessions.	`vyatta@R1# set service twamp server port 862`
Set the timeout value for control-session inactivity to 5 seconds.	`vyatta@R1# set service twamp server server-inactivity-timeout 5`
Set the timeout value for test-session inactivity to 10 seconds.	`vyatta@R1# set service twamp server test-inactivity-timeout 10`
Create the test user account and assign a password to it.	`vyatta@R1# set service twamp server user test password pass1`
Commit the configuration.	`vyatta@R1# commit`
Exit the configuration mode.	`vyatta@R1# exit`

Show the status of the server.	`vyatta@R1:~$ show service twamp server status`
	`TWAMP Server is configured and is running.`

Step	Command
Show TWAMP server session activity.	

```
vyatta@R1:~$ show service twamp server sessions all
Total number of sessions: 16
Total number of active sessions: 16
--> Control Session initiated by [Rtr2-tap0]:39549 in Authenticated mode
SID                              SENDER               REFLECTOR            STATUS  DSCP
0a00020fd863fa0eeecc1444b5000a61 [Rtr2-tap0]:8833    [Rtr1]:56967         ACTIVE   0x0
0a00020fd863fa0eee08638005774d72 [Rtr2-tap0]:8879    [Rtr1]:38676         ACTIVE   0x0
0a00020fd863fa0eecd9f0a68f69a638 [Rtr2-tap0]:8935    [Rtr1]:46080         ACTIVE   0x0
0a00020fd863fa0eeb2dba5175914887 [Rtr2-tap0]:8859    [Rtr1]:60355         ACTIVE   0x0
0a00020fd863fa0ee5a9b06cff9bacf1 [Rtr2-tap0]:8769    [Rtr1]:56089         ACTIVE   0x0
0a00020fd863fa0ee4b11c71bcba56af [Rtr2-tap0]:8778    [Rtr1]:48298         ACTIVE   0x0
0a00020fd863fa0ee3f7c679e973a4f4 [Rtr2-tap0]:8917    [Rtr1]:36039         ACTIVE   0x0
0a00020fd863fa0ee31e6473f8fc1a74 [Rtr2-tap0]:8845    [Rtr1]:45534         ACTIVE   0x0
<--
--> Control Session initiated by [Rtr2-tap0]:39550 in Authenticated mode
SID                              SENDER               REFLECTOR            STATUS  DSCP
0a00020fd863fa1a1f73b8df73463109 [Rtr2-tap0]:8958    [Rtr1]:54254         ACTIVE   0x0
0a00020fd863fa1a1d4dbe795b7d70d7 [Rtr2-tap0]:8949    [Rtr1]:58983         ACTIVE   0x0
0a00020fd863fa1a1bd7fe03660ed58a [Rtr2-tap0]:8817    [Rtr1]:49623         ACTIVE   0x0
0a00020fd863fa1a1a223635b8669238 [Rtr2-tap0]:8871    [Rtr1]:55287         ACTIVE   0x0
0a00020fd863fa1a18d1f250ed2c2c23 [Rtr2-tap0]:8863    [Rtr1]:57902         ACTIVE   0x0
0a00020fd863fa1a17756d14abae00a0 [Rtr2-tap0]:8831    [Rtr1]:37197         ACTIVE   0x0
0a00020fd863fa1a1696dda7c4c0d57b [Rtr2-tap0]:8800    [Rtr1]:54229         ACTIVE   0x0
0a00020fd863fa1a1594751787e08a1f [Rtr2-tap0]:8779    [Rtr1]:53110         ACTIVE   0x0
<--
```

Using twping to measure IP performance

To measure round-trip IP Performance, perform the following steps in operational mode.

Table 41: Measuring round-trip IP performance by using twping

Step	Command

Use the `twping` command to send a request to the TWAMP server to measure the round-trip IP performance.	`vyatta@R2:~$ twping 11.1.0.1 count 10 session-count 2`
	`Approximately 4.2 seconds until results available`
	
	`--- twping statistics from [11.1.0.1]:8889 to [Rtr2]:55026 ---`
	`SID:    0b010001d961f8d25daa301413dddec8`
	
	`first:  2015-07-28T12:49:23.476`
	
	`last:   2015-07-28T12:49:24.605`
	
	`10 sent, 0 lost (0.000%), 0 send duplicates, 0 reflect duplicates`
	`round-trip time min/median/max = 0.746/1.5/2.39 ms, (err=1.11 ms)`
	`send time min/median/max = 0.196/0.6/1.54 ms, (err=0.556 ms)`
	`reflect time min/median/max = 0.543/0.9/1.79 ms, (err=0.556 ms)`
	`reflector processing time min/max = 0.0129/0.0277 ms`
	`two-way PDV = 0.9 ms (P95-P50)`
	
	`send PDV = 1 ms (P95-P50)`
	
	`reflect PDV = 0.9 ms (P95-P50)`
	
	`send hops = 0 (consistently)`
	
	`reflect hops = 0 (consistently)`
	
	`--- twping statistics from [11.1.0.1]:8904 to [Rtr2]:42203 ---`
	`SID:    0b010001d961f8d25eff1b28f68b20f7`
	
	`first:  2015-07-28T12:49:23.495`
	
	`last:   2015-07-28T12:49:24.940`
	
	`10 sent, 0 lost (0.000%), 0 send duplicates, 0 reflect duplicates`
	`round-trip time min/median/max = 1.24/1.6/3.24 ms, (err=1.11 ms)`
	`send time min/median/max = 0.37/0.7/2.52 ms, (err=0.556 ms)`
	`reflect time min/median/max = 0.713/0.9/1.51 ms, (err=0.556 ms)`
	`reflector processing time min/max = 0.0157/0.031 ms`
	`two-way PDV = 1.7 ms (P95-P50)`
	
	`send PDV = 1.9 ms (P95-P50)`
	
	`reflect PDV = 0.7 ms (P95-P50)`
	
	`send hops = 0 (consistently)`
	
	`reflect hops = 0 (consistently)`

TWAMP Commands

service twamp server

Starts the TWAMP server.

Syntax:

```
set service twamp server
```

Syntax:

```
delete service twamp server
```

Syntax:

```
show service twamp server
```

Configuration mode

```
service {
   twamp {
       server
   }
}
```

If other server parameters are not configured, this command configures the TWAMP server with the default values.

Use the `set` form of this command to start the TWAMP server.

Use the `delete` form of this command to stop the TWAMP server.

Use the `show` form of this command to view the configuration parameters of the TWAMP server.

service twamp server client-list <ip-address>

Adds a client in a specified network to the list of clients that can connect to the TWAMP server in the unauthenticated mode.

Syntax:

`set service twamp server client-list` *ip-address*

Syntax:

`delete service twamp server client-list` *ip-address*

ip-address

IPv4 or IPv6 IP address of the system to add to the list of clients that can connect to the server. To specify multiple systems, use a subnet range.

Configuration mode

```
service {
    twamp {
        server {
            client-list ip-address
        }
    }
}
```

By default, any client can connect to the server. However, after you configure a client list, only those clients in the list can connect in the unauthenticated mode. Clients that are not in this list can connect to the server in the authenticated, encrypted, or mixed mode.

Use the `set` form of this command to add a client to the list of clients that can connect to the server in the unauthenticated mode. You can use this form multiple times to add multiple clients to the list.

Use the `delete` form of this command to delete a client from the list.

service twamp server dscp-value <value>

Specifies the base-10 value of the DSCP byte in the IP header of control packets sent from the server.

Syntax:
`set service twamp server dscp-value` *value*

Syntax:
`delete service twamp server dscp-value` *value*

0.

value

Base-10 value of the DSCP byte.

Configuration mode

```
service {
    twamp {
        server {
            dscp-value value
        }
    }
}
```

Use the `set` form of this command to set the DSCP value.

Use the `delete` form of this command to set the DSCP value to the default value.

service twamp server maximum-connections <count>

Specifies the maximum number of control sessions for each TWAMP server.

Syntax:
`set service twamp server maximum-connections` *count*

Syntax:
`delete service twamp server maximum-connections` *count*

16.

count

Maximum number of control sessions for each TWAMP server. The count ranges from 1 through 64.

Configuration mode

```
service {
    twamp {
        server {
            maximum-connections count
        }
    }
}
```

Use the `set` form of this command to specify the maximum number of control sessions for each TWAMP server.

Use the `delete` form of this command to reset the maximum number of control sessions to the default count, which is 16.

service twamp server maximum-sessions-per-connection <count>

Specifies the maximum number of test sessions for each control session.

Syntax:
`set service twamp server maximum-sessions-per-connection` *count*

Syntax:
`delete service twamp server maximum-sessions-per-connection` *count*

8.

count
Maximum number of test sessions for each control session. The count ranges from 1 through 64.

Configuration mode

```
service {
    twamp {
        server {
            maximum-sessions-per-connection count
        }
    }
}
```

Use the `set` form of this command to specify the maximum number of test sessions for each control session.

Use the `delete` form of this command to reset the maximum number of test sessions to the default count, which is 8.

service twamp server mode <authentication-mode>

Disables a client authentication mode.

Syntax:
`set service twamp server mode` *authentication-mode*

Syntax:
`delete service twamp server mode` *authentication-mode*

By default, the following modes are enabled: unauthenticated, authenticated, encrypted, and mixed.

authentication-mode
One of the following TWAMP authentication modes:

- **`no-authenticated`** —Disables support for authenticated sessions.
- **`no-encrypted`** —Disables support for encrypted sessions.
- **`no-mixed`** —Disables support for mixed mode sessions.
- **`no-unauthenticated`** —Disables support for unauthenticated sessions.

Configuration mode

```
service {
    twamp {
        server {
            mode authentication-mode
        }
    }
}
```

Use the `set` form of this command to disable a client-authentication mode.

Use the `delete` form of this command to enable a client-authentication mode.

service twamp server port <port-number>

Specifies the TCP port for a control session.

Syntax:
`set service twamp server port` *port-number*

Syntax:
`delete service twamp server port` *port-number*

862.

port-number
A TCP port number. The port number ranges from 1 through 65535.

Configuration mode

```
service {
    twamp {
        server {
            port port-number
        }
    }
}
```

Use the `set` form of this command to specify the TCP port for a control session.

Use the `delete` form of this command to reset the TCP port number to the default port number, which is 862.

service twamp server server-inactivity-timeout <seconds>

Specifies the timeout value for control-session inactivity.

Syntax:
`set service twamp server server-inactivity-timeout` *seconds*

Syntax:
`delete service twamp server server-inactivity-timeout` *seconds*

900.

seconds
Number of seconds before a control session times out due to inactivity. The number of seconds ranges from 1 through 3600.

> **Note:** The inactivity timer starts only when no test sessions are active on the associated control session.

Configuration mode

```
service {
    twamp {
        server {
            server-inactivity-timeout seconds
        }
    }
}
```

Use the `set` form of this command to specify the control-session timeout value.

Use the `delete` form of this command to reset the timeout value to the default number of seconds, which is 900.

service twamp server test-inactivity-timeout <seconds>

Specifies the timeout value for test-session inactivity.

Syntax:
`set service twamp server test-inactivity-timeout` *seconds*

Syntax:
`delete service twamp server test-inactivity-timeout` *seconds*

900.

seconds
Number of seconds before a test session times out due to inactivity. The number of seconds ranges from 1 through 3600.

Configuration mode

```
service {
    twamp {
        server {
            test-inactivity-timeout seconds
        }
    }
}
```

Use the `set` form of this command to specify the test-session timeout value.

Use the `delete` form of this command to reset the timeout value to the default number of seconds, which is 900.

service twamp server user <username> password <pwd>

Configures a username and password for use with authenticated, encrypted, and mixed mode sessions.

Syntax:
`set service twamp server user` *username* **`password`** *pwd*

Syntax:
`delete service twamp server user` *username*

username
Specifies the username to create.

pwd
Specifies the user account password.

Configuration mode

```
service {
    twamp {
        server {
            user username {
                password pwd
            }
        }
    }
}
```

Use the `set` form of this command to create a user account for accessing the TWAMP server.

> **Note:** Pressing Return after entering the `set service twamp server user username password` command, allows you to enter the password without it being shown on the console.

Use the `delete` form of this command to delete a user account.

show service twamp server session [all | client <ip-address> | summary]

Shows TWAMP session information.

Syntax:

`show service twamp server session [ all | client` *ip-address* `|` **`summary`** `]`

Displays the details of all current TWAMP sessions.

`all`

Displays the details of all current TWAMP sessions.

`client` ***ip-address***

Displays information for a specific TWAMP session.

`summary`

Displays a summary of current TWAMP sessions.

Operational mode

Use this command to display TWAMP session information.

The following example shows how to display the details of all current TWAMP sessions.

```
vyatta@R1:~$ show service twamp server session all
Total number of sessions: 20
Total number of active sessions: 20
--> Control Session initiated by [11.2.0.2]:40972 in Open mode

SID                               SENDER           REFLECTOR          STATUS   DSCP
0b020001d86c04daae30446b81fd2db7  [11.2.0.2]:8772  [11.2.0.1]:50150   ACTIVE   0x22
0b020001d86c04daa6a044ae89bbc550  [11.2.0.2]:8955  [11.2.0.1]:35927   ACTIVE   0x22
0b020001d86c04da9ed83c6cef1afdcd  [11.2.0.2]:8864  [11.2.0.1]:53680   ACTIVE   0x22
0b020001d86c04da9720535c5eee195b  [11.2.0.2]:8927  [11.2.0.1]:47290   ACTIVE   0x22
0b020001d86c04da8fc0c1fc9ca93c01  [11.2.0.2]:8928  [11.2.0.1]:53642   ACTIVE   0x22
<--
--> Control Session initiated by [11:4::2]:36319 in Open mode
SID                               SENDER           REFLECTOR          STATUS   DSCP
00000001d86c04f64ad1f1cf230de104  [11:4::2]:8778   [11:4::1]:57929    ACTIVE   0x24
00000001d86c04f63dc12f0962f9e295  [11:4::2]:8767   [11:4::1]:59181    ACTIVE   0x24
00000001d86c04f6315b573eb64f175c  [11:4::2]:8788   [11:4::1]:45432    ACTIVE   0x24
00000001d86c04f6242f2f98b2210c93  [11:4::2]:8817   [11:4::1]:41301    ACTIVE   0x24
00000001d86c04f6154ff0023445572a  [11:4::2]:8908   [11:4::1]:60239    ACTIVE   0x24
<--
--> Control Session initiated by [11.1.0.2]:47329 in Open mode
SID                               SENDER           REFLECTOR         STATUS   DSCP
0b010001d86c04c3f56d3b6c3ac57bb6  [11.1.0.2]:8815  [11.1.0.1]:52183  ACTIVE   0x21
0b010001d86c04c3e38b47c75ee48e5b  [11.1.0.2]:8769  [11.1.0.1]:54913  ACTIVE   0x21
0b010001d86c04c3d20663c7f080668a  [11.1.0.2]:8847  [11.1.0.1]:33790  ACTIVE   0x21
0b010001d86c04c3c7b3636fc33e9784  [11.1.0.2]:8960  [11.1.0.1]:40148  ACTIVE   0x21
0b010001d86c04c3bde8b3b3bc4bd31f  [11.1.0.2]:8819  [11.1.0.1]:40670  ACTIVE   0x21
<--
--> Control Session initiated by [11:3::2]:33644 in Open mode
SID                               SENDER           REFLECTOR         STATUS   DSCP
00000001d86c04eb699b6f5c6e85f63f  [11:3::2]:8763   [11:3::1]:57336   ACTIVE   0x23
00000001d86c04eb5c679cc7b99e8083  [11:3::2]:8835   [11:3::1]:53602   ACTIVE   0x23
00000001d86c04eb50058149325653cb  [11:3::2]:8760   [11:3::1]:52560   ACTIVE   0x23
00000001d86c04eb1876d117e72ea6e0  [11:3::2]:8817   [11:3::1]:42781   ACTIVE   0x23
00000001d86c04eb0b238972b68ae13d  [11:3::2]:8935   [11:3::1]:55790   ACTIVE   0x23
```

```
<--
```

The following example shows how to display the information for the TWAMP session that is associated with the 11.1.0.2 IP address.

```
vyatta@R1:~$ show service twamp server session client 11.1.0.2
--> Control Session initiated by [11.2.0.2]:40972 in Open mode

SID                                 SENDER           REFLECTOR          STATUS   DSCP
0b020001d86c04daae30446b81fd2db7  [11.2.0.2]:8772  [11.2.0.1]:50150  ACTIVE   0x22
0b020001d86c04daa6a044ae89bbc550  [11.2.0.2]:8955  [11.2.0.1]:35927  ACTIVE   0x22
0b020001d86c04da9ed83c6cef1afdcd  [11.2.0.2]:8864  [11.2.0.1]:53680  ACTIVE   0x22
0b020001d86c04da9720535c5eee195b  [11.2.0.2]:8927  [11.2.0.1]:47290  ACTIVE   0x22
0b020001d86c04da8fc0c1fc9ca93c01  [11.2.0.2]:8928  [11.2.0.1]:53642  ACTIVE   0x22
<--
```

The following example shows how to display a summary of current TWAMP sessions.

```
vyatta@R1:~$ show service twamp server session summary
Total connected clients:       4
Total active test sessions:    20
Total inactive test sessions:  0

Client 0: Initiated by [11.2.0.2]:40972 in Open mode
        Active sessions: 5
        Inactive sessions: 0

Client 1: Initiated by [11:4::2]:36319 in Open mode
        Active sessions: 5
        Inactive sessions: 0

Client 2: Initiated by [11.1.0.2]:47329 in Open mode
        Active sessions: 5
        Inactive sessions: 0

Client 3: Initiated by [11:3::2]:33644 in Open mode
        Active sessions: 5
        Inactive sessions: 0
```

twping <host-address>

Measures the round-trip IP performance using the TWAMP server.

Syntax:

`twping` *host-address* [**`auth-mode`** { **`authenticated`** | **`encrypted`** | **`mixed`** } **`user`** *user*] [**`control-port`** *port*] [**`count`** *count*] [**`interval`** *seconds*] [**`padding`** *size*] [**`port-range`** *port1* **-** *port2*] [**`sample`** *seconds*] [**`session-count`** *s-count*] [**`test-dscp-value`** *dscp-value*]

host-address

Host name or IP address (IPv4 or IPv6) of the TWAMP server.

`auth-mode { authenticated | encrypted | mixed } user` ***user***

Authentication mode (authenticated, encrypted, or mixed).

`control-port` ***port***

Port for server control connection.

`count` ***count***

Number of test packets to send (default is 100).

`interval` ***seconds***

Number of seconds between test packets.

`padding` ***size***

Padding, in bytes, to add to test packets.

`port-range` ***port1*** **`-`** ***port2***

UDP port range to use for test packets.

`sample` ***seconds***

Interval, in seconds, to display statistics during the session.

`session-count` ***s-count***

Number of test sessions to create and use.

`test-dscp-value` ***dscp-value***

Base-10 DSCP value. The value ranges from 0 through 63; the default value is 0.

Operational mode

Use this command to measure round-trip IP performance. If the authentication mode is authenticated, encrypted, or mixed, then this command prompts you to enter the required username and password before continuing.

TWAMP RPC Calls

Overview

The vRouter supports Network Configuration Protocol (NETCONF) Remote Procedure Calls (RPCs), which allow you to remotely run certain vRouter CLI commands. The vRouter supports `twping`, an RPC equivalent of the `twping` operational CLI command. The following sections describe the `twping` RPC call and how to use it.

> **Note:** For more information about NETCONF support on the vRouter, refer to AT&T Vyatta Network Operating System Remote Management Configuration Guide.

YANG model for the twping RPC call

The `twping` RPC call has the following YANG definition.

```
rpc twping {
    description "Measure the round trip time using TWAMP";
    typedef time-interval-ms {
        description "Floating point value representing a time interval";
        type decimal64 {
            fraction-digits 12;
        }
        units "milliseconds";
    }
    grouping min-max-time {
        leaf min {
            description "Minimum time observed during the sample period";
            type time-interval-ms;
        }
        leaf max {
            description "Maximum time observed during the sample period";
            type time-interval-ms;
        }
    }
    grouping addr-port {
        leaf address {
            description "IP address or domain name";
            type union {
                type types:ip-address;
                type types:domain-name;
            }
        }
        leaf port {
            description "Port number";
            type inet:port-number;
        }
    }
    grouping time-stats {
        uses min-max-time;
        leaf median {
            description "Median time observed during the sample period";
            type time-interval-ms;
        }
        leaf error {
            description "Calculated error for timing values";
            type time-interval-ms;
        }
        leaf pdv {
            description "Packet Delay Variation";
```

```
                type time-interval-ms;
            }
        }
        grouping hops {
            leaf diff-num-ttl {
                description "A count of how many different hop count values were observed during the
test.";
                type uint32;
            }
            leaf min {
                description "Minimum number of hops taken by a test packet";
                type uint32;
            }
            leaf max {
                description "Maximum number of hops taken by a test packet";
                type uint32;
            }
        }
        input {
            leaf host {
                description "IP address or domain name of the test reflector";
                type union {
                    type types:ip-address;
                    type types:domain-name;
                }
                mandatory true;
            }
            leaf count {
                description "Number of test packets to send";
                type uint32 {
                    range 1..1000;
                }
                default 100;
            }
            leaf padding {
                description "Size of the padding to add to each test packet";
                type uint32 {
                    range 0..65000;
                }
                units "bytes";
            }
            leaf session-count {
                description "Number of test sessions to create and use";
                type uint32 {
                    range 1..65535;
                }
                default 1;
            }
            leaf test-dscp-value {
                description "RFC 2474 style DSCP value for TOS byte in test packets";
                type uint32 {
                    range 0..63;
                }
                default 0;
            }
            leaf control-port {
                description "Port to be used for the server control connection";
                type inet:port-number {
                    range 1..65535;
                }
                default 862;
            }
            leaf interval {
                description "Mean average time between
                         each test packet sent";
                type decimal64 {
                    fraction-digits 12;
                    range 0.0..max;
                }
                units "seconds";
            }
            container port-range {
                must "start <= end" {
                    error-message
                        "port-range start must be lower than or equal to port-range end";
                }
                must "(end - start + 1) >= ../session-count" {
                    description
                        "Each session uses a different port
                         number, therefore, there must be
                         at least as many available ports
                         in the port-range as sessions that
                         are going to be opened";
                    error-message "Size of the port-range must be at least as large as session-
count";
                }
                leaf start {
                    description "The lowest port number that can be used during the test";
                    type inet:port-number {
                        range 1..65535;
                    }
                    default 8760;
                }
                leaf end {
                    description "The highest port number that can be used during the test";
                    type inet:port-number {
                        range 1..65535;
                    }
                    default 8960;
                }
            }
            container authentication {
                presence "Enables authentication";
                leaf mode {
                    description "Authentication mode";
                    type enumeration {
                        enum "authenticate";
                        enum "encrypt";
                        enum "mixed";
                    }
                    default "authenticate";
                }
                leaf user {
                    description "User name";
                    type string {
                        length 1..16;
                    }
                    mandatory true;
                }
                leaf passphrase {
                    description "Passphrase for user";
                    type string {
                        length 1..1024;
                    }
                    mandatory true;
                }
            }
        }
        output {
            list results {
                key sid;
                leaf sid {
                    description "Session Identifier";
                    type string;
                }
                container source {
                    description "Source address that test packets originated from";
                    uses addr-port;
                }
                container destination {
                    description "Destination address of the test reflector";
                    uses addr-port;
                }
                container packets {
                    leaf time-of-first {
                        description "Time that the first test packet was sent";
                        type ietf:date-and-time;
                    }
                    leaf time-of-last {
                        description "Time that the last test packet was sent";
                        type ietf:date-and-time;
                    }
                    leaf num-pkts-sent {
                        description "Number of test packets that were sent";
                        type uint32;
                    }
                    leaf num-pkts-lost {
                        description "Number of test packets that were lost";
                        type uint32;
                    }
                    leaf send-duplicates {
                        description "Number of duplicate test packets received by the reflector";
                        type uint32;
                    }
                    leaf reflect-duplicates {
                        description "Number of duplicate test packets received by the sender";
                        type uint32;
                    }
                }
                container round-trip-time {
                    description "Total round trip time, from when the test packet is sent from the 
 test
                                 client, until it is received back from the reflector";
                    uses time-stats;
                }
                container send-time {
                    description "Wire time for test packets to go from sender to reflector";
                    uses time-stats;
                }
                container reflect-time {
                    description "Wire time for test packets to go from the reflector, back to the 
 sender";
                    uses time-stats;
                }
                container reflector-processing-time {
                    description "Time taken to process test packets within the reflector";
                    uses min-max-time;
                }
                container send-hops {
                    description "Number of hops taken by the test packets going from the sender to
                                 the reflector.";
                    uses hops;
                }
                container reflect-hops {
                    description "Number of hops taken by the test packets going from the reflector 
 back to
                                 the sender";
                    uses hops;
```

```
                }
            }
        }
        configd:call-rpc "twamp-ping-rpc";
    }
```

Input parameters

The following table lists the input parameters of the `twping` RPC call.

Parameter	Description
host	IP address or domain name of a test reflector.
count	Number of test packets to send. The number ranges from 1 through 1000. The default is 100.
control-port	Port to be used for the server control connection. The port number ranges from 1 through 65535. The default is 862.
interval	Mean average time in seconds between each test packet sent. The mean ranges from 0.0 through 9223372.036855.
padding	Number of bytes of padding that is added to each test packet. The number ranges from 0 through 65000. If not specified, the vRouter uses an implicit default value.
session-count	Number of test sessions to create and use. The number ranges from 1 through 65535. The default is 1.
test-dscp-value	RFC 2474-style DSCP value for the TOS byte in test packets. The value ranges from 0 through 63. The default is 0.
port-range	Range of allowed port numbers.
	start: Lowest port number that can be used during the test. The port number ranges from 1 through 65535. The default is 8760. The starting port number must be less than or equal to the ending port number.
	end: Highest port number that can be used during the test. The port number ranges from 1 through 65535. The default is 8960.
authentication	The authentication mode and user credentials.
	mode: Authentication mode. The mode can be one of the following modes: • authenticate • encrypt • mixed The default mode is authenticate.
	user: Username. The username is mandatory.
	passphrase: Passphrase for the user. The length of the passphrase ranges from 1 through 1024 characters and there are no explicit restrictions on characters. The passphrase is mandatory.

Output parameters

The following table lists the output parameters of the `twping` RPC call.

Parameter	Description
sid	Identifier of the session between the client and TWAMP server.
source	Address of the vRouter from which test packets originate.
	address: IP address or domain name of the source vRouter.
	port: Port number from which the test packets are sent.
round-trip-time	Length of time it takes for the test packets to be sent from the source vRouter (client) plus the length of time it takes for the response packets to be received from the session reflector of the TWAMP server, which can run on a vRouter or third-party system.
	median: Median round-trip time observed during the sampling period.
	pdv: Packet-delay variation (PDV), measured in milliseconds.
	error: Calculated error for timing values, measured in milliseconds.
	min: Minimum round-trip time observed during the sampling period, measured in milliseconds.
	max: Maximum round-trip time observed during the sampling period, measured in milliseconds.
destination	Destination address of the session reflector on the TWAMP server.
	address: IP address or domain name of the reflector.
	port: Port number to which the test packets are sent.
reflect-hops	Number of hops it takes for the response packets to get from the session reflector to the client.
	diff-num-ttl: A count of how many different hop count values were observed during the test.
	min: Minimum number of hops taken by a test packet.
	max: Maximum number of hops taken by a test packet.
send-time	Wire time for test packets to go from the sender to the reflector.
	median: Median send time observed during the sampling period.
	pdv: Packet-delay variation (PDV), measured in milliseconds.
	error: Calculated error for timing values, measured in milliseconds.
	min: Minimum send time observed during the sampling period, measured in milliseconds.

Parameter	Description
	max: Maximum send time observed during the sampling period, measured in milliseconds.
reflector-processing-time	Time it takes to process test packets within the reflector.
	min: Minimum amount of time for processing the test packets.
	max: Maximum amount of time for processing the test packets.
packets	
	num-pkts-sent: Number of test packets that were sent.
	time-of-first: Time at which the first test packet was sent.
	num-pkts-lost: Number of test packets that were lost.
	send-duplicates: Number of duplicate test packets received by the reflector.
	time-of-last: Time at which the last test packet was sent.
	reflect-duplicates: Number of duplicate test packets received by the sender.
reflect-time	Wire time for test packets to get from the reflector back to the sender.
	median: Median reflect time observed during the sampling period.
	pdv: Packet-delay variation (PDV), measured in milliseconds.
	error: Calculated error for timing values, measured in milliseconds.
	min: Minimum reflect time observed during the sampling period, measured in milliseconds.
	max: Maximum reflect time observed during the sampling period, measured in milliseconds.
send-hops	Number of hops it takes for the packets to get from the sender to the reflector.
	diff-num-ttl: A count of how many different hop count values were observed during the test.
	min: Minimum number of hops taken by a test packet.
	max: Maximum number of hops taken by a test packet.

Differences between the twping CLI command and the twping RPC call

The CLI and RPC versions of `twping` have the same functionality, but they have minor differences that are described in the following sections.

Authentication

When using the `twping` CLI command, you must specify the authentication mode. However, when invoking a `twping` RPC call, you do not have to explicitly specify the authentication mode because, by default, the mode is set to **`authenticate`**. However, you must specify a username and passphrase because RPC calls do not support interactive prompts.

The following example shows how to specify the authentication mode when running the `twping` CLI command. After entering the command, the CLI prompts you for a passphrase. After you enter the passphrase, the vRouter pings the TWAMP server and displays the results.

```
v@vyatta:~$ twping localhost auth-mode authenticate user v
Enter passphrase for identity 'v':
Approximately 13.2 seconds until results available
--- twping statistics from [localhost]:8904 to [localhost]:44984 ---
SID:    00000001da3a464baf1eeada687abc61
```

The following example shows two `twping` RPC calls. The first call does not specify the authentication mode, but the second call does. Both calls specify a user and the corresponding passphrase.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>localhost</host>
    <authentication>
      <user>user1</user>
      <passphrase>example-passphrase</passphrase>
    </authentication>
  </twping>
</rpc>]]>]]>

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>localhost</host>
    <authentication>
      <mode>authenticate</mode>
      <user>user1</user>
      <passphrase>example-passphrase</passphrase>
    </authentication>
  </twping>
</rpc>]]>]]>
```

Port range

Slight differences exist in how you specify port ranges when using the CLI version of the `twping` command as opposed to the RPC versions.

The `twping` RPC call has a default port range (8760 through 8960), which is equivalent to **`twping`** *reflector* **`port-range 8760-8960`**. The `twping` CLI command lets you specify the starting and ending port numbers or just a single port number. If you specify a single port number, the `twping` command uses the same number for the starting and ending port numbers. For example, **`twping`** *reflector* **`port-range 8000`** is equivalent to **`twping`** *reflector* **`port-range 8000-8000`**.

When invoking a `twping` RPC call, the behavior is slightly different. You can specify starting and ending ports for the range, or specify just one port. If you specify only one port, the `twping` RPC call uses the default port for the other port in the range.

For example, if you specify 9000 as the ending port number, the command sets the starting port number to 8760.

```
<port-range>
      <end>9000</end>
</port-range>

<port-range>
```

```
      <start>8760</start>
      <end>9000</end>
</port-range>
```

If you specify 8800 as the starting port number, the command sets the ending value to 8960.

```
<port-range>
      <start>8800</start>
</port-range>

<port-range>
      <start>8800</start>
      <end>8960</end>
</port-range>
```

Time stamps

The `twping` RPC call supports ISO8601-formatted time stamps, which differ slightly from those that are supported by the `twping` CLI command. The time-of-first and time-of-last time stamps that are supported by the `twping` RPC call include time zone offsets. These offsets are not included by the `twping` CLI command.

The following example shows the time-of-first and time-of-last time stamps that are generated by the `twping` CLI command.

```
first: 2016-01-08T13:51:41.855
last: 2016-01-08T13:51:52.016
```

The following example shows the RPC-equivalent time stamps when the time zone is set to UTC.

```
<time-of-first>2016-01-08T13:51:41.855+00:00</time-of-first>
<time-of-last>2016-01-08T13:51:52.016+00:00</time-of-last>
```

Making twping RPC calls

The following sections show examples of `twping` RPC calls with their equivalent `twping` CLI commands.

Making a basic RPC twping call

The following example shows the RPC equivalent of the `twping localhost` CLI command.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>localhost</host>
  </twping>
</rpc>]]>]]>
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <results xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <sid>00000001da3a58d956c9be3a604b90fa</sid>
    <reflector-processing-time>
      <min>0.00191</min>
      <max>0.0181</max>
    </reflector-processing-time>
    <source>
      <address>localhost</address>
      <port>8890</port>
    </source>
    <send-time>
      <min>0.0162</min>
      <pdv>0.0</pdv>
      <max>0.0839</max>
      <median>0.1</median>
      <error>0.0441</error>
    </send-time>
    <packets>
      <reflect-duplicates>0</reflect-duplicates>
      <num-pkts-lost>0</num-pkts-lost>
      <time-of-last>2016-01-08T15:48:49.952+00:00</time-of-last>
      <send-duplicates>0</send-duplicates>
      <time-of-first>2016-01-08T15:48:42.721+00:00</time-of-first>
      <num-pkts-sent>100</num-pkts-sent>
    </packets>
    <round-trip-time>
      <error>0.0882</error>
      <min>0.0234</min>
      <pdv>0.0</pdv>
      <median>0.1</median>
      <max>0.108</max>
    </round-trip-time>
    <reflect-time>
      <max>0.0343</max>
      <error>0.0441</error>
      <min>0.00668</min>
      <pdv>0.0</pdv>
      <median>0.1</median>
    </reflect-time>
    <send-hops>
      <min>0</min>
      <diff-num-ttl>1</diff-num-ttl>
      <max>0</max>
    </send-hops>
    <reflect-hops>
      <diff-num-ttl>1</diff-num-ttl>
      <max>0</max>
      <min>0</min>
    </reflect-hops>
    <destination>
      <port>53612</port>
      <address>localhost</address>
    </destination>
  </results>
</rpc-reply>
]]>]]>
```

Initiating multiple twping sessions

The following example shows the RPC equivalent of the `twping localhost session-count 2` CLI command. The RPC call initiates two sessions.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>localhost</host>
    <session-count>2</session-count>
  </twping>
</rpc>]]>]]>
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <results xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <reflector-processing-time>
      <max>0.0238</max>
      <min>0.00334</min>
    </reflector-processing-time>
    <send-hops>
      <max>0</max>
      <min>0</min>
      <diff-num-ttl>1</diff-num-ttl>
    </send-hops>
    <round-trip-time>
      <max>0.13</max>
      <pdv>0.1</pdv>
      <min>0.0339</min>
      <median>0.1</median>
      <error>0.000001</error>
    </round-trip-time>
    <packets>
      <time-of-last>2016-01-13T14:51:30.480+00:00</time-of-last>
      <reflect-duplicates>0</reflect-duplicates>
      <num-pkts-lost>0</num-pkts-lost>
      <time-of-first>2016-01-13T14:51:22.587+00:00</time-of-first>
      <num-pkts-sent>100</num-pkts-sent>
      <send-duplicates>0</send-duplicates>
    </packets>
    <destination>
      <address>localhost</address>
      <port>55938</port>
    </destination>
    <source>
      <port>8834</port>
      <address>localhost</address>
    </source>
    <sid>00000001da40e2e976cebbba7be73c29</sid>
    <reflect-hops>
      <max>0</max>
      <diff-num-ttl>1</diff-num-ttl>
      <min>0</min>
    </reflect-hops>
    <send-time>
      <min>0.0229</min>
      <median>0.1</median>
      <error>0.0</error>
      <max>0.0987</max>
      <pdv>0.0</pdv>
    </send-time>
    <reflect-time>
      <min>0.00811</min>
      <median>0.1</median>
      <error>0.0</error>
      <max>0.073</max>
      <pdv>0.0</pdv>
    </reflect-time>
  </results>
  <results xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <reflector-processing-time>
      <max>0.0319</max>
      <min>0.00286</min>
    </reflector-processing-time>
    <send-hops>
      <max>0</max>
      <min>0</min>
      <diff-num-ttl>1</diff-num-ttl>
    </send-hops>
    <destination>
      <port>35818</port>
      <address>localhost</address>
    </destination>
    <packets>
      <num-pkts-lost>0</num-pkts-lost>
      <time-of-first>2016-01-13T14:51:22.845+00:00</time-of-first>
      <num-pkts-sent>100</num-pkts-sent>
      <reflect-duplicates>0</reflect-duplicates>
      <time-of-last>2016-01-13T14:51:32.199+00:00</time-of-last>
      <send-duplicates>0</send-duplicates>
    </packets>
    <sid>00000001da40e2e97730835514472ecb</sid>
    <source>
```

```
      <address>localhost</address>
      <port>8905</port>
    </source>
    <reflect-hops>
      <min>0</min>
      <diff-num-ttl>1</diff-num-ttl>
      <max>0</max>
    </reflect-hops>
    <round-trip-time>
      <min>0.0238</min>
      <median>0.1</median>
      <error>0.000001</error>
      <max>0.129</max>
      <pdv>0.1</pdv>
    </round-trip-time>
    <send-time>
      <error>0.0</error>
      <max>0.107</max>
      <pdv>0.0</pdv>
      <min>0.0157</min>
      <median>0.1</median>
    </send-time>
    <reflect-time>
      <min>0.00811</min>
      <median>0.1</median>
      <error>0.0</error>
      <max>0.0491</max>
      <pdv>0.0</pdv>
    </reflect-time>
  </results>
</rpc-reply>
]]>]]>
```

Making an RPC twping call with all possible input parameters

The following example shows the RPC equivalent of the `twping localhost session-count 1 padding 1000 interval 2.0 count 10 test-dscp-value 0 auth-mode authenticate user v control-port 862 port-range 6000` CLI command. This RPC call uses all the possible input parameters.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>localhost</host>
    <count>10</count>
    <control-port>862</control-port>
    <interval>2.0</interval>
    <padding>1000</padding>
    <port-range>
        <start>6000</start>
        <end>6000</end>
    </port-range>
    <session-count>1</session-count>
    <test-dscp-value>0</test-dscp-value>
    <authentication>
        <mode>authenticate</mode>
        <user>v</user>
        <passphrase>v</passphrase>
    </authentication>
  </twping>
</rpc>]]>]]>
<?xml version="1.0"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <results xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <source>
      <address>localhost</address>
      <port>6000</port>
    </source>
    <round-trip-time>
      <median>0.1</median>
      <pdv>0.0</pdv>
      <error>8.79</error>
      <min>0.0591</min>
      <max>0.0968</max>
    </round-trip-time>
    <sid>00000001da3a5bac42f2882ee23c8231</sid>
    <destination>
      <port>52123</port>
      <address>localhost</address>
    </destination>
    <reflect-hops>
      <diff-num-ttl>1</diff-num-ttl>
      <min>0</min>
      <max>0</max>
    </reflect-hops>
    <send-time>
      <max>0.0701</max>
      <error>4.39</error>
      <pdv>0.0</pdv>
      <min>0.0467</min>
      <median>0.1</median>
    </send-time>
    <reflector-processing-time>
      <max>0.0401</max>
      <min>0.00906</min>
    </reflector-processing-time>
    <packets>
      <num-pkts-sent>10</num-pkts-sent>
      <time-of-first>2016-01-08T16:00:46.270+00:00</time-of-first>
      <num-pkts-lost>0</num-pkts-lost>
      <send-duplicates>0</send-duplicates>
      <time-of-last>2016-01-08T16:01:05.768+00:00</time-of-last>
      <reflect-duplicates>0</reflect-duplicates>
    </packets>
    <reflect-time>
      <error>4.39</error>
      <min>0.0119</min>
      <max>0.0286</max>
      <median>0.1</median>
      <pdv>0.0</pdv>
    </reflect-time>
    <send-hops>
      <max>0</max>
      <diff-num-ttl>1</diff-num-ttl>
      <min>0</min>
    </send-hops>
  </results>
</rpc-reply>
]]>]]>
```

Error messages

When a NETCONF RPC fails, it describes the cause of the failure in the `error-message` field of the reply structure that is returned by the RPC.

If the failure is caused by a **`twping`**-related issue, the error messages that are returned by the RPC reply are the same as the error messages that are returned by the **`twping`** CLI command.

If the failure is related to the RPC (for example, missing or invalid RPC call parameters), the following error messages are returned.

Invalid port range

When making an RPC call, if, for a range of ports, you specify a starting number that is greater than the ending number, the following error message is returned.

```
[port-range] port-range start must be lower than or equal to port-range end
```

The following example shows the XML structure of an RPC call that is equivalent to the `twping localhost session-count 1 port-range 6000-5000` CLI command and the error message that is returned in the RPC reply.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>localhost</host>
    <session-count>1</session-count>
    <port-range>
        <start>6000</start>
        <end>5000</end>
    </port-range>
  </twping>
</rpc>]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message>Failed to parse xml input: Invalid XML - [port-range]
port-range start must be lower than or equal to port-range end
[port-range]
Size of the port-range must be at least as large as session-count

&lt;twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1"&gt;
    &lt;host&gt;localhost&lt;/host&gt;
    &lt;session-count&gt;1&lt;/session-count&gt;
    &lt;port-range&gt;
      &lt;start&gt;6000&lt;/start&gt;
      &lt;end&gt;5000&lt;/end&gt;
    &lt;/port-range&gt;
  &lt;/twping&gt;
</error-message>
  </rpc-error>
</rpc-reply>
]]>]]>
```

Invalid port range size

When making a `twping` RPC call, you can specify the number of test sessions to create. However, because each session requires a port, if you specify a port range such that the number of ports is fewer than the number of test sessions, the following error message is returned.

```
[port-range] Size of the port-range must be at least as large as session-count
```

To prevent this error from occurring, make sure that you specify a wide port range in the RPC call. If you do not specify a port range, make sure that you do not specify more than 201 test sessions in the call because the default range (8760 through 8960) accommodates as many as 201 ports.

The following example shows the XML structure of a `twping` RPC call in which the number of test sessions is greater than the number of ports in the port range.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>localhost</host>
    <session-count>100</session-count>
    <port-range>
```

```
       <start>8000</start>
       <end>8005</end>
     </port-range>
   </twping>
</rpc>]]>]]>
```

Missing mandatory parameters

When making an RPC call, you must supply the following parameters.

- host (always mandatory)
- user (mandatory when authentication is required)
- passphrase (mandatory when authentication is required)

If any parameter is missing, the following error message is returned.

```
Missing mandatory node
```

The following example shows the XML structure of an RPC call and the returned error message. In this example, the RPC call does not specify a host. In addition, even though the call requires authentication, no user or passphrase is specified.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <authentication>
       <mode>authenticate</mode>
    </authentication>
  </twping>
</rpc>]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-message>Failed to parse xml input: Invalid XML - []

Missing mandatory node host

[authentication]

Missing mandatory node user

[authentication]

Missing mandatory node passphrase


&lt;twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1"&gt;
    &lt;authentication&gt;
      &lt;mode&gt;authenticate&lt;/mode&gt;
    &lt;/authentication&gt;
  &lt;/twping&gt;
</error-message>
  </rpc-error>
</rpc-reply>
```

VRF Support

VRF support for DHCP

The implementation of VRF on the AT&T Vyatta vRouter supports DHCPv4 server, DHCPv6 server, DHCPv4 relay, DHCPv6 relay, DHCPv4 client, and DHCPv6 client configurations.

You can configure DHCP on individual routing instances. If you configure DHCP without specifying a routing instance, the default routing instance is used.

The DHCP server recognizes which address pool belongs to which routing instance. You can switch configurations between routing instances. However, you cannot create a DHCP relay that involves interfaces from different routing instances.

The following examples show DHCP configurations that use these values:

- routing instance = BLUE
- ipAddress = 42.42.42.42

The following example shows how to configure DHCP for the default routing instance.

```
vyatta@R1# set service dhcp-server listento interface 'dp0s3'
vyatta@R1# set service dhcp-server shared-network-name CORP subnet 42.42.42.0/24 start 42.42.42.1
 stop '42.42.42.253'
vyatta@R1# commit
vyatta@R1# run show configuration
service {
        dhcp-server {
                listento {
                        interface dp0s3
                }
                shared-network-name CORP {
                        subnet 42.42.42.0/24 {
                                start 42.42.42.1 {
                                        stop 42.42.42.253
                                }
                        }
                }
        }
}
```

The following example shows the same configuration sequence for the BLUE routing instance.

```
vyatta@R1# set routing routing-instance BLUE service dhcp-server listento interface 'dp0s4'
vyatta@R1# set routing routing-instance BLUE service dhcp-server shared-network-name CORP subnet
 42.42.42.0/24 start 42.42.42.1 stop '42.42.42.253'
vyatta@R1# commit
vyatta@R1# run show configuration
routing {
        routing-instance BLUE {
                interface dp0s4
                service {
                        dhcp-server {
                                listento {
                                        interface dp0s4
                                }
                                shared-network-name CORP {
                                        subnet 42.42.42.0/24 {
                                                start 42.42.42.1 {
                                                        stop 42.42.42.253
                                                }
                                        }
                                }
                        }
                }
        }
}}
```

For more information about DHCP and configuring DHCP, see AT&T Vyatta Network Operating System Basic System Configuration Guide.

VRF support for DNS

The AT&T Vyatta vRouter uses DNS in both the client (resolver) and server (proxy) roles. You can configure DNS for individual routing instances. If you configure DNS without specifying a routing instance, the default routing instance is used.

For DNS client (resolver) operations, configure DNS name servers when creating a new routing instance to support DNS clients and dynamic synchronization. As a client, a set of name server addresses is used to resolve queried domain names and update DNS records dynamically.

The following example shows how to configure the 10.70.20.23 DNS name server for the default routing instance.

```
vyatta@R1# set system name-server 10.70.20.23
vyatta@R1# commit
vyatta@R1# run show configuration
system {
 name-server 10.70.20.23
}
```

The following example shows the same configuration sequence for the BLUE routing instance.

```
vyatta@R1# set routing routing-instance BLUE system name-server 10.70.20.23
vyatta@R1# commit
vyatta@R1# run show configuration
routing {
        routing-instance BLUE {
                system {
                        name-server 10.70.20.23
                }
        }
}
```

For server (proxy) operations, if the queried record is not in the cache, the AT&T Vyatta vRouter sends the query to DNS servers that are listed in the name server list for the specified routing instance. This name server list can apply to each DNS forwarding instance when configuring DNS forwarding. If not configured, the routing-instance-specific name servers that are configured are used.

The following example shows how to configure proxy settings (listen-on interface dp0s4 and cache size 1024) for the default routing instance.

```
vyatta@R1# set service dns forwarding listen-on dp0s4
vyatta@R1# set service dns forwarding cache-size 1024
vyatta@R1# commit
vyatta@R1# run show configuration
service {
        dns {
                forwarding {
                        cache-size 1024
                        listen-on dp0s3
                }
        }
}
```

The following example shows the same configuration sequence for the BLUE routing instance.

```
vyatta@R1# set routing routing-instance BLUE service dns forwarding listen-on dp0s4
vyatta@R1# set routing routing-instance BLUE service dns forwarding cache-size 1024
```

```
vyatta@R1# commit
vyatta@R1# run show configuration
routing {
        routing-instance BLUE {
                service {
                        dns {
                                forwarding {
                                        cache-size 1024
                                        listen-on dp0s4
                                }
                        }
                }
        }
}
```

For more information about DNS and configuring DNS, see AT&T Vyatta Network Operating System Basic System Configuration Guide.

VRF support for TWAMP

The vRouter supports the configuration of a TWAMP service for individual routing instances and the operation of the twping TWAMP client in the context of a specified routing instance.

To specify the routing instance in which a TWAMP server runs, use the optional **`routing routing-instance`** *routing-instance* keywords and variable. For example, the following command configures a TWAMP server to run in the BLUE routing instance.

```
$ set routing routing-instance BLUE service twamp server
```

If you do not specify a routing instance, the TWAMP server runs in the context of the default routing instance.

TWAMP message logging

TWAMP messages that are logged by twampd in the context of a user-configured routing instance are prepended with the name of the instance, as shown in the following example. In this example, the log message was logged in the context of the GREEN routing instance.

```
Apr 08 07:31:24 vm-next-1 twampd[22829]: [twampd@green.service] StartSessions 1 sessions
```

TWAMP messages that are logged by twampd in the context of the default routing instance do not specify the name of the instance, as shown in the following example.

```
Apr 05 15:32:47 vm-torrance-1 twampd[5149]: StartSessions 1 sessions
```

twping RPC VRF support

When making twping RPC calls, you can specify a routing instance, as specified by the leaf definition of the routing instance in the YANG model.

```
leaf routing-instance {
     description "The routing instance context for this session";
     type routing:routing-instance-name;
     default "default";
}
```

The following is an example of a twping RPC call.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <twping xmlns="urn:vyatta.com:mgmt:vyatta-service-twamp:1">
    <host>10.10.3.2</host>
    <routing-instance>blue</routing-instance>
  </twping>
</rpc>]]>]]>
```

Command support for VRF routing instances

VRF allows an AT&T Vyatta vRouter to support multiple routing tables, one for each VRF routing instance. Some commands in this guide support VRF and can be applied to particular routing instances.

Use the guidelines in this section to determine correct syntax when adding VRF routing instances to commands. For more information about VRF, refer to AT&T Vyatta Network Operating System Basic Routing Configuration Guide. This guide includes an overview of VRF, VRF configuration examples, information about VRF-specific features, and a list of commands that support VRF routing instances.

Adding a VRF routing instance to a Configuration mode command

For most Configuration mode commands, specify the VRF routing instance at the beginning of a command. Add the appropriate VRF keywords and variable to follow the initial action (**set**, **show**, or **delete**) and before the other keywords and variables in the command.

Example: Configuration mode example: syslog

The following command configures the syslog logging level for the specified syslog host. The command does not include a VRF routing instance, so the command applies to the default routing instance.

```
vyatta@R1# set system syslog host 10.10.10.1 facility all level debug
vyatta@R1# show system syslog
syslog {
    host 10.10.10.1 {
            facility all {
                    level debug
            }
    }
}
```

The following example shows the same command with the VRF routing instance (GREEN) added. Notice that **routing routing-instance GREEN** has been inserted between the basic action (**set** in the example) and the rest of the command. Most Configuration mode commands follow this convention.

```
vyatta@R1# set routing routing-instance GREEN system syslog host 10.10.10.1 facility all
 level debug
vyatta@R1# show routing
routing {
     routing-instance GREEN {
             system {
                     syslog {
                            host 11.12.13.2:514 {
                                     facility all {
                                             level debug
                                     }
                             }
                     }
             }
     }
}
```

Example: Configuration mode example: SNMP

Some features, such as SNMP, are not available on a per-routing instance basis but can be bound to a specific routing instance. For these features, the command syntax is an exception to the convention of specifying the routing instance at the beginning of Configuration mode commands.

The following example shows how to configure the SNMPv1 or SNMPv2c community and context for the RED and BLUE routing instances. The first two commands specify the RED routing instance as the context for community A and BLUE routing instance as the context for community B. The subsequent commands complete the configuration.

For more information about configuring SNMP, refer to AT&T Vyatta Network Operating System Remote Management Configuration Guide.

```
vyatta@R1# set service snmp community commA context RED
vyatta@R1# set service snmp community commB context BLUE
vyatta@R1# set service snmp view all oid 1
vyatta@R1# set service snmp community commA view all
vyatta@R1# set service snmp community commB view all
vyatta@R1# show service snmp community
 community commA {
        context RED
        view all
 }
 community commB {
        context BLUE
        view all
 }
[edit]
vyatta@vyatta#
```

Adding a VRF routing instance to an Operational mode command

The syntax for adding a VRF routing instance to an Operational mode command varies according to the type of command parameters:

- If the command does not have optional parameters, specify the routing instance at the end of the command.
- If the command has optional parameters, specify the routing instance after the required parameters and before the optional parameters.

Example: Operational mode examples without optional parameters

The following command displays dynamic DNS information for the default routing instance.

```
vyatta@vyatta:~$ show dns dynamic status
```

The following command displays the same information for the specified routing instance (GREEN). The command does not have any optional parameters, so the routing instance is specified at the end of the command.

```
vyatta@vyatta:~$ show dns dynamic status routing-instance GREEN
```

Example: Operational mode example with optional parameters

The following command obtains multicast path information for the specified host (10.33.2.5). A routing instance is not specified, so the command applies to the default routing instance.

```
vyatta@vyatta:~$ mtrace 10.33.2.5 detail
```

The following command obtains multicast path information for the specified host (10.33.2.5) and routing instance (GREEN). Notice that the routing instance is specified before the optional **detail** keyword.

```
vyatta@vyatta:~$ mtrace 10.33.2.5 routing-instance GREEN detail
```

Example: Operational mode example output: SNMP

The following SNMP **show** commands display output for routing instances.

```
vyatta@vyatta:~$ show snmp routing-instance
Routing Instance SNMP Agent is Listening on for Incoming Requests:
Routing-Instance            RDID
-----------------           ----
RED                         5

vyatta@vyatta:~$ show snmp community-mapping
SNMPv1/v2c Community/Context Mapping:
Community                   Context
---------                   -------
commA                       'RED'
commB                       'BLUE'
deva                        'default'


vyatta@vyatta:~$ show snmp trap-target
SNMPv1/v2c Trap-targets:
Trap-target                   Port   Routing-Instance Community
-----------                   ----   ---------------- ---------
1.1.1.1                              'RED'           'test'


vyatta@vyatta:~$ show snmp v3 trap-target
SNMPv3 Trap-targets:
Trap-target                   Port   Protocol Auth Priv Type   EngineID              Routing-
Instance User
-----------                   ----   -------- ---- ---- ----   --------
 ---------------- ----
2.2.2.2                       '162'  'udp'    'md5      'infor                       'BLUE'
         'test'
```

List of Acronyms

Acronym	Description
ACL	access control list
ADSL	Asymmetric Digital Subscriber Line
AH	Authentication Header
AMI	Amazon Machine Image
API	Application Programming Interface
AS	autonomous system
ARP	Address Resolution Protocol
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BPDU	Bridge Protocol Data Unit
CA	certificate authority
CCMP	AES in counter mode with CBC-MAC
CHAP	Challenge Handshake Authentication Protocol
CLI	command-line interface
DDNS	dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
DLCI	data-link connection identifier
DMI	desktop management interface
DMVPN	dynamic multipoint VPN
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System

Acronym	Description
DSCP	Differentiated Services Code Point
DSL	Digital Subscriber Line
eBGP	external BGP
EBS	Amazon Elastic Block Storage
EC2	Amazon Elastic Compute Cloud
EGP	Exterior Gateway Protocol
ECMP	equal-cost multipath
ESP	Encapsulating Security Payload
FIB	Forwarding Information Base
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
HDLC	High-Level Data Link Control
I/O	Input/Output
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IPS	Intrusion Protection System
IKE	Internet Key Exchange
IP	Internet Protocol
IPOA	IP over ATM
IPsec	IP Security
IPv4	IP Version 4
IPv6	IP Version 6
ISAKMP	Internet Security Association and Key Management Protocol
ISM	Internet Standard Multicast

Acronym	Description
ISP	Internet Service Provider
KVM	Kernel-Based Virtual Machine
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LAN	local area network
LDAP	Lightweight Directory Access Protocol
LLDP	Link Layer Discovery Protocol
MAC	medium access control
mGRE	multipoint GRE
MIB	Management Information Base
MLD	Multicast Listener Discovery
MLPPP	multilink PPP
MRRU	maximum received reconstructed unit
MTU	maximum transmission unit
NAT	Network Address Translation
NBMA	Non-Broadcast Multi-Access
ND	Neighbor Discovery
NHRP	Next Hop Resolution Protocol
NIC	network interface card
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OSPFv2	OSPF Version 2
OSPFv3	OSPF Version 3
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PAT	Port Address Translation
PCI	peripheral component interconnect

Acronym	Description
PIM	Protocol Independent Multicast
PIM-DM	PIM Dense Mode
PIM-SM	PIM Sparse Mode
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPPoA	PPP over ATM
PPPoE	PPP over Ethernet
PPTP	Point-to-Point Tunneling Protocol
PTMU	Path Maximum Transfer Unit
PVC	permanent virtual circuit
QoS	quality of service
RADIUS	Remote Authentication Dial-In User Service
RHEL	Red Hat Enterprise Linux
RIB	Routing Information Base
RIP	Routing Information Protocol
RIPng	RIP next generation
RP	Rendezvous Point
RPF	Reverse Path Forwarding
RSA	Rivest, Shamir, and Adleman
Rx	receive
S3	Amazon Simple Storage Service
SLAAC	Stateless Address Auto-Configuration
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
SPT	Shortest Path Tree
SSH	Secure Shell

Acronym	Description
SSID	Service Set Identifier
SSM	Source-Specific Multicast
STP	Spanning Tree Protocol
TACACS+	Terminal Access Controller Access Control System Plus
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
ToS	Type of Service
TSS	TCP Maximum Segment Size
Tx	transmit
UDP	User Datagram Protocol
VHD	virtual hard disk
VIF	virtual interface
VLAN	virtual LAN
VPC	Amazon virtual private cloud
VPN	virtual private network
VRRP	Virtual Router Redundancy Protocol
WAN	wide area network
WAP	wireless access point
WPA	Wired Protected Access