Remote Management 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 SSH, Telnet, Web GUI access (HTTPS), NETCONF and SNMP for remote management of the AT&T Vyatta vRouter (referred to as a virtual router, vRouter, or router in the guide).

For information about how to remotely manage the vRouter by using the Remote Access API, a REST API, refer to AT&T Vyatta Network Operating System Remote Access API Reference Guide.

SSH

SSH configuration

Secure Shell (SSH) provides a secure mechanism to log on to the AT&T Vyatta vRouter and access the Command Line Interface (CLI). Configuring SSH is optional, but is recommended to provide secure remote access to the AT&T Vyatta vRouter. In addition to the standard password authentication provided by SSH, shared public key authentication is also available.

The following table enables SSH for password authentication on the default port (port 22), as shown in the following figure. By default, only SSH version 2 is enabled.

Figure 1: Enabling SSH access

To enable the SSH service on the AT&T Vyatta vRouter, perform the following steps in configuration mode.

Table 1: Enabling SSH access

Step	Command
Create the configuration node for the SSH service.	`vyatta@R1# set service ssh`
Commit the information	`vyatta@R1# commit` `Restarting OpenBSD Secure Shell server: sshd.`

Show the configuration.	`vyatta@R1# show service`
	` ssh {`
	` }`

SSH Commands

service ssh allow-root

Specifies that root logins are to be allowed on SSH connections.

Syntax:
```
set service ssh allow-root
```

Syntax:
```
delete service ssh allow-root
```

Syntax:
```
show service ssh
```

Root logins are not allowed on SSH connections.

Configuration mode

```
service {
 ssh {
  allow-root
 }
}
```

Use this command to specify that root logins are to be allowed on SSH connections.

> **Note:** The root account is often the target of external attacks so its use is discouraged. The vyatta account provides sufficient privileges to administer the system.

Use the `set` form of this command to specify that root logins are to be allowed on SSH connections.

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

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

service ssh

Enables SSH as an access protocol on the AT&T Vyatta vRouter.

Syntax:
```
set service ssh
```

Syntax:
```
delete service ssh
```

Syntax:
```
show service ssh
```

Configuration mode

```
service {
 ssh {
 }
}
```

Use this command to configure the system to allow SSH requests from remote systems to the local system.

Creating the SSH configuration node enables SSH as an access protocol. By default, the router uses port 22 for the SSH service, and SSH version 2 alone is used.

Use the `set` form of this command to create the SSH configuration.

Use the `delete` form of this command to remove the SSH configuration. If you delete the SSH configuration node you will disable SSH access to the system.

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

service ssh disable-host-validation

Specifies that SSH should not validate clients via reverse DNS lookup.

Syntax:
```
set service ssh disable-host-validation
```

Syntax:
```
delete service ssh disable-host-validation
```

Syntax:
```
show service ssh
```

Client PTR/reverse-DNS records are resolved through DNS.

Configuration mode

```
service {
 ssh {
  disable-host-validation
 }
}
```

Use this command to specify that SSH should not resolve client PTR/reverse-DNS records via a reverse DNS (PTR) lookup. This process can be time consuming and cause long delays for clients trying to connect.

Use the `set` form of this command to specify that SSH should not resolve client PTR/reverse-DNS records via a reverse DNS (PTR) lookup.

Use the `delete` form of this command to restore the default configuration and allow reverse DNS lookups.

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

service ssh disable-password-authentication

Specifies that SSH users are not to be authenticated using passwords.

Syntax:
```
set service ssh disable-password-authentication
```

Syntax:
```
delete service ssh disable-password-authentication
```

Syntax:
```
show service ssh
```

Users are authenticated using passwords.

Configuration mode

```
service {
```

```
 ssh {
  disable-password-authentication
 }
}
```

Use this command to specify that SSH users are not to be authenticated using passwords. This is typically done in order for SSH users to be authenticated using shared public keys instead. Shared public key authentication is less susceptible to brute force guessing of common passwords. If password authentication is disabled then shared public keys must be configured for user authentication. For information on configuring public keys for user authentication see AT&T Vyatta Network Operating System Basic System Configuration Guide.

Use the `set` form of this command to specify that users are not to be authenticated by using passwords.

Use the `delete` form of this command to restore the default configuration and allow authentication by passwords.

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

service ssh listen-address <ipv4>

Configures access to SSH on a specific address.

Syntax:
```
set service ssh listen-address ipv4
```

Syntax:
```
delete service ssh listen-address ipv4
```

Syntax:
```
show service ssh listen-address
```

Requests to access SSH will be accepted on any system IP address.

ipv4

Multi-node. An IP address that the ssh service listens for connection requests on. The address must be assigned to an interface.

You can define more than one **`listen-address`** by creating multiple **`listen-address`** configuration nodes.

Configuration mode

```
service {
 ssh {
  listen-address ipv4
 }
}
```

Use this command to configure the system to accept requests for SSH access on specific addresses. This provides a way to limit access to the system.

Use the `set` form of this command to configure the system to accept requests for SSH access on specific addresses.

Use the `delete` form of this command to remove a listen-address.

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

service ssh port <port>

Specifies the port the system will use for the SSH service.

Syntax:
```
set service ssh port port
```

Syntax:
```
delete service ssh port
```

Syntax:
```
show service ssh port
```

The SSH service runs on port 22.

port
The port the system uses for the SSH service. The numbers range from 1 through 65534.

Configuration mode

```
service {
 ssh {
  port port
 }
}
```

Use this command to specify the port the system will use for the SSH service.

Use the `set` form of this command to specify the port the system will use for the SSH service.

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

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

service ssh protocol-version <version>

Specifies which versions of SSH are enabled.

Syntax:
```
set service ssh protocol-version version
```

Syntax:
```
delete service ssh protocol-version
```

Syntax:
```
show service ssh protocol-version
```

SSH version 2 alone is enabled.

version
Specifies which versions of SSH are enabled. Supported values are as follows:

v1
SSH version 1 alone is enabled.

v1
SSH version 1 alone is enabled.

v2
SSH version 2 alone is enabled. This is the recommended setting as **v1** is considered insecure.

all
Both SSH version 1 and SSH version 2 are both enabled.

Configuration mode

```
service {
 ssh {
  protocol-version version
 }
}
```

Use this command to specify which versions of SSH are enabled.

Use the `set` form of this command to specify which versions of SSH are enabled.

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

Use the `show` form of this command to view the protocol-version configuration.

Telnet

Telnet configuration

Configuring Telnet is optional, but creating the Telnet service will allow you to access the AT&T Vyatta vRouter remotely. The following table enables Telnet on the default port (port 23), as shown in the following figure.

Figure 2: Enabling Telnet access

To enable the Telnet service on the AT&T Vyatta vRouter, perform the following steps in configuration mode.

Table 2: Enabling Telnet access

Step	Command
Create the configuration node for the Telnet service.	`vyatta@R1# set service telnet`
Commit the information.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show service`
	` telnet {`
	` }`

Telnet Commands

service telnet

Configures Telnet as an access protocol on the system.

Syntax:
```
set service telnet
```

Syntax:
```
delete service telnet
```

Syntax:
```
show service telnet
```

Configuration mode

```
service {
 telnet {
 }
}
```

Use this command to configure the system to accept Telnet as an access service to the system.

Creating the Telnet configuration node enables Telnet as an access protocol. By default, the system uses port 23 for the Telnet service.

Use the `set` form of this command to create the Telnet configuration.

Use the `delete` form of this command to remove the Telnet configuration. If you delete the Telnet configuration node you will disable Telnet access to the system.

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

service telnet allow-root

Specifies that root logins are allowed on Telnet connections.

Syntax:
```
set service telnet allow-root
```

Syntax:
```
delete service telnet allow-root
```

Syntax:
```
show service telnet
```

Root logins are not allowed on Telnet connections.

Configuration mode

```
service {
 telnet {
  allow-root
 }
}
```

Use this command to specify that root logins are to be allowed on Telnet connections.

Use the `set` form of this command to specify that root logins are to be allowed on Telnet connections.

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

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

service telnet listen-address <address>

Configures access to Telnet on a specific address.

Syntax:
`set service telnet` **`listen-address`** *address*

Syntax:
`delete service telnet` **`listen-address`** *address*

Syntax:
`show service telnet`

Requests to access Telnet will be accepted on any system IP address.

`listen-address` ***address***
Multi-node. An IPv4 or IPv6 address that the telnet service listens for connection requests on. The address must be assigned to an interface.

You can define more than one **`listen-address`** by creating multiple **`listen-address`** configuration nodes.

Configuration mode

```
service {
 telnet {
  listen-address address
 }
}
```

Use this command to configure the system to accept requests for Telnet access on specific addresses. This provides a way to limit access to the system.

Use the `set` form of this command to configure the system to accept requests for Telnet access on specific addresses.

Use the `delete` form of this command to remove a listen-address.

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

service telnet port <port>

Specifies the port the system will use for the Telnet service.

Syntax:
`set service telnet port` *port*

Syntax:
`delete service telnet port`

Syntax:
`show service telnet port`

The default is port 23.

port

The port the system will use for the Telnet service. The range is 1 to 65534.

Configuration mode

```
service {
 telnet {
  port port
 }
}
```

Use this command to specify the port the system will use for the Telnet service.

Use the `set` form of this command to specify the port the system will use for the Telnet service.

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

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

system telnet allow-root

Enables root logins on Telnet connections.

Syntax:
```
set system telnet allow-root
```

Syntax:
```
delete system telnet allow-root
```

Syntax:
```
show service telnet
```

Root logins are not enabled on Telnet connections.

Configuration mode

```
system {
 telnet {
  allow-root
 }
}
```

Use this command to enable root logins on Telnet connections.

Use the `set` form of this command to enable root logins on Telnet connections.

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

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

Note: `set service telnet allow-root` command is deprecated. Use the above command to enable root logins on Telnet connections.

telnet <address>

Creates a terminal session to a Telnet server.

Syntax:
`telnet` *address*

address

Mandatory. The IP address or hostname of the Telnet server to connect to. The system connects through port 23 (the well-known port for the Telnet service).

Operational mode

Use this command to create a terminal session to a remote machine running a Telnet service.

The following example shows a telnet session being established to 192.168.1.77.

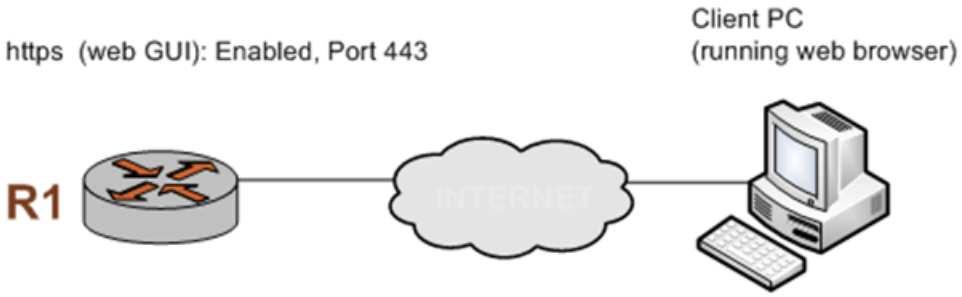
```
vyatta@R1:~$ telnet 192.168.1.77
Entering character mode
Escape character is '^]'.
Welcome to Vyatta
vyatta login:
```

Web GUI Access (HTTPS)

Web GUI access configuration

Configuring web GUI access is optional, but creating the HTTPS service will allow you to access the web GUI on the AT&T Vyatta vRouter remotely via a web browser. The following table enables HTTPS on the default port (port 443), as shown in the following figure.

Figure 3: Enabling web GUI access



To enable the HTTPS service on the AT&T Vyatta vRouter to provide access to the web GUI, perform the following steps in configuration mode.

Table 3: Enabling web GUI access

Step	Command
Create the configuration node for the HTTPS service.	`vyatta@R1# set service https`
Commit the information.	`vyatta@R1# commit`

Show the configuration.	`vyatta@R1# show service`
	` https {`
	` }`

Web GUI Access Commands

service https

Configures access to the web GUI.

Syntax:
```
set service https
```

Syntax:
```
delete service https
```

Syntax:
```
show service https
```

Configuration mode

```
service {
 https
}
```

Use this command to configure access to the web GUI via HTTPS (port 443). Once configured, the web GUI can be accessed by specifying one of the system IP addresses from a web browser.

When the HTTPS service is enabled, HTTP redirection is also enabled. If you want to disable HTTP redirection, use the `service https http-redirect` command.

Use the `set` form of this command to create the HTTPS configuration and enable access to the web GUI.

Use the `delete` form of this command to remove the HTTPS configuration. If you delete the HTTPS configuration node you will disable web GUI access to the system.

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

service https http-redirect

Enables or disables HTTP redirection.

Syntax:
```
set service https http-redirect { enable | disable }
```

Syntax:
```
delete service https http-redirect
```

Syntax:
```
show service https http-redirect
```

HTTP redirection is enabled.

`enable`
HTTP redirection is enabled.

`disable`
HTTP redirection is disabled.

Configuration mode

```
service {
```

```
 https
  http-redirect [enable|disable]
 }
}
```

Use this command to specify whether to enable or disable HTTP redirection.

When the AT&T Vyatta vRouter web GUI service (https) is enabled, HTTP redirection is enabled automatically. This allows the user to specify an HTTP URL rather than an HTTPS URL to connect to the web GUI. In certain circumstances this behavior is undesirable. Use this command to disable this behavior.

Use the `set` form of this command to enable or disable HTTP redirection.

Use the `delete` form of this command to restore the default behavior for HTTP redirection.

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

service https listen-address <ipv4>

Configures access to the web GUI on a specific address.

Syntax:
`set service https listen-address` *ipv4*

Syntax:
`delete service https listen-address` *ipv4*

Syntax:
`show service https listen-address`

Requests to access the web GUI will be accepted on any system IP address.

ipv4

Multi-node. An IP address that the HTTPS service listens for connection requests on. The address must be assigned to an interface.

You can define more than one **`listen-address`** by creating multiple **`listen-address`** configuration nodes.

Configuration mode

```
service {
 https {
  listen-address ipv4
 }
}
```

Use this command to configure the system to accept requests for web GUI access on specific addresses. This provides a way to limit access to the system.

Use the `set` form of this command to configure the system to accept requests for web GUI access on specific addresses.

Use the `delete` form of this command to remove a listen-address.

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

restart https

Restarts the HTTPS server.

Syntax:
`restart https`

Operational mode

Use this command to restart the HTTPS server.

The following example shows the output resulting from the `restart https` command.

```
vyatta@R1> restart https
Stopping web server: lighttpd.
Starting web server: lighttpd.
Stopping PAGER server
Starting PAGER server
Stopping API PAGER server
Starting API PAGER server
spawn-fcgi: child spawned successfully: PID: 4219
vyatta@R1>
```

NETCONF

NETCONF overview

NETCONF is a protocol that provides mechanisms for installing, manipulating, and deleting the configuration of network devices. It uses Extensible Markup Language (XML)-based data encoding for configuration data and protocol messages. The NETCONF operations are realized as remote procedure calls (RPCs).

Refer to RFC 6241, *Network Configuration Protocol (NETCONF),* at https://tools.ietf.org/html/rfc6241 for more information.

NETCONF on the AT&T Vyatta vRouter

On the AT&T Vyatta vRouter, NETCONF is used within an SSH session through the SSH connection protocol. This mapping allows NETCONF to be run from a secure shell session by a user or an application. This mapping also makes sure that NETCONF complies with SSH IPv6.

On the AT&T Vyatta vRouter, NETCONF is intended as a machine interface for management software and not intended as a user interface.

Refer to RFC 6242, *Using the NETCONF Protocol over Secure Shell (SSH)*, at https://tools.ietf.org/html/rfc6242 for more information on using the NETCONF configuration protocol over SSH.

NETCONF capabilities supported on the AT&T Vyatta vRouter

A NETCONF capability is a set of functions that supplements the base NETCONF specification. The capability is identified by a uniform resource identifier (URI). Capabilities augment the base operations of the device, describing both additional operations and the content that is allowed inside the operations. The client discovers the capabilities of the server and uses any additional operations, parameters, and content that are defined by those capabilities.

Following are the NETCONF capabilities that are supported on the AT&T Vyatta vRouter:

- capability:candidate 1.0
- capability:startup 1.0
- capability:rollback-on-error 1.0
- capability:validate 1.1

Refer to RFC 6241, *Network Configuration Protocol (NETCONF),* at https://tools.ietf.org/html/rfc6241 for more information on these capabilities.

Initiating a NETCONF session

To allow an inbound NETCONF session request from a remote system to be accepted, use the commands that are shown in the following table.

Table 4: Initiating a NETCONF session

Step	Command
Initiate NETCONF.	`vyatta@R1# set service netconf`

Step	Command
Define the port for NETCONF over SSH.	`vyatta@R1# set service ssh port 830`
Commit the change.	`vyatta@R1# commit`

YANG model for NETCONF monitoring

The `<get-schema>` operation is supported on the AT&T Vyatta vRouter to query and retrieve schema information and NETCONF state information from a NETCONF server.

Refer to RFC 6022, *YANG Module for NETCONF Monitoring*, at https://tools.ietf.org/html/rfc6022 for more information on using `<get-schema>`.

Verifying Connections and Retrieving Operational Data Using NETCONF

ping

The **ping** command displays whether a destination responded and how long the destination took to receive a reply. If an error occurs in the delivery to the destination, the command displays an error message.

Table 5: Information about the ping command

Parameter	Description

Sample XML program	`<ping xmlns="urn:vyatta.com:mgmt:vyatta-op">`
	`<host>127.0.0.1</host>`
	`<count>5</count>`
	`<ttl>3</ttl>`
	`</ping>`

XML tags	• host: IP address you want to ping. • count: Number of packets with which you are pinging. • ttl: Time to live (ttl) in an IP packet in seconds that tells a network router whether the packet has been in the network too long and should be discarded. By default, the TTL value is 255.

Sample rpc#reply	`<tx-packet-count>5</tx-packet-count><rx-packe`
	`t-count>5</rx-packet-count><min-delay>54</`
	`min-delay><average-delay>62</average-delay><max-de`
	`lay>74</max-delay>netconf>`

interface

The `interface` command displays information about an interface name. The command output displays all the IP addresses that are associated with the interface, administrator status, operational status, and description of the interface.

Table 6: Information about the interface command

Parameter	Description

Sample XML program	`<interface xmlns="urn:vyatta.com:mgmt:vyatta-op">`
	`<name>br0</name>`
	`</interface>`

XML tags	name: Name of an interface

Parameter	Description
Sample rpc#reply	<address> <ip>15.15.15.15/24</ip> </address> <address> <ip>2020::/64</ip> </address> <address> <ip>2001::15/64</ip> </address> <address> <ip>3001::211:22ff:fe33:4455/64</ip> </address> <admin-status>up</admin-status> <oper-status>up</oper-status> <description>sample bridge</description>

route

The `route` command displays information about the path taken to a particular destination address.

Table 7: Information about the route command

Parameter	Description
Sample XML program	<route xmlns="urn:vyatta.com:mgmt:vyatta-op"> <destination>192.168.14.0/24</ destination> </route>
XML tags	• destination (optional): IP address or IP prefix • family: ipv4 (default) or ipv6 **Note:** When the destination is not present, the entire route table for the specified family is returned as the output.
Sample rpc#reply	<route> <destination>192.168.14.0 </destination> <path> <entry>1</entry> <nexthop>31.31.31.32</nexthop> <device>dp0p256p1</device> </path> </route>

Retrieving statistics using NETCONF

Overview

You can retrieve AT&T Vyatta vRouter statistics of all types by using NETCONF. Some of the statistics are also available from the CLI. The configuration model is a subtree of the overall YANG tree, which is included in the ISO distribution for the AT&T Vyatta vRouter.

Use **ssh** to obtain NETCONF information in XML format from the vrouter as follows.

1. Configure **set service ssh** and **set service netconf** on the AT&T Vyatta vRouter.
2. From a remote system, use SSH as follows to access and sign in to the AT&T Vyatta vRouter, when prompted.

```
ssh vyatta@<ip-address> -s netconf
```

3. Send a capabilities list enclosed within a **<hello>...<hello/>** element.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <capabilities>
   <capability>urn:ietf:params:netconf:base:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>
   </capabilities>
</hello>
]]>]]>
```

4. Respond to the AT&T Vyatta vRouter hello element with a hello message and capabilities list. End the list with **]]>]]>** to inform SSH that the XML element is completed and should be sent.

The following sections provide XML examples for different categories of statistics and other information.

CPU, memory usage, uptime, and version information

The following examples provide AT&T Vyatta vRouter system metrics, including CPU, memory usage, uptime, and version.

CPU information

The XML examples in this section provide CPU information.

The following example shows an RPC request for CPU information.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
        <system>
         <state>
         <processor xmlns="urn:vyatta.com:mgmt:vyatta-system:1">
         </processor>
         </state>
        </system>
    </filter>
  </get>
</rpc>]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <data>
        <system xmlns="urn:vyatta.com:mgmt:vyatta-system:1">
            <state>
                <processor>
                    <utilization>
                        <cpu>0</cpu>
                        <guest>0.00</guest>
                        <idle>98.41</idle>
                        <iowait>0.11</iowait>
                        <irq>0.00</irq>
                        <nice>0.00</nice>
                        <niced-guest>0.00</niced-guest>
                        <soft>0.00</soft>
                        <steal>0.07</steal>
                        <sys>1.39</sys>
                        <user>0.01</user>
                    </utilization>
                    <utilization>
                        <cpu>1</cpu>
                        <guest>0.00</guest>
                        <idle>97.27</idle>
                        <iowait>0.06</iowait>
                        <irq>0.00</irq>
                        <nice>0.00</nice>
                        <niced-guest>0.00</niced-guest>
                        <soft>0.00</soft>
                        <steal>0.35</steal>
                        <sys>1.87</sys>
                        <user>0.44</user>
                    </utilization>
                    <utilization>
                        <cpu>all</cpu>
                        <guest>0.00</guest>
                        <idle>97.85</idle>
                        <iowait>0.09</iowait>
                        <irq>0.00</irq>
                        <nice>0.00</nice>
                        <niced-guest>0.00</niced-guest>
                        <soft>0.00</soft>
                        <steal>0.21</steal>
                        <sys>1.63</sys>
                        <user>0.23</user>
                    </utilization>
                </processor>
      </state>
    </system>
  </data>
</rpc-reply>
]]>]]>
```

Memory information

The XML examples in this section provide memory information.

The following example shows an RPC request for memory information.

```
 <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <filter type="subtree">
            <system>
                <state>
                    <memory xmlns="urn:vyatta.com:mgmt:vyatta-system:1">
                    </memory>
                </state>
            </system>
```

```
        </filter>
    </get>
</rpc>]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <data>
        <system xmlns="urn:vyatta.com:mgmt:vyatta-system:1">
            <state>
                <memory>
                    <active>444076032</active>
                    <active-file>141246464</active-file>
                    <anonymous>
                        <active>302829568</active>
                        <inactive>8691712</inactive>
                    </anonymous>
                    <anonymous-pages>274878464</anonymous-pages>
                    <available-memory>2527830016</available-memory>
                    <bounce>0</bounce>
                    <buffers>56684544</buffers>
                    <cached>272482304</cached>
                    <commit-limit>1555976192</commit-limit>
                    <dirty>122880</dirty>
                    <free-memory>2399358976</free-memory>
                    <huge-pages>
                        <anonymous>102760448</anonymous>
                        <free>0</free>
                        <reserved>0</reserved>
                        <shared-memory>0</shared-memory>
                        <size>2097152</size>
                        <surplus>0</surplus>
                        <total>494</total>
                    </huge-pages>
                    <inactive>187539456</inactive>
                    <inactive-file>178847744</inactive-file>
                    <kernel-stack>3047424</kernel-stack>
                    <mapped>75395072</mapped>
                    <memory-locked>0</memory-locked>
                    <nfs-unstable>0</nfs-unstable>
                    <page-tables>6393856</page-tables>
                    <shared-memory>9076736</shared-memory>
                    <slab>53288960</slab>
                    <slab-non-reclaimable>16318464</slab-non-reclaimable>
                    <slab-reclaimable>36970496</slab-reclaimable>
                    <swap-cached>0</swap-cached>
                    <swap-free>0</swap-free>
                    <swap-total>0</swap-total>
                    <total-committed-memory>1081114624</total-committed-memory>
                    <total-memory>4147949568</total-memory>
                    <unevictable>0</unevictable>
                    <vmalloc-chunk>0</vmalloc-chunk>
                    <vmalloc-total>35184372087808</vmalloc-total>
                    <vmalloc-used>0</vmalloc-used>
                    <writeback>0</writeback>
                    <writeback-tmp>0</writeback-tmp>
                </memory>
            </state>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

Uptime information

The XML examples in this section provide uptime information.

The following example shows an RPC request for uptime information.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <filter type="subtree">
            <system>
                <state>
                    <times xmlns="urn:vyatta.com:mgmt:vyatta-system:1">
                    </times>
                </state>
            </system>
        </filter>
    </get>
</rpc>
]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <data>
        <system xmlns="urn:vyatta.com:mgmt:vyatta-system:1">
            <state>
                <times>
                    <uptime>3614</uptime>
                </times>
            </state>
        </system>
    </data>
</rpc-reply>
]]>]]>
```

Version information

The XML examples in this section provide version information.

The following example shows an RPC request for version information.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
        <system>
         <state>
   <platform>
          <os-version xmlns="urn:vyatta.com:mgmt:vyatta-system:1"></os-version>
         </platform>
        </state>
        </system>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <system xmlns="urn:vyatta.com:mgmt:vyatta-system:1">
      <state>
        <platform>
```

```
          <os-version>17.2.0</os-version>
        </platform>
      </state>
    </system>
  </data>
</rpc-reply>
]]>]]>
```

CPU, memory, network device, and block device information for each hosted VM

The following sample output reports on CPU, memory, network device, and block device information for each hosted VM.

```
<?xml version="1.0" encoding="UTF-8" ?>
<guest>
    <net-devices>
        <net>vnet0</net>
        <tx>
            <drop>0</drop>
            <errs>0</errs>
            <pkts>13</pkts>
            <bytes>2358</bytes>
        </tx>
        <rx>
            <drop>0</drop>
            <pkts>84</pkts>
            <errs>0</errs>
            <bytes>9118</bytes>
        </rx>
    </net-devices>
    <vcpu-devices>
        <current>1</current>
        <cpu>
            <state>1</state>
            <wait>0</wait>
            <time>47750000000</time>
            <sock>0</sock>
        </cpu>
        <maximum>1</maximum>
    </vcpu-devices>
    <name>debian8</name>
    <block-devices>
        <rd>
            <reqs>4870</reqs>
            <bytes>63318016</bytes>
            <times>267824192</times>
        </rd>
        <wr>
            <times>153279006</times>
            <bytes>1794048</bytes>
            <reqs>393</reqs>
        </wr>
        <name>vda</name>
    </block-devices>
    <block-devices>
        <name>hda</name>
        <wr>
            <bytes>0</bytes>
            <reqs>0</reqs>
            <times>0</times>
        </wr>
        <rd>
            <times>49092</times>
```

```
            <bytes>152</bytes>
            <reqs>4</reqs>
        </rd>
    </block-devices>
    <memory>
        <actual>1048576</actual>
    </memory>
    <cpu>
        <system>2050000000</system>
        <time>48364864062</time>
        <user>560000000</user>
    </cpu>
</guest>
```

DHCP server statistics

The sample XML output in this section provides DHCP server statistics for single and multiple routing instances.

The following sample requests DHCP server leases for the default routing instance.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="6">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
     <service>
      <dhcp-server>
      <state>
       <leases xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server:1"></leases>
      </state>
      </dhcp-server>
      </service>
    </filter>
  </get>
</rpc>]]>]]>
```

The following sample output shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="6">
  <data>
    <service xmlns="urn:vyatta.com:mgmt:vyatta-services:1">
      <dhcp-server xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server:1">
        <state>
          <leases>
            <ip-address>192.168.111.101</ip-address>
            <host>vyatta</host>
            <hw-address>52:54:00:e3:c3:24</hw-address>
            <lease-expiration>2017-03-15T17:19:21+00:00</lease-expiration>
            <pool>net111</pool>
          </leases>
          <leases>
            <ip-address>192.168.111.200</ip-address>
            <host>vyatta</host>
            <hw-address>52:54:00:b0:18:91</hw-address>
            <lease-expiration>2017-03-15T17:19:21+00:00</lease-expiration>
            <pool>net111</pool>
          </leases>
        </state>
      </dhcp-server>
    </service>
  </data>
</rpc-reply>
]]>]]>
```

The following sample requests DHCP server leases for the TEN routing instance.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
      <routing>
        <routing-instance>
          <instance-name>TEN</instance-name>
            <service>
             <dhcp-server>
              <state xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server-routing-instance:1"></
state>
             </dhcp-server>
            </service>
      </routing-instance>
      </routing>
    </filter>
  </get>
</rpc>]]>]]>
```

The following sample output shows the RPC reply.

```
 <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
      <routing>
        <routing-instance>
          <instance-name>TEN</instance-name>
            <service>
             <dhcp-server>
              <state xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server-routing-instance:1"></
state>
             </dhcp-server>
            </service>
      </routing-instance>
      </routing>
    </filter>
  </get>
</rpc>]]>]]>
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <routing xmlns="urn:vyatta.com:mgmt:vyatta-routing-v1:1">
      <routing-instance>
        <instance-name>TEN</instance-name>
        <service>
          <dhcp-server xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server-routing-instance:1">
            <state>
              <leases>
                <ip-address>192.168.100.133</ip-address>
                <host>vyatta</host>
                <hw-address>52:54:00:87:a6:d6</hw-address>
                <lease-expiration>2017-03-23T14:13:21+00:00</lease-expiration>
                <pool>RT-TEN</pool>
              </leases>
              <leases>
                <ip-address>192.168.100.134</ip-address>
                <host>vyatta</host>
                <hw-address>52:54:00:42:1a:62</hw-address>
                <lease-expiration>2017-03-23T16:50:00+00:00</lease-expiration>
                <pool>RT-TEN</pool>
              </leases>
            </state>
          </dhcp-server>
        </service>
      </routing-instance>
    </routing>
```

```
  </data>
</rpc-reply>
]]>]]>
```

The following sample requests DHCP server leases for all non default routing instances.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="6">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
     <routing>
      <routing-instance>
        <service>
        <dhcp-server>
        <state xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server-routing-instance:1"></state>
         </dhcp-server>
        </service>
      </routing-instance>
     </routing>
    </filter>
  </get>
</rpc>]]>]]>
```

The following sample shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="6">
  <data>
    <routing xmlns="urn:vyatta.com:mgmt:vyatta-routing-v1:1">
      <routing-instance>
        <service>
          <dhcp-server xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server-routing-instance:1">
            <state>
              <leases>
                <ip-address>192.168.102.130</ip-address>
                <host>vyatta</host>
                <hw-address>52:54:00:10:5f:54</hw-address>
                <lease-expiration>2017-03-22T21:01:43+00:00</lease-expiration>
                <pool>my102net</pool>
              </leases>
            </state>
          </dhcp-server>
        </service>
      </routing-instance>
      <routing-instance>
        <service>
          <dhcp-server xmlns="urn:vyatta.com:mgmt:vyatta-service-dhcp-server-routing-instance:1">
            <state>
              <leases>
                <ip-address>192.168.100.133</ip-address>
                <host>vyatta</host>
                <hw-address>52:54:00:87:a6:d6</hw-address>
                <lease-expiration>2017-03-22T19:24:28+00:00</lease-expiration>
                <pool>RT-TEN</pool>
              </leases>
              <leases>
                <ip-address>192.168.100.134</ip-address>
                <host>vyatta</host>
                <hw-address>52:54:00:42:1a:62</hw-address>
                <lease-expiration>2017-03-22T19:24:28+00:00</lease-expiration>
                <pool>RT-TEN</pool>
              </leases>
            </state>
          </dhcp-server>
        </service>
      </routing-instance>
    </routing>
```

```
  </data>
</rpc-reply>
]]>]]>
```

Firewall, PBR, and NAT information

The following XML examples provide firewall, PBR, and NAT information.

The following example is an RPC request for the firewall state information associated with the dp0s10 interface.

```
# FW interface filter - dataplane - note change <tagnode> value to match
# the interface you want state for
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
    <get
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <filter type="subtree">
            <interfaces
                xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
                <dataplane
                    xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-dataplane:1">
                    <tagnode>dp0s10</tagnode>
                    <firewall
                        xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
                        <state></state>
                    </firewall>
                </dataplane>
            </interfaces>
        </filter>
    </get>
</rpc>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
      <dataplane xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-dataplane:1">
        <tagnode>dp0s10</tagnode>
        <firewall xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
          <state>
            <in>
              <name>
                <group-name>FW1</group-name>
                <rule>
                  <rule-number>10</rule-number>
                  <bytes>0</bytes>
                  <packets>0</packets>
                </rule>
                <rule>
                  <rule-number>100</rule-number>
                  <bytes>0</bytes>
                  <packets>0</packets>
                </rule>
                <rule>
                  <rule-number>10000</rule-number>
                  <bytes>0</bytes>
                  <packets>0</packets>
                </rule>
              </name>
              <name>
                <group-name>fw1</group-name>
                <rule>
                  <rule-number>10</rule-number>
                  <bytes>0</bytes>
```

```
                <packets>0</packets>
              </rule>
              <rule>
                <rule-number>20</rule-number>
                <bytes>816</bytes>
                <packets>8</packets>
              </rule>
            </name>
          </in>
          <local>
            <name>
              <group-name>LOCAL2</group-name>
              <rule>
                <rule-number>20</rule-number>
                <bytes>0</bytes>
                <packets>0</packets>
              </rule>
            </name>
          </local>
          <out/>
        </state>
      </firewall>
    </dataplane>
  </interfaces>
 </data>
</rpc-reply>
]]>]]>
```

The following example is an RPC request for the firewall state information of dp0s10 vif 100.

```
# FW VIF interface filter - dataplane - note change first <tagnode> value to
# match the interface you want state for, and the second to the vif number
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
       <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
         <dataplane xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-dataplane:1">
           <tagnode>dp0s10</tagnode>
           <vif>
              <tagnode>100</tagnode>
              <firewall xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
                  <state></state>
              </firewall>
           </vif>
         </dataplane>
       </interfaces>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
         <dataplane xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-dataplane:1">
            <tagnode>dp0s10</tagnode>
            <vif>
               <tagnode>100</tagnode>
               <firewall xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
                  <state>
                     <in>
                        <name>
                           <group-name>FW1</group-name>
```

```
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>100</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>10000</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </in>
                  <local />
                  <out>
                     <name>
                        <group-name>FW1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>100</rule-number>
                           <bytes>738</bytes>
                           <packets>7</packets>
                        </rule>
                        <rule>
                           <rule-number>10000</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </out>
               </state>
            </firewall>
         </vif>
      </dataplane>
   </interfaces>
</data>
</rpc-reply>
]]>]]>
```

The following example is an RPC request for the PBR state on the dp0s10 interface.

```
# PBR interface filter - dataplane - note change <tagnode> value to match
# the interface you want state for
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
      <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
         <dataplane xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-dataplane:1">
           <tagnode>dp0s10</tagnode>
           <policy xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-policy:1">
              <route xmlns="urn:vyatta.com:mgmt:vyatta-policy-pbr:1">
                 <pbr-state></pbr-state>
              </route>
           </policy>
         </dataplane>
      </interfaces>
    </filter>
  </get>
```

```
</rpc>]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
         <dataplane xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-  dataplane:1">
            <tagnode>dp0s10</tagnode>
            <policy xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-policy:1">
               <route xmlns="urn:vyatta.com:mgmt:vyatta-policy-pbr:1">
                  <pbr-state>
                     <name>
                        <group-name>PBR-1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </pbr-state>
               </route>
            </policy>
         </dataplane>
      </interfaces>
   </data>
</rpc-reply>
]]>]]>
```

The following example is an RPC request for the state of the bridge interface.

```
# interface filter - bridge - note change <tagnode> value to match
# the interface you want state for
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
       <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
          <bridge xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-bridge:1">
             <tagnode>br1</tagnode>
             <firewall xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
                <state></state>
             </firewall>
          </bridge>
       </interfaces>
    </filter>
  </get>
</rpc>]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
         <bridge xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-  bridge:1">
            <tagnode>br1</tagnode>
            <firewall xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
               <state>
                  <in>
                     <name>
                        <group-name>FW1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
```

```
                        </rule>
                        <rule>
                           <rule-number>100</rule-number>
                           <bytes>5696</bytes>
                           <packets>40</packets>
                        </rule>
                        <rule>
                           <rule-number>10000</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </in>
                  <l2>
                     <name>
                        <group-name>FW1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>100</rule-number>
                           <bytes>5696</bytes>
                           <packets>40</packets>
                        </rule>
                        <rule>
                           <rule-number>10000</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </l2>
                  <local />
                  <out>
                     <name>
                        <group-name>FW1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>100</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>10000</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </out>
               </state>
            </firewall>
         </bridge>
      </interfaces>
   </data>
</rpc-reply>
]]>]]>
```

The following example is an RPC request for the state of the loopback interface.

```
# interface filter - loopback
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
```

```
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
      <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
        <loopback xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-loopback:1">
           <tagnode>lo</tagnode>
           <firewall xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
               <state></state>
           </firewall>
        </loopback>
      </interfaces>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
         <loopback xmlns="urn:vyatta.com:mgmt:vyatta-interfaces-loopback:1">
            <tagnode>lo</tagnode>
            <firewall xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
               <state>
                  <local>
                     <name>
                        <group-name>LOCAL1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>20</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </local>
               </state>
            </firewall>
         </loopback>
      </interfaces>
   </data>
</rpc-reply>
]]>]]>
```

The following example is an RPC request for the zone policy state.

```
# zone policy state
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
        <security xmlns="urn:vyatta.com:mgmt:vyatta-security:1">
           <zone-policy xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
             <state></state>
           </zone-policy>
        </security>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <security xmlns="urn:vyatta.com:mgmt:vyatta-security:1">
         <zone-policy xmlns="urn:vyatta.com:mgmt:vyatta-security-firewall:1">
            <state>
               <zone>
                  <input-zone-name>INSIDE</input-zone-name>
                  <to>
                     <output-zone-name>OUTSIDE</output-zone-name>
                     <name>
                        <group-name>FW1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>100</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>10000</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </to>
               </zone>
               <zone>
                  <input-zone-name>OUTSIDE</input-zone-name>
                  <to>
                     <output-zone-name>INSIDE</output-zone-name>
                     <name>
                        <group-name>FW1</group-name>
                        <rule>
                           <rule-number>10</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>100</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                        <rule>
                           <rule-number>10000</rule-number>
                           <bytes>0</bytes>
                           <packets>0</packets>
                        </rule>
                     </name>
                  </to>
               </zone>
            </state>
         </zone-policy>
      </security>
   </data>
</rpc-reply>
]]>]]>
```

The following example is an RPC request for the NAT state.

```
# NAT
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
```

```
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
        <service xmlns="urn:vyatta.com:mgmt:vyatta-services:1">
           <nat xmlns="urn:vyatta.com:mgmt:vyatta-service-nat:1">
               <state></state>
           </nat>
        </service>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
      <service xmlns="urn:vyatta.com:mgmt:vyatta-services:1">
         <nat xmlns="urn:vyatta.com:mgmt:vyatta-service-nat:1">
            <state>
               <destination>
                  <rule>
                     <rule-number>12</rule-number>
                     <bytes>0</bytes>
                     <packets>0</packets>
                  </rule>
               </destination>
               <ipv6-to-ipv4>
                  <rule>
                     <rule-number>10</rule-number>
                     <bytes>0</bytes>
                     <packets>0</packets>
                  </rule>
               </ipv6-to-ipv4>
               <source>
                  <rule>
                     <rule-number>11</rule-number>
                     <bytes>0</bytes>
                     <packets>0</packets>
                  </rule>
               </source>
            </state>
         </nat>
      </service>
   </data>
</rpc-reply>
]]>]]>
```

Interface statistics

The following examples provide interface statistics for data plane, vhost, and tunnel interfaces, including Rx bytes, Rx packets, Rx errors, Tx bytes, Tx packets, Tx errors, and Tx dropped.

The following example shows an RPC request for interface statistics.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
        <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
        <statistics xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
        </statistics>
        </interfaces>
    </filter>
  </get>
```

```
</rpc>]]>]]>
```

The following example shows the RPC reply.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <data>
    <interfaces xmlns="urn:vyatta.com:mgmt:vyatta-interfaces:1">
      <statistics>
        <interface>
          <name>dp0s3</name>
          <receive-statistics>
            <bytes>695635</bytes>
            <dropped>0</dropped>
            <errors>0</errors>
            <multicast>9996</multicast>
            <oversized-packets>0</oversized-packets>
            <packets>11287</packets>
          </receive-statistics>
          <transmit-statistics>
            <bytes>568822</bytes>
            <carrier-errors>0</carrier-errors>
            <collisions>0</collisions>
            <dropped>0</dropped>
            <errors>0</errors>
            <packets>1239</packets>
          </transmit-statistics>
          <type>dataplane</type>
        </interface>
      </statistics>
    </interfaces>
  </data>
</rpc-reply>
]]>]]>
```

QoS information

The XML examples in this section provide QoS information.

The following sample XML output shows an RPC request for QoS information.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="6">
  <get xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <filter type="subtree">
       <policy xmlns="urn:vyatta.com:mgmt:vyatta-policy-qos:1">
         <qos xmlns="urn:vyatta.com:mgmt:vyatta-policy-qos:1">
            <state xmlns="urn:vyatta.com:mgmt:vyatta-policy-qos:1"/>
         </qos>
       </policy>
    </filter>
  </get>
</rpc>
]]>]]>
```

The following sample XML output shows one possible RPC reply. The structure and content of the RPC reply will depend on the QoS configuration of the system that is interrogated.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
 xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:d9ffce02-995e-4c4c-
a502-10c1c8105ab5">
  <data>
    <policy xmlns="urn:vyatta.com:mgmt:vyatta-policy:1">
      <qos xmlns="urn:vyatta.com:mgmt:vyatta-policy-qos:1">
        <state>
```

```
          <if-list>
            <ifname>dp0s4</ifname>
            <shaper>
              <subport-list>
                <subport>0</subport>
                <pipe-list>
                  <pipe>0</pipe>
                  <dscp-to-queue-map>
                    <dscp>0</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>1</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>2</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>3</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>4</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>5</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>6</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>7</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>8</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>9</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>10</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>11</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
```

```
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>12</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>13</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>14</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>15</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>16</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>17</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>18</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>19</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>20</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>21</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>22</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>23</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>24</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
```

```
                <dscp-to-queue-map>
                  <dscp>25</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>26</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>27</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>28</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>29</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>30</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>31</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>32</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>33</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>34</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>35</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>36</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>37</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
```

```
                  <dscp>38</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>39</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>40</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>41</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>42</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>43</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>44</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>45</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>46</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>47</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>48</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>49</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>50</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>51</dscp>
```

```
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>52</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>53</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>54</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>55</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>56</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>57</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>58</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>59</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>60</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>61</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>62</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>63</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <qos-class>0</qos-class>
                <qos-profile>default-profile</qos-profile>
                <token-bucket-rate>1250000000</token-bucket-rate>
```

```
                <token-bucket-size>5000000</token-bucket-size>
                <traffic-class-period>10</traffic-class-period>
                <traffic-class-queues-list>
                  <traffic-class>0</traffic-class>
                  <queue-statistics>
                    <queue>0</queue>
                    <bytes>0</bytes>
                    <dropped>0</dropped>
                    <dscp-values>
                      <dscp>48</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>49</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>50</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>51</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>52</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>53</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>54</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>55</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>56</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>57</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>58</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>59</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>60</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>61</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>62</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>63</dscp>
                    </dscp-values>
                    <packets>0</packets>
                    <qlen>0</qlen>
                    <random-drop>0</random-drop>
                  </queue-statistics>
                </traffic-class-queues-list>
                <traffic-class-queues-list>
                  <traffic-class>1</traffic-class>
                  <queue-statistics>
                    <queue>0</queue>
                    <bytes>0</bytes>
```

```
        <dropped>0</dropped>
        <dscp-values>
          <dscp>32</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>33</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>34</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>35</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>36</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>37</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>38</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>39</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>40</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>41</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>42</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>43</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>44</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>45</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>46</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>47</dscp>
        </dscp-values>
        <packets>0</packets>
        <qlen>0</qlen>
        <random-drop>0</random-drop>
      </queue-statistics>
    </traffic-class-queues-list>
    <traffic-class-queues-list>
      <traffic-class>2</traffic-class>
      <queue-statistics>
        <queue>0</queue>
        <bytes>0</bytes>
        <dropped>0</dropped>
        <dscp-values>
          <dscp>16</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>17</dscp>
        </dscp-values>
```

```
                    <dscp-values>
                      <dscp>18</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>19</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>20</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>21</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>22</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>23</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>24</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>25</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>26</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>27</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>28</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>29</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>30</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>31</dscp>
                    </dscp-values>
                    <packets>0</packets>
                    <qlen>0</qlen>
                    <random-drop>0</random-drop>
                  </queue-statistics>
                </traffic-class-queues-list>
                <traffic-class-queues-list>
                  <traffic-class>3</traffic-class>
                  <queue-statistics>
                    <queue>0</queue>
                    <bytes>0</bytes>
                    <dropped>0</dropped>
                    <dscp-values>
                      <dscp>0</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>1</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>2</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>3</dscp>
                    </dscp-values>
                    <dscp-values>
```

```
            <dscp>4</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>5</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>6</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>7</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>8</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>9</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>10</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>11</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>12</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>13</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>14</dscp>
          </dscp-values>
          <dscp-values>
            <dscp>15</dscp>
          </dscp-values>
          <packets>0</packets>
          <qlen>0</qlen>
          <random-drop>0</random-drop>
        </queue-statistics>
      </traffic-class-queues-list>
      <traffic-class-rates>
        <traffic-class>0</traffic-class>
        <rate>1250000000</rate>
      </traffic-class-rates>
      <traffic-class-rates>
        <traffic-class>1</traffic-class>
        <rate>1250000000</rate>
      </traffic-class-rates>
      <traffic-class-rates>
        <traffic-class>2</traffic-class>
        <rate>1250000000</rate>
      </traffic-class-rates>
      <traffic-class-rates>
        <traffic-class>3</traffic-class>
        <rate>1250000000</rate>
      </traffic-class-rates>
      <weighted-round-robin-weights>
        <queue>0</queue>
        <weight>1</weight>
      </weighted-round-robin-weights>
      <weighted-round-robin-weights>
        <queue>1</queue>
        <weight>1</weight>
      </weighted-round-robin-weights>
      <weighted-round-robin-weights>
        <queue>2</queue>
```

```
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>3</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>4</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>5</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>6</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>7</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>8</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>9</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>10</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>11</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>12</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>13</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>14</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>15</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                </pipe-list>
                <rules/>
                <subport-name>dp0s4</subport-name>
                <traffic-class-list>
                  <traffic-class>0</traffic-class>
                  <bytes>0</bytes>
                  <dropped>0</dropped>
                  <packets>0</packets>
                  <random-drop>0</random-drop>
                </traffic-class-list>
                <traffic-class-list>
                  <traffic-class>1</traffic-class>
```

```
                <bytes>0</bytes>
                <dropped>0</dropped>
                <packets>0</packets>
                <random-drop>0</random-drop>
              </traffic-class-list>
              <traffic-class-list>
                <traffic-class>2</traffic-class>
                <bytes>0</bytes>
                <dropped>0</dropped>
                <packets>0</packets>
                <random-drop>0</random-drop>
              </traffic-class-list>
              <traffic-class-list>
                <traffic-class>3</traffic-class>
                <bytes>0</bytes>
                <dropped>0</dropped>
                <packets>0</packets>
                <random-drop>0</random-drop>
              </traffic-class-list>
            </subport-list>
          </shaper>
        </if-list>
        <if-list>
          <ifname>dp0s5</ifname>
          <shaper>
            <subport-list>
              <subport>0</subport>
              <pipe-list>
                <pipe>0</pipe>
                <dscp-to-queue-map>
                  <dscp>0</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>1</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>2</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>3</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>4</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>5</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>6</dscp>
                  <queue>0</queue>
                  <traffic-class>3</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>7</dscp>
```

```
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>8</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>9</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>10</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>11</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>12</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>13</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>14</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>15</dscp>
                    <queue>0</queue>
                    <traffic-class>3</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>16</dscp>
                    <queue>0</queue>
                    <traffic-class>2</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>17</dscp>
                    <queue>0</queue>
                    <traffic-class>2</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>18</dscp>
                    <queue>0</queue>
                    <traffic-class>2</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>19</dscp>
                    <queue>0</queue>
                    <traffic-class>2</traffic-class>
                  </dscp-to-queue-map>
                  <dscp-to-queue-map>
                    <dscp>20</dscp>
                    <queue>0</queue>
```

```
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>21</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>22</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>23</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>24</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>25</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>26</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>27</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>28</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>29</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>30</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>31</dscp>
                  <queue>0</queue>
                  <traffic-class>2</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>32</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>33</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
```

```
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>34</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>35</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>36</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>37</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>38</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>39</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>40</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>41</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>42</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>43</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>44</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>45</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>46</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
```

```
                <dscp-to-queue-map>
                  <dscp>47</dscp>
                  <queue>0</queue>
                  <traffic-class>1</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>48</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>49</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>50</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>51</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>52</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>53</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>54</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>55</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>56</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>57</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>58</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>59</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
```

```
                  <dscp>60</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>61</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>62</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <dscp-to-queue-map>
                  <dscp>63</dscp>
                  <queue>0</queue>
                  <traffic-class>0</traffic-class>
                </dscp-to-queue-map>
                <qos-class>0</qos-class>
                <qos-profile>profile-2</qos-profile>
                <token-bucket-rate>1250000000</token-bucket-rate>
                <token-bucket-size>5000000</token-bucket-size>
                <traffic-class-period>10</traffic-class-period>
                <traffic-class-queues-list>
                  <traffic-class>0</traffic-class>
                  <queue-statistics>
                    <queue>0</queue>
                    <bytes>0</bytes>
                    <dropped>0</dropped>
                    <dscp-values>
                      <dscp>48</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>49</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>50</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>51</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>52</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>53</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>54</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>55</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>56</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>57</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>58</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>59</dscp>
                    </dscp-values>
```

```
                    <dscp-values>
                      <dscp>60</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>61</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>62</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>63</dscp>
                    </dscp-values>
                    <packets>0</packets>
                    <qlen>0</qlen>
                    <random-drop>0</random-drop>
                  </queue-statistics>
                </traffic-class-queues-list>
                <traffic-class-queues-list>
                  <traffic-class>1</traffic-class>
                  <queue-statistics>
                    <queue>0</queue>
                    <bytes>0</bytes>
                    <dropped>0</dropped>
                    <dscp-values>
                      <dscp>32</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>33</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>34</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>35</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>36</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>37</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>38</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>39</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>40</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>41</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>42</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>43</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>44</dscp>
                    </dscp-values>
                    <dscp-values>
                      <dscp>45</dscp>
                    </dscp-values>
                    <dscp-values>
```

```
                        <dscp>46</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>47</dscp>
                      </dscp-values>
                      <packets>0</packets>
                      <qlen>0</qlen>
                      <random-drop>0</random-drop>
                    </queue-statistics>
                  </traffic-class-queues-list>
                  <traffic-class-queues-list>
                    <traffic-class>2</traffic-class>
                    <queue-statistics>
                      <queue>0</queue>
                      <bytes>0</bytes>
                      <dropped>0</dropped>
                      <dscp-values>
                        <dscp>16</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>17</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>18</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>19</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>20</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>21</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>22</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>23</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>24</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>25</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>26</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>27</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>28</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>29</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>30</dscp>
                      </dscp-values>
                      <dscp-values>
                        <dscp>31</dscp>
                      </dscp-values>
                      <packets>0</packets>
                      <qlen>0</qlen>
```

```
        <random-drop>0</random-drop>
      </queue-statistics>
    </traffic-class-queues-list>
    <traffic-class-queues-list>
      <traffic-class>3</traffic-class>
      <queue-statistics>
        <queue>0</queue>
        <bytes>0</bytes>
        <dropped>0</dropped>
        <dscp-values>
          <dscp>0</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>1</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>2</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>3</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>4</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>5</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>6</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>7</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>8</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>9</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>10</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>11</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>12</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>13</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>14</dscp>
        </dscp-values>
        <dscp-values>
          <dscp>15</dscp>
        </dscp-values>
        <packets>0</packets>
        <qlen>0</qlen>
        <random-drop>0</random-drop>
      </queue-statistics>
    </traffic-class-queues-list>
    <traffic-class-rates>
      <traffic-class>0</traffic-class>
      <rate>1250000000</rate>
    </traffic-class-rates>
```

```
                <traffic-class-rates>
                  <traffic-class>1</traffic-class>
                  <rate>1250000000</rate>
                </traffic-class-rates>
                <traffic-class-rates>
                  <traffic-class>2</traffic-class>
                  <rate>1250000000</rate>
                </traffic-class-rates>
                <traffic-class-rates>
                  <traffic-class>3</traffic-class>
                  <rate>1250000000</rate>
                </traffic-class-rates>
                <weighted-round-robin-weights>
                  <queue>0</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>1</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>2</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>3</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>4</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>5</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>6</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>7</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>8</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>9</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>10</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>11</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>12</queue>
                  <weight>1</weight>
                </weighted-round-robin-weights>
                <weighted-round-robin-weights>
                  <queue>13</queue>
```

```
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>14</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                  <weighted-round-robin-weights>
                    <queue>15</queue>
                    <weight>1</weight>
                  </weighted-round-robin-weights>
                </pipe-list>
                <rules/>
                <subport-name>dp0s5</subport-name>
                <traffic-class-list>
                  <traffic-class>0</traffic-class>
                  <bytes>0</bytes>
                  <dropped>0</dropped>
                  <packets>0</packets>
                  <random-drop>0</random-drop>
                </traffic-class-list>
                <traffic-class-list>
                  <traffic-class>1</traffic-class>
                  <bytes>0</bytes>
                  <dropped>0</dropped>
                  <packets>0</packets>
                  <random-drop>0</random-drop>
                </traffic-class-list>
                <traffic-class-list>
                  <traffic-class>2</traffic-class>
                  <bytes>0</bytes>
                  <dropped>0</dropped>
                  <packets>0</packets>
                  <random-drop>0</random-drop>
                </traffic-class-list>
                <traffic-class-list>
                  <traffic-class>3</traffic-class>
                  <bytes>0</bytes>
                  <dropped>0</dropped>
                  <packets>0</packets>
                  <random-drop>0</random-drop>
                </traffic-class-list>
              </subport-list>
            </shaper>
          </if-list>
        </state>
      </qos>
    </policy>
  </data>
</rpc-reply>
]]>]]>
```

SNMP

SNMP overview

SNMP (Simple Network Management Protocol) is a mechanism for managing network and computer devices.

SNMP uses a manager/agent model for managing the devices. The agent resides in the device and provides the interface to the physical device being managed. The manager resides on the management system and provides the interface between the user and the SNMP agent. The interface between the SNMP manager and the SNMP agent uses a Management Information Base (MIB) and a small set of commands to exchange information.

The AT&T Vyatta vRouter supports SNMP over both IPv4 and IPv6 networks.

The following list describes the SNMP components.

- MIB objects—A MIB contains the set of variables and objects that are managed (for example, MTU on a network interface). The objects are organized into a tree structure in which each object is a leaf node. Each object has its unique Object Identifier (OID). Objects are of two types: *scalar* and *tabular*. A scalar object defines a single object instance. A tabular object defines multiple related object instances that are grouped in MIB tables. For example, the uptime on a device is a scalar object, but the routing table in a system is a tabular object.
- Traps—In addition to MIB objects, the SNMP agent on a device can formulate alarms and notifications into SNMP traps. The device asynchronously sends the traps to the SNMP managers that are configured as trap destinations or targets. This sending of traps keeps the network manager informed of the status and health of the device. Traps are unacknowledged by the remote application that receives the message. The AT&T Vyatta vRouter uses User Datagram Protocol (UDP) for traps. For SNMP requests, UDP port 161 is used. For SNMP traps, UDP port 162 is used. SNMPv2 and SNMPv3 support traps. Traps can be configured for each routing protocol.

  Note: Protocols BFD, BGP, and OSPF are supported for SNMP traps and these traps are disabled by default.

- Informs— Informs are acknowledged traps. After receiving an inform notification, a remote application sends back an acknowledge message indicating that it received the message. By default, the AT&T Vyatta vRouter uses UDP for inform notifications and sends inform notifications to trap targets.

  Note: SNMPv3 supports informs.

SNMP commands

SNMP commands can be used to read or change configuration or to perform actions on a device, such as resetting it. The set of commands used in SNMP are: `GET`, `GET-NEXT`, `GET-RESPONSE`, `SET`, and `TRAP`.

- `GET` and `GET-NEXT` are used by the SNMP manager to request information about an object. These commands are used to view configuration or status or to poll information, such as statistics.
- `SET` is used by the SNMP manager to change the value of a specific object. Setting a configuration object changes the configuration of the device. Setting an executable object performs an action, such as a file operation or a reset.
- `GET-RESPONSE` is used by the SNMP agent on the device to return the requested information by `GET` or `GET-NEXT` or the status of the `SET` operation.
- The `TRAP` command is used by the agent to asynchronously inform the manager about events important to the manager.

SNMP versions

Currently, SNMP has three versions:

- SNMPv1—This version is the first version of the protocol. It is described in RFC 1157.

- SNMPv2—This version is an evolution of the first version, and it adds a number of improvements to SNMPv1. It is described in RFCs 1902 through 1908.
- SNMPv3—This version improves the security model in SNMPv2 and adds support for proxies. It is described in RFCs 3413 through 3415.

The AT&T Vyatta vRouter supports SNMPv2 with community string (SNMPv2c) and SNMPv3 with SNMPv3 users.

SNMPv3

SNMPv3 adds security features to address the security shortcomings of SNMPv1 and SNMPv2. For information standards for SNMPv3 that are supported on the AT&T Vyatta vRouter, see Supported standards *(page 71)*.

The SNMPv3 architecture uses a modular approach to allow the protocol to be adapted in the future, if and when other types of features are added. The architecture supports the simultaneous use of different security, access control, and message processing models.

The SNMPv3 architecture provides the following security-related models:

- User-based Security Model (USM)—Used for message security. This model is defined in RFC 3414.
- Transport Security Model (TSM)—Used for message security. This model is defined in RFC 5591.
- View-based Access Control Model (VACM)—Used for access control. This model is defined in RFC 2275.

The AT&T Vyatta vRouter currently supports all three models.

The SNMPv3 architecture supports the following security features through USM and TSM:

- Data integrity—Ensures that packets have not been altered or destroyed in transit.
- Data-origin authentication—Verifies that the received packets come from a valid source.
- Data confidentiality— Encrypts packets to prevent data from being disclosed to unauthorized sources.
- Message timeliness and replay protection— Ensures a packet whose generation time is outside of a specified time window is not accepted.

USM

The User-based Security Model (USM) provides SNMP message-level security and is the default security model for SNMPv3. It also uses the traditional concept of a user (identified by a username) with which to associate security information. This model uses UDP to send the SNMP packets.

The following table lists the security protocols and modules used in the USM model to provide the SNMP message-level security.

Table 8: Security protocols and modules used in the USM model

Module	Function	Notes
Authentication	Provides for data integrity and data-origin authentication	The following authentication protocols are supported: • HMAC-MD5-96 • HMAC-SHA-96 The entire message is checked for integrity. For a message to be authenticated, it needs to pass the authentication check and the timeliness check.

Module	Function	Notes
Privacy	Provides data confidentiality	The following encryption protocols are supported to encrypt messages: • Advanced Encryption Standard (AES) • Data Encryption Standard (DES) Note: If privacy is used, then the message also requires authentication.
Timeliness	Provides message timeliness and replay protection	The timeliness values in an SNMP message are used to do timeliness checking. This checking is performed only if authentication is applied to the message.

To authenticate or encrypt, or authenticate and encrypt the messages between an SNMP manager and an SNMP agent, the SNMP pair must share secret keys—an authentication secret key for authentication and an encryption secret key for encryption. Before using SNMPv3, you must first configure the secret keys so that they are added to the databases of the SNMP managers and agents that are to share the keys.

TSM

The Transport Security Model (TSM) within the SNMPv3 architecture is designed for use with secure transport protocols, such as SNMP over Secure Shell (SSH), Transport Layer Security (TLS), or Datagram Transport Layer Security (DTLS) to send SNMPv3 packets through secure tunnels. The AT&T Vyatta vRouter supports TLS and DTLS in its SNMPv3 implementation.

Note: The current implementation of TSM does not support SNMP over SSH.

TLS and DTLS use X.509 certificates to authenticate both the client and server of the secure tunnel connections. A public key infrastructure (PKI) is required to generate these certificates. To employ TLS and DTLS, you are required to generate X.509 security keys and certificates and install them on both the SNMP manager and the SNMP agent. The generation and distribution of certificates and keys using PKI involves numerous complex security issues, which are outside the scope of this document. Consult your particular PKI deployment documentation for the necessary procedures to generate and distribute these certificates and keys.

VACM

The View-based Access Control Model (VACM) is used for access control. In this model, access control is determined based on V3 groups and community. A group defines the access policy or the read-and-write access privileges for a set of SNMPv3 users. A group also defines the type of MIB view provided to a set of users. A group defines the following:

- Which users are allowed to access which view (a MIB or MIB object within a MIB)
- What type of access privileges are allowed into a view

Note: The AT&T Vyatta vRouter supports the access privilege types of read-only (ro) and read-write (rw) for groups.

Choosing USM or TSM

With two security models available, how do you determine which model to use in your network environment?

The main advantage of using TSM is the ability to integrate SNMP management into the existing X.509 public key security infrastructure of an organization.

Consider implementing TSM if you already have an X.509 public key infrastructure, need to deploy an X.509 public key infrastructure, or do not have a system for managing USM private keys in SNMPv3.

Consider implementing USM if you do not need to deploy an X.509 public key infrastructure or you already have a system for managing USM private keys for use in SNMPv3.

Default object IDs

Two default object IDs set by the AT&T Vyatta vRouter are as follows:

- sysObjectID = 1.3.6.1.4.1.30803
- sysDescr = Vyatta VSE6.6R0S6

The sysDescr object ID is updated automatically with each new release. It can also can be changed by using the `service snmp description` *desc* command.

Supported standards

The AT&T Vyatta vRouter implementation of SNMPv1, SNMPv2, and SNMPv3 complies with the following standards:

- RFC 1525, *Definitions of Managed Objects for Source Routing Bridges*
- RFC 2742, *Definitions of Managed Objects for Extensible SNMP Agents*
- RFC 2786, *Diffie-Helman USM Key Management Information Base and Textual Convention*
- RFC 2856, *Textual Conventions for Additional High Capacity Data Types*
- RFC 2864, *The Inverted Stack Table Extension to the Interfaces Group MIB*
- RFC 3165, *Definitions of Managed Objects for the Delegation of Management Scripts*
- RFC 3231, *Definitions of Managed Objects for Scheduling Management Operations*
- RFC 3411, *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*
- RFC 3412, *Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)*
- RFC 3413, *Simple Network Management Protocol (SNMP) Applications*
- RFC 3414, *User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)*
- RFC 3415, *View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)*
- RFC 3417, *Transport Mappings for the Simple Network Management Protocol (SNMP)*
- RFC 3419, *Textual Conventions for Transport Addresses*
- RFC 3584, *Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework*
- RFC 3635, *Definitions of Managed Objects for the Ethernet-like Interface Types*
- RFC 3826, *The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model*
- RFC 4001, *Textual Conventions for Internet Network Addresses*
- RFC 4273, *Definitions of Managed Objects for BGP-4*
- RFC 5591, *Transport Security Model for the Simple Network Management Protocol (SNMP)*
- RFC 5953, *Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)*

Supported MIBs

MIBs are typically located in the `/usr/share/snmp/mibs` directory.

The following table lists the standard MIBs and traps supported by the AT&T Vyatta vRouter. RFCs can be found at http://tools.ietf.org.

Table 9: Supported standard MIBs

MIB Name	Document Title	OIDs	Notes
BFD#MIB	RFC 7331, *Bidirectional Forwarding Detection (BFD) Management Information Base*	1.3.6.1.2.1.222	Version 22 of the BFD protocol MIB is supported, except the Echo Packet and Drop Counters. The following traps are supported: • bfdSessDiag • bfdSessDown • bfdSessUp
BGP4-MIB	RFC 1657, *Definitions of Managed Objects for the Fourth Version of Border Gateway Protocol (BGP-4)*	1.3.6.1.2.1.15	The protocol MIB is supported plus the following traps: • bgpEstablished • bgpBackwardTransition
HOST-RESOURCES-MIB	RFC 2790, *Host Resources MIB*	1.3.6.1.2.1.25	
RMON-MIB	RFC 2819, *Remote Network Monitoring Management Information Base*	1.3.6.1.2.1.16	
IF-MIB	RFC 2863, *The Interfaces Group MIB*	1.3.6.1.2.1.31	The following traps are supported: • linkUp • linkDown **Note:** IF-MIB lists lord*N* routing domain loopback interfaces, which are internal interfaces that cannot be configured and should be ignored.
IGMP-MIB	RFC2933, *Internet Group Management Protocol MIB*	1.3.6.1.2.1.85	
EVENT-MIB	RFC 2981, *Event MIB*		

MIB Name	Document Title	OIDs	Notes
IP-MIB	RFC 2011, *SNMPv2 Management Information Base for the Internet Protocol using SMIv2*	1.3.6.1.2.1.48	Only packets in which the AT&T Vyatta vRouter is an endpoint are accounted. Forwarded traffic is not accounted.
NOTIFICATION-LOG-MIB	RFC 3014, *Notification Log MIB*	1.3.6.1.2.1.92	
IPv6-MLD-MIB	RFC 3019, *IP Version 6 Management Information Base for The Multicast Listener Discovery Protocol*	1.3.6.1.2.1.91	
IPM-ROUTE	RFC 2932, *IPv4 Multicast Routing MIB*	1.3.6.1.2.1.83	
IPV6-UDP-MIB	RFC 2454, *IP Version 6 Management Information Base for the User Datagram Protocol*	1.3.6.1.2.1.7	Only packets in which the AT&T Vyatta vRouter is an endpoint are accounted. Forwarded traffic is not accounted.
KEEPALIVED-MIB	Authored by Vincent Bernat. Extends the keepalived daemon to support the Net-SNMP agentx protocol. Provides additional information specific to the AT&T Vyatta vRouter implementation, such as state information, sync group state information, and so on.	1.3.6.1.4.1.9586.100.5	
NAT-MIB	RFC 4008, *Definitions of Managed Objects for Network Address Translators (NAT)*	1.3.6.1.2.1.123	
PIM-MIB	RFC 2934, *Protocol Independent Multicast MIB for IPv4*	1.3.6.1.3.61	
RFC1213-MIB	RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*	1.3.6.1.2.1	
RIPv2-MIB	RFC 1724, *RIP Version 2 MIB Extension*	1.3.6.1.2.1.23	

MIB Name	Document Title	OIDs	Notes
SNMPv2-MIB	RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*	1.3.6.1.6.3.1	The following traps are supported: • coldStart • warmStart
TCP-MIB	RFC 4022, *Management Information Base for the Transmission Control Protocol (TCP)*	1.3.6.1.2.1.49	Only packets in which the AT&T Vyatta vRouter is an endpoint are accounted. Forwarded traffic is not accounted.
UDP-MIB	RFC 4113, *Management Information Base for the User Datagram Protocol (UDP)*	1.3.6.1.2.1.50	Only packets in which the AT&T Vyatta vRouter is an endpoint are accounted. Forwarded traffic is not accounted.
IP-Forward-MIB	RFC 4292, *IP Forwarding Table MIB*	1.3.6.1.2.1.4.24	
OSPF-MIB	RFC 4750, *OSPF Version 2 Management Information Base*	1.3.6.1.2.1.14	The following OSPF traps are supported: • ospfVirtIfStateChange • ospfNbrStateChange • ospfVirtNbrStateChange • ospfIfConfigError • ospfVirtIfConfigError • ospfIfAuthFailure • ospfVirtIfAuthFailure • ospfIfRxBadPacket • ospfVirtIfRxBadPacket • ospfTxRetransmit • ospfVirtIfTxRetransmit • ospfOriginateLsa • ospfMaxAgeLsa • ospfLsdbOverflow • ospfLsdbApproachingOverflow • ospfIfStateChange • ospfNssaTranslatorStatusChange
LLDPD-MIB	no RFC		The MIB module defines objects for Linux implementation of IEEE 802.1ab Link Layer Discovery Protocol (LLDP).
UCD-DISKIO-MIB	no RFC	1.3.6.1.4.1.2021.13.15.1	A table of IO devices and how much data they have read and written.

SNMP configuration examples

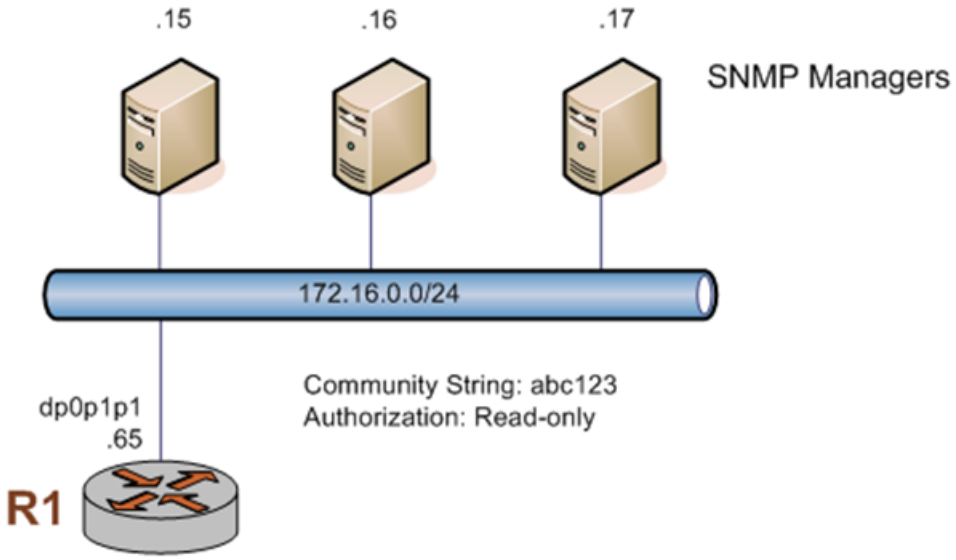
This section presents the following topics:

- Defining the SNMP community
- Assigning views to an SNMP community
- Specifying protocol-specific SNMP traps
- Specifying trap destinations
- SNMP over IPv6

To configure SNMP, the AT&T Vyatta vRouter MIB model must be loaded.

At the end of running these configuration procedures, you set up an SNMP community that includes three hosts, which serves as SNMP managers, and configures the system R1 to send traps to all the three managers. When you have finished, the system is configured as shown in the following figure.

Figure 4: Configuring SNMP communities and traps



Defining the SNMP community

SNMP community strings are used only by systems that support SNMPv1 and SNMPv2c protocols. SNMPv3 uses a username and password authentication, along with an encryption key.

The SNMP community of a system is the list of SNMP clients authorized to make requests of the system. Authorization for the community is in the form of a community string. The community string acts as a password, providing basic security and protecting the system against spurious SNMP requests.

- If no SNMP clients or networks are explicitly defined, then any client presenting the correct community string is granted the access privilege specified in the **`authorization`** option.
- If any client or network is defined, then only explicitly listed clients or networks are granted access to the system. Those clients have the access privilege specified by the **`authorization`** option. (The default is read-only.)

With reference to the figure **Configuring SNMP communities and traps**, the following configuration example shows how to set the SNMP community string for the system R1 to `abc123` and specify three clients for the community with the following IP addresses: `176.16.0.15`, `176.16.0.16`, and `176.16.0.17`. Read-only access is provided for this community.

Table 10: Defining an SNMP community

Step	Command
Create the **snmp** configuration node and the **community** configuration node. Set the community string. Note that using the **edit** command creates the community if it does not already exist. Navigate to the configuration node of the community for easier configuration.	`vyatta@R1# edit service snmp community abc123` `[edit service snmp community abc123]`

List the SNMP clients making up this community.	`vyatta@R1# set client 176.16.0.15`
	`vyatta@R1# set client 176.16.0.16`
	`vyatta@R1# set client 176.16.0.17`

Set the privilege level for this community to read-only.	`vyatta@R1# set authorization ro`
Commit the change.	`vyatta@R1# commit`

Verify the configuration.	`vyatta@R1# show`
	` authorization ro`
	` client 176.16.0.15`
	` client 176.16.0.16`
	` client 176.16.0.17`

Return to the top of the configuration tree.	`vyatta@R1# top`

Assigning views to an SNMP community

After you define an SNMP community and a view, you can associate each community with any number of views.

With reference to the figure **Configuring SNMP communities and traps**, the following example shows how to add views to a community.

Table 11: Assigning Views to an SNMP Community

Step	Command
Specify a sub#tree to appear in the view.	`vyatta@R1# set service snmp view myview oid` `1.3.6.1.2.1.4`

Associate the view with the community.	`vyatta@R1# set service snmp community abc123`
	` view`
	`myview`

Commit the changes.	`vyatta@R1# commit`

Step	Command

Display the configuration.	`vyatta@R1# show service snmp`
	`snmp {`
	`community abc123 {`
	`view myview`
	`}`
	`view myview {`
	`oid 1.3.6.1.2.1.4`
	`}`
	`}`

Specifying protocol-specific traps

The AT&T Vyatta vRouter supports specifying SNMP traps for each routing protocol in your configuration. Currently, this feature is supported for BGP, BFD, and OSPF protocols.

> **Note:** SNMP traps for all protocols are disabled by default on the AT&T Vyatta vRouter system. You must enable the SNMP traps by using the `set service snmp notification` command.

All notifications as defined in the supported MIBs for each routing protocol are enabled by using the keyword all in the command syntax.

In the following figure, the community string for R1 is defined as `abc123`. The traps from R1 are configured to be sent to the three SNMP Managers that are defined by the IP addresses of `176.16.0.15`, `176.16.0.16`, and `176.16.0.17`, respectively. R1 is connected to R2 and R3 by using a routing protocol, such as BFD. The following configuration example shows how to specify BFD-specific SNMP traps for R1.

> **Note:** The command to enable protocol-specific SNMP traps is similar for all routing protocols. Refer to the SNMP commands section for more information on the command syntax.

Figure 5: Specifying protocol-specific traps

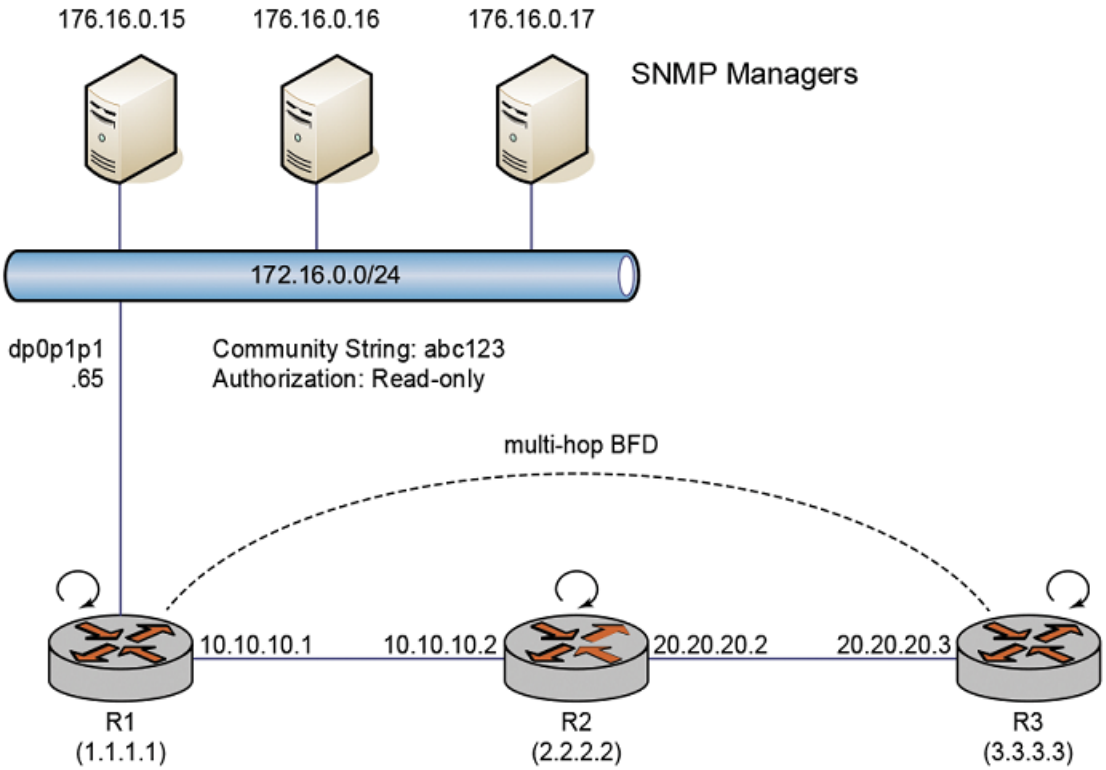


Table 12: Specifying protocol-specific traps

Step	Command
Enable all SNMP traps for BFD on R1.	`vyatta@R1# set service snmp notification bfd all`
Commit the change.	`vyatta@R1# commit`

View the change.	`vyatta@R1# show bfd`
	`BFD ID: 00 Start Time:Thu Jan 1 00:00:16 1970`
	`BFD Admin State: DOWN`
	`Number of Sessions: 0`
	`Slow Timer: 2000 Image type: MONOLITHIC`
	`Echo Mode: Disabled BFD Notifications enabled`
	`Next Session Discriminator: 1`
	
	`vyatta@R1# show service snmp`
	`service {`
	`  snmp {`
	`  community abc123 {`
	`  authorization ro`
	`  }`
	`  notification {`
	`  bfd {`
	`  all`
	`  }`
	`  }`
	`  }`
	`  ssh`
	`}`

Specifying trap destinations

After you specify an SNMP trap for a particular protocol, you must specify the SNMP trap destination as one or some of the configured SNMP managers.

With reference to the figure **Configuring SNMP communities and traps**, the following configuration example shows how to direct the system R1 to send SNMP traps to the configured network managers at `176.16.0.15`, `176.16.0.16`, and `176.16.0.17`.

Table 13: Specifying SNMP trap destinations

Step	Command

Define the trap destinations, one at a time.	`vyatta@R1# set service snmp trap-target 176.16.0.15`
	`vyatta@R1# set service snmp trap-target 176.16.0.16`
	`vyatta@R1# set service snmp trap-target 176.16.0.17`

Commit the change.	`vyatta@R1# commit`

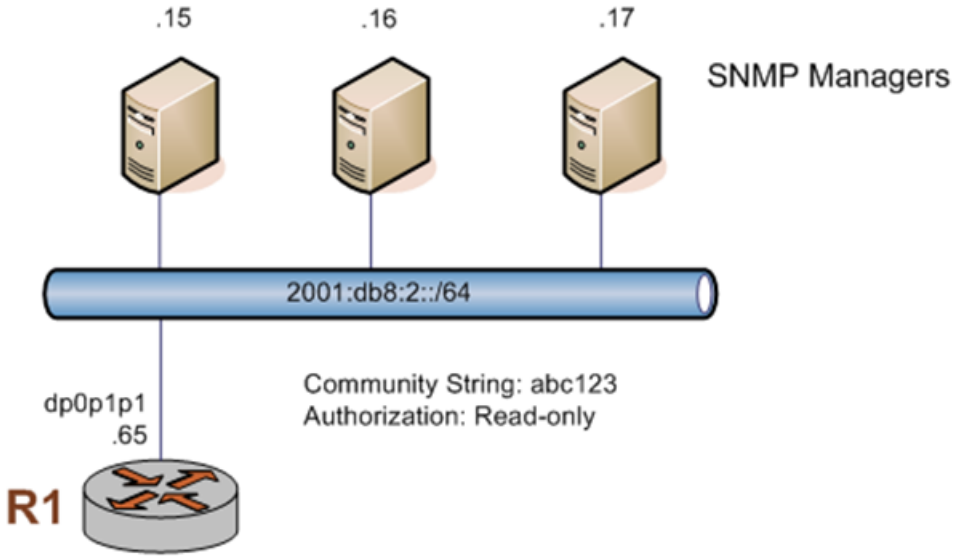
Step	Command

Verify the configuration.	`vyatta@R1# show service snmp trap-target`
	` trap-target 176.16.0.15 {`
	` }`
	` trap-target 176.16.0.16 {`
	` }`
	` trap-target 176.16.0.17 {`
	` }`

SNMP over IPv6

This sequence is the same as the previous example but uses IPv6 addresses. When you have finished, the system is configured as shown in the following figure.

Figure 6: Configuring SNMP communities and traps - IPv6



To define the SNMP configuration, perform the following steps in configuration mode.

Table 14: Defining the SNMP configuration

Step	Command
Create the **snmp** configuration node and the **community** configuration node. Set the community string.	`vyatta@R1# set service snmp community abc123`

List the SNMP clients making up this community.	`vyatta@R1# set service snmp community abc123`
	` client 2001:db8:2::15`
	`vyatta@R1# set service snmp community abc123`
	` client 2001:db8:2::16`
	`vyatta@R1# set service snmp community abc123`
	` client 2001:db8:2::17`

Set the privilege level for this community to read-only.	`vyatta@R1# set service snmp community abc123` ` authorization ro`

Step	Command

Define the trap destinations, one at a time.	`vyatta@R1# set service snmp trap-target 2001:db8:2::15`
	`vyatta@R1# set service snmp trap-target 2001:db8:2::16`
	`vyatta@R1# set service snmp trap-target 2001:db8:2::17`

Commit the change.	`vyatta@R1# commit`

Verify the configuration.	`vyatta@R1# show service snmp`
	`community abc123 {`
	`authorization ro`
	`client 176.16.0.15`
	`client 176.16.0.16`
	`client 176.16.0.17`
	`client 2001:db8:2::15`
	`client 2001:db8:2::16`
	`client 2001:db8:2::17`
	`}`
	`trap-target 176.16.0.15 {`
	`}`
	`trap-target 176.16.0.16 {`
	`}`
	`trap-target 176.16.0.17`
	`}`
	`trap-target 2001:db8:2::15 {`
	`}`
	`trap-target 2001:db8:2::16`
	`}`
	`trap-target 2001:db8:2::17 {`
	`}`

SNMPv3 configuration examples

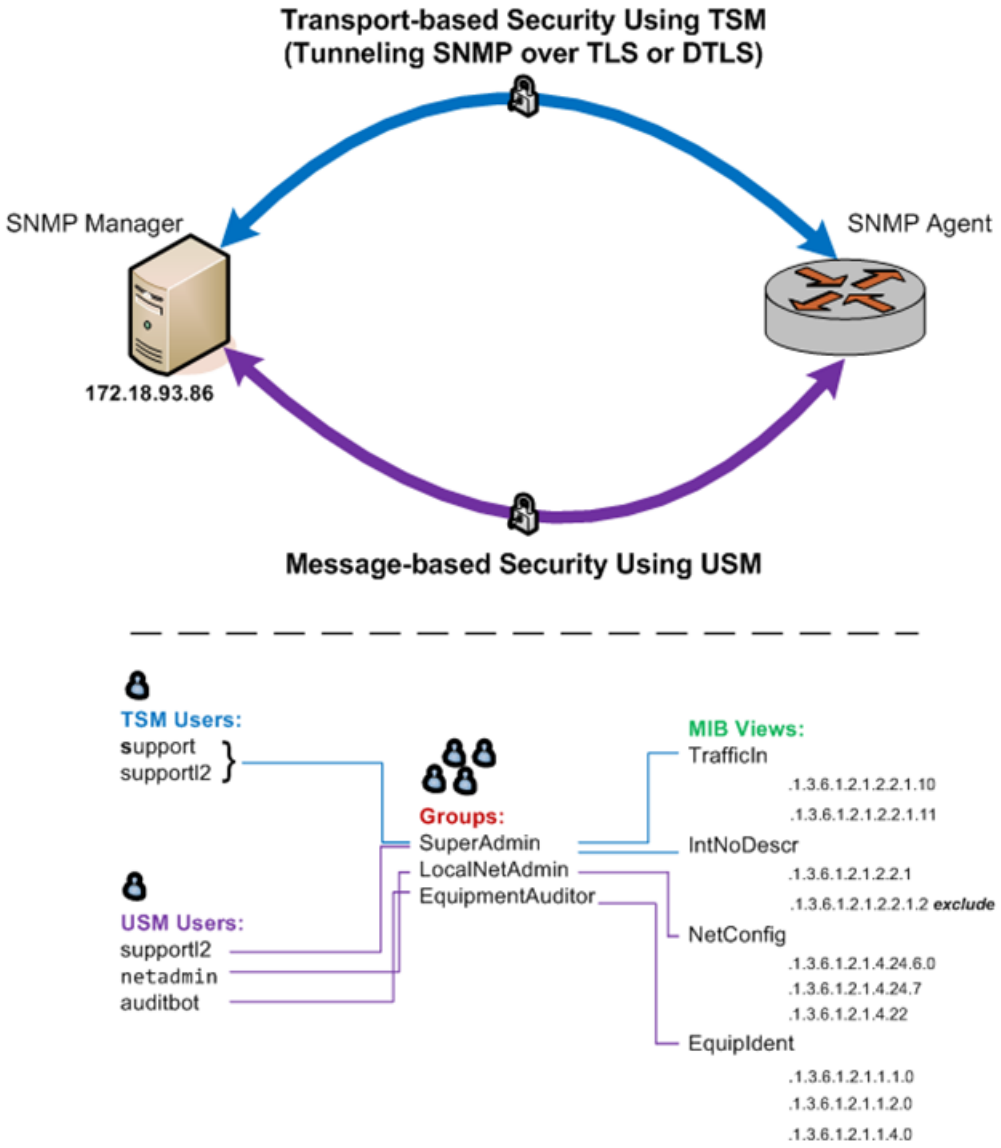
To configure SNMPv3, the AT&T Vyatta vRouter MIB model must be loaded.

The configurations in this section does the following:

- Defines the SNMPv3 users (USM and TSM users) and SNMPv3 groups
- Assigns the users and views to SNMPv3 groups
- Specifies the destinations to which to send the SNMP trap notifications to the configured SNMP manager at 172.18.93.86

Refer to the following figure for an example of SNMPv3 topology for the configurations in this section.

Figure 7: SNMPv3 topology example



Defining the users

This section provides the following topics:

- Defining the USM users
- Defining the TSM users

Defining the USM users

As part of the configuration steps to define the USM users, you are also required to specify the following information:

- Type of security protocol (authentication, privacy, or both) to apply to the SNMP messages sent between an SNMP manager and an SNMP client
- Secret keys associated with the selected security protocols

Before defining the USM users, configure the secret keys associated with the security protocols so that these are added to the databases of the SNMP entities that are to share the keys.

The following table shows the following configurations for USM:

- The auditbot user employs authentication only
- The netadmin and supportl2 users employ authentication and privacy

To define the USM users, perform the following steps in configuration mode. You must specify at least one of the security protocols (authentication or privacy).

Note: After defining a user by using the `service snmp v3 user` *username* **`auth plaintext-key`** *passwd* command and committing the command, a user engine ID (*engineid*) to be associated with the given SNMPv3 user is automatically added to the configuration. The engine ID is used during the generation of an encrypted password based on the configured plain-text password and the validation of passwords.

Note: During an upgrade to a new AT&T Vyatta vRouter image, ensure that you use the same user engine IDs for each of the existing SNMPv3 users.

Table 15: Defining the USM users

Step	Command

If you are using the authentication protocol to authenticate the user, specify the name of the user, authentication protocol, and authentication password. In this example, a clear-text password is used to authenticate a user.
 	
Note: The clear-text passwords are converted to encrypted keys after the commands are committed.
	`vyatta@R1# set service snmp v3 user auditbot auth plaintext-key auditbotkey`
	`vyatta@R1# set service snmp v3 user netadmin auth plaintext-key netadminkey`
	`vyatta@R1# set service snmp v3 user supportl2 auth plaintext-key supportl2key`

If you are using the privacy protocol to provide data confidentiality for SNMPv3 traffic, specify the name of the user, privacy protocol, and privacy password. In this example, a clear-text password is used to encrypt the SNMP traffic. **Note:** The clear-text passwords are converted to encrypted keys after the commands are committed.	`vyatta@R1# set service snmp v3 user netadmin privacy plaintext-key netadminkey1` `vyatta@R1# set service snmp v3 user supportl2 privacy plaintext-key supportl2key1`
Commit the change.	`vyatta@R1# commit`

Step	Command

Verify the configuration.
Note that the clear-text passwords configured for each of the users have been converted to encrypted keys and that engine IDs have been added to each user configuration where the user authentication protocol is used for authenticating the user.
	`vyatta@R1# show service snmp`
	`v3 {`
	`user auditbot {`
	`auth {`
	`encrypted-key`
	`0xba6273b420a64b415ad0a44e80106dbd`
	`}`
	`engineid`
	`0x80001f8880141fcc01ca3edd51`
	`}`
	`user netadmin {`
	`auth {`
	`encrypted-key`
	`0x110c1e3aa857084f9bf7ce4faaf44496`
	`}`
	`engineid`
	`0x80001f8880141fcc01ca3edd51`
	`privacy {`
	`encrypted-key`
	`0x4d8590f7fb640e35b673443823fccb71`
	`}`
	`}`
	`user support12 {`
	`auth {`
	`encrypted-key`
	`0x9a72fc4e7a3cf01c0eadecb13dcf6f7c`
	`}`
	`engineid`
	`0x80001f8880141fcc01ca3edd51`
	`privacy {`
	`encrypted-key`
	`0x792dedb243b0fcbb7662b802f1444671`
	`}`
	`}`
	`}`

Verify the configuration.	`vyatta@R1:~$ show snmp v3 user`
	
	`SNMPv3 Users:`
	
	`User Auth Priv Mode Group`
	`---- ---- ---- ----- -----`
	`auditbot md5 ro`
	`netadmin md5 des ro`
	`support12 md5 des ro`

Defining the TSM users

When defining a TSM user, you are also required to specify the TSM certificate of the user (either the certificate fingerprint or the file that holds the certificate). During the user configuration, also specify the TSM certificate of the SNMP agent (either the certificate fingerprint or the file that holds the certificate).

Before configuring a TSM user, create the X.509 user keys and certificates for the associated SNMP manager and agent, and then install each key-and-certificate pair on the paired SNMP entities.

> **Note:** The generation and distribution of certificates and keys by using PKI involves numerous complex security issues, which are outside the scope of this document. Consult your particular PKI deployment documentation for the necessary procedures to generate and distribute these certificates and keys.

Note: The location of the certificates and keys on the SNMP-manager system is dependent on the specific SNMP management software used.

Perform the following steps before configuring TSM:

1. Generate the X.509 user key and certificate (one pair) for each of the paired SNMP entities.
2. Add the security keys for the SNMP agent and SNMP manager to the `~/.snmp/tls/private/` directory.
3. Add the certificates for the SNMP agent and SNMP manager to the `~/.snmp/tls/certs/` directory.

TSM configuration example

The example shows the following configurations for TSM:.

- TSM user support and the file support.crt that holds the TSM certificate of this user
- TSM user supportl2 and the file supportl2.crt that holds the TSM certificate of this user

  Note: The TSM user supportl2 is also configured as a USM user. See Defining the USM users *(page 81)*

- File `snmpd.crt` that holds the TSM certificate of the SNMP agent

To define the TSM users and specify the TSM certificates for the TSM users and an SNMP agent, perform the following steps in configuration mode.

Table 16: Defining the TSM Users and Specifying TSM Certificates for TSM Users and an SNMP Agent

Step	Command
Specify the name of the TSM user and TSM certificate of the user (either the certificate fingerprint or the file that holds the certificate).	`vyatta@R1# set service snmp v3 user support tsm-key support.crt` `vyatta@R1# set service snmp v3 user supportl2 tsm-key supportl2.crt`
Specify the TSM certificate of the SNMP agent (either the certificate fingerprint or file that holds the certificate).	`vyatta@R1#set service snmp v3 tsm local-key snmpd.crt`
Commit the change.	`vyatta@R1# commit`

Verify the configuration.	`vyatta@R1:~$show snmp v3 user`
	
	`SNMPv3 Users:`
	
	`User Auth Priv Mode Group`
	`---- ---- ---- ---- -----`
	`auditbot md5 ro`
	`netadmin md5 des ro`
	`support ro`
	`supportl2 md5 des ro`

Step	Command
Verify the configuration.	vyatta@R1:~$ show snmp v3 certificates /etc/snmp/tls: certs/snmpd.crt: subject= /C=US/ST=CA/L=Davis/ O=Net-SNMP/OU=Development/CN=raji/ emailAddress=raji.mti@vyatta.com SHA1 Fingerprint=DB:C8:BC:07:40:0D:A6:68:EF:7D: 3E:CB:1B:22:52:E5:FA:FE:3D:D3 certs/support.crt: subject= /C=US/ST=CA/L=Davis/ O=Vyatta/OU=Development/CN=raji/ emailAddress=raji.mti@vyatta.com SHA1 Fingerprint=A9:E7:17:31:2A:84:96:DE:19:EE: 2D:36:D8:FD:B1:97:F9:A3:FF:1B certs/supportl2.crt: subject= /C=US/ST=CA/L=Davis/ O=Vyatta/OU=Development/CN=raji/ emailAddress=raji.mti@vyatta.com SHA1 Fingerprint=E9:4B:07:26:8E:65:C2:EC:25:37: 76:15:9C:12:DC:EF:FA:FA:81:04

Defining MIB views

To define a MIB view, specify the name of the view and the SNMP Object Identifier (OID) of the subtree to be included or excluded in the view. To identify a single row in a MIB table to be included within the view, specify a bit mask.

The following table shows the MIB views named EquipIdent, NetConfig, TrafficIn, and IntNoDescr and the OID subtrees to be included or excluded in these views.

To define MIB views, perform the following steps in configuration mode.

Table 17: Defining MIB views

Step	Command

Specify a name for a MIB view and define an OID subtree to be included in the view. Configure each MIB view one at a time.	`vyatta@R1# set service snmpview EquipIdent oid 1.3.6.1.2.1.1.1.0`
	`vyatta@R1# set service snmpview EquipIdent oid 1.3.6.1.2.1.1.2.0`
	`vyatta@R1# set service snmpview EquipIdent oid 1.3.6.1.2.1.1.4.0`
	`vyatta@R1# set service snmpview NetConfig oid 1.3.6.1.2.1.4.24.6.0`
	`vyatta@R1# set service snmpview NetConfig oid 1.3.6.1.2.1.4.24.7`
	`vyatta@R1# set service snmpview NetConfig oid 1.3.6.1.2.1.4.22`
	`vyatta@R1# set service snmpview TrafficIn oid 1.3.6.1.2.1.2.2.1.10`
	`vyatta@R1# set service snmpview TrafficIn oid 1.3.6.1.2.1.2.2.1.11`
	`vyatta@R1# set service snmpview IntNoDescr oid 1.3.6.1.2.1.2.2.1`

Specify a name for the MIB view and define an OID subtree to be excluded from the view.	`vyatta@R1# set service snmpview IntNoDescr oid 1.3.6.1.2.1.2.2.1.2 exclude`
Commit the change.	`vyatta@R1# commit`

Verify the configuration.	`vyatta@R1:~$ show snmpview`
	`SNMP Views:`
	`View : EquipIdent`
	`OIDs :`
	`.1.3.6.1.2.1.1.1.0`
	`.1.3.6.1.2.1.1.2.0`
	`.1.3.6.1.2.1.1.4.0`
	`View : IntNoDescr`
	`OIDs :`
	`.1.3.6.1.2.1.2.2.1`
	`.1.3.6.1.2.1.2.2.1.2 exclude`
	`View : NetConfig`
	`OIDs :`
	`.1.3.6.1.2.1.4.22`
	`.1.3.6.1.2.1.4.24.6.0`
	`.1.3.6.1.2.1.4.24.7`
	`View : TrafficIn`
	`OIDs :`
	`.1.3.6.1.2.1.2.2.1.10`
	`.1.3.6.1.2.1.2.2.1.11`

Defining user groups and assigning users and views to groups

To define an SNMPv3 user group, specify the name of the group. By default, the AT&T Vyatta vRouter supports the access privilege type of read-only (ro) for user groups. You do not need to set this parameter explicitly when defining a user group. After defining user groups, assign the configured users and views to them.

Note: Currently, the AT&T Vyatta vRouter only supports read-only privileges. It does not support read-write privileges.

The following table shows the following configurations:

- The user groups named EquipmentAuditor, LocalNetAdmin, and SuperAdmin
- Assignment of the users auditbot, netadmin, supportl2, and support to a user group
- Assignment of the views EquipIdent, NetConfig, TrafficIn, and IntNoDescr to a user group

To define user groups and assign users and views to the groups, perform the following steps in configuration mode.

Table 18: Defining user groups and assigning users and views to user groups

Step	Command

Define an SNMPv3 user group, one user at a time.	`vyatta@R1# set service snmp v3 group EquipmentAuditor`
	`vyatta@R1# set service snmp v3 group LocalNetAdmin`
	`vyatta@R1# set service snmp v3 group SuperAdmin`

Assign users to an SNMPv3 user group, one user at a time.	`vyatta@R1# set service snmp v3 user auditbot group EquipmentAuditor`
	`vyatta@R1# set service snmp v3 user netadmin group LocalNetAdmin`
	`vyatta@R1# set service snmp v3 user supportl2 group SuperAdmin`
	`vyatta@R1# set service snmp v3 user support group SuperAdmin`

Assign views to an SNMPv3 user group, one view at a time.	`vyatta@R1# set service snmp v3 group EquipmentAuditor view EquipIdent`
	`vyatta@R1# set service snmp v3 group LocalNetAdmin view NetConfig`
	`vyatta@R1# set service snmp v3 group SuperAdmin view TrafficIn`
	`vyatta@R1# set service snmp v3 group SuperAdmin view IntNoDescr`

Commit the change.	`vyatta@R1# commit`

Verify the configuration.	`vyatta@R1:~$ show snmp v3 group`
	
	`SNMPv3 Groups:`
	
	`Group View`
	`----- ----`
	`EquipmentAuditor EquipIdent(ro)`
	`LocalNetAdmin NetConfig(ro)`
	`SuperAdmin IntNoDescr(ro)`

Specifying trap destinations

To configure the destination of trap and inform notifications, specify the IP address (IPv4 or IPv6) of the SNMPv3 trap target and the name of the SNMPv3 user at the trap target. To authenticate the SNMPv3 user at the trap target, specify the `service snmp v3 trap-target` *addr* **`auth`** [**`encrypted-key`** | **`plaintext-key`**] *passwd* command. To encrypt the notifications at the trap target, specify the `service snmp v3 trap-target` *addr* **`privacy`** [**`encrypted-key`** | **`plaintext-key`**] *passwd* command.

By default, the AT&T Vyatta vRouter sends inform notifications to trap targets.

The following table shows an example of the AT&T Vyatta vRouter configured to send SNMP traps to the configured SNMP manager at 172.18.93.86.

To specify trap destinations, SNMPv3 user names at the trap target, and security keys, perform the following steps in configuration mode.

Table 19: Specifying trap destinations, SNMPv3 user names and security keys

Step	Command
Specify the IP address of the trap target and the clear-text password used for authenticating the user at the trap target. Note: The clear-text password is stored in the system in this form.	`vyatta@R1# set service snmp v3 trap-target 172.18.93.86 auth plaintext-key password7`
Optional. Specify the IP address of the trap target and the clear-text password used to encrypt the traps and informs. Note: The clear-text password is stored in the system in this form.	`vyatta@R1# set service snmp v3 trap-target 172.18.93.86 privacy plaintext-key password8`
Specify the type of notifications to send to the trap target.	`vyatta@R1#set service snmp v3 trap-target 72.18.93.86 type trap`
Specify the SNMP engine ID of the SNMPv3 trap target. **Note:** If the `service snmp v3 trap-target` *addr* **`type`** *type* command is set to trap, you must also the specify the engine ID of the SNMPv3 trap target using `service snmp v3 trap-target` *addr* **`engineid`** *engineid* command.	`vyatta@R1# set service snmp v3 trap-target 172.18.93.86 engineid 80001f8880634bd405730cdc50`
Specify the IP address of the trap target and the username at the trap target.	`vyatta@R1# set service snmp v3 trap-target 172.18.93.86 user Adminuser`
Commit the change.	`vyatta@R1# commit`

Verify the configuration.	`vyatta@R1:~$ show snmp v3 trap-target`
	`SNMPv3 Trap-targets:`
	`Trap-target Port Protocol`
	`Auth Priv Type EngineID`
	`User`
	`----------- ---- --------`
	`---- ---- ---- --------`
	`----`
	`172.18.93.86 162 udp`
	`md5 trap 80001f8880634bd405730...`
	`Administrator`

SNMP Commands

service snmp

Defines SNMP information for the AT&T Vyatta vRouter.

Syntax:
```
set service snmp
```

Syntax:
```
delete service snmp
```

Syntax:
```
show service snmp
```

Configuration mode

```
service {
 snmp {
 }
}
```

Use this command to define information about the SNMP communities to which this system should respond, location of and contact information for the system, and destinations for the SNMP traps.

Use the `set` form of this command to define SNMP settings.

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

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

service snmp community <community>

Defines an SNMP community.

Syntax:
`set service snmp community` *community* [**`authorization`** *auth* | **`client`** *addr* | **`network`** *net*]

Syntax:
`delete service snmp community` *community* [**`authorization`** | **`client`** | **`network`**]

Syntax:
`show service snmp community` *community* [**`authorization`** | **`client`** | **`network`**]

By default, no community is defined.

community

Multi-node. An SNMP community. The argument is the community string to be used to authorize SNMP managers making requests of this system. Letters, numbers, and hyphens are supported.

You can define more than one community by creating multiple **`community`** configuration nodes.

auth

Optional. The privileges for the community. The privileges are as follows:

`ro`

The community can view system information, but not change it. This is the default privilege.

`rw`

The community can read and write information.

The default privileges are ro.

addr

Optional. Multi-node. The IPv4 or IPv6 address of an SNMP client in the community that is authorized to access the system.

You can define more than one client by creating the **client** configuration node multiple times.

net

Optional. Multi-node. The IPv4 or IPv6 network of SNMP networks in the community that are authorized to access the server.

You can define more than one network by creating the **network** configuration node multiple times.

Configuration mode

```
service {
    snmp {
        community community {
            authorization auth
            client addr
            network net
        }
    }
}
```

Use this command to define an SNMP community.

If no SNMP clients or networks are explicitly defined, then any client presenting the correct community string is granted the access privilege specified by the authorization option. If a client or network is defined, then only explicitly listed clients or networks are granted access to the system.

Use the `set` form of this command to define an SNMP community.

Use the `delete` form of this command to remove SNMP community configuration or to restore the default value of an option.

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

service snmp community <community> view <viewname>

Associates a view with an SNMP community.

Syntax:

`set service snmp community` *community* **`view`** *viewname*

Syntax:

`delete service snmp community` *community* **`view`** *viewname*

Syntax:

`show service snmp community` *community* **`view`** *viewname*

Not applicable

community

The name of an SNMP community.

viewname

The name of a view to be associated with the SNMP community. Only alphanumeric characters for a view name are allowed.

Configuration mode.

```
service {
    snmp {
        community community
```

```
        view viewname
    }
}
```

Use this command to associate a view with an SNMP community.

The view must first be defined by using the `service snmp view` *viewname* command.

Use the `set` form of this command to associate a view with an SNMP community.

Use the `delete` form of this command to remove the association between the view and an SNMP community.

Use the `show` form of this command to display the name of the view associated with an SNMP community.

service snmp contact <contact>

Records contact information for the system.

Syntax:
`set service snmp contact` *contact*

Syntax:
`delete service snmp contact`

Syntax:
`show service snmp contact`

contact
Optional. Contact information for the system. This information is stored as MIB-2 system information. Letters, numbers, and hyphens are supported.

Configuration mode

```
service {
 snmp {
  contact contact
 }
}
```

Use this command to record contact information for the system.

Use the `set` form of this command to record contact information for the system.

Use the `delete` form of this command to remove contact information for the system.

Use the `show` form of this command to view contact information for the system.

service snmp description <desc>

Records a brief description of the system.

Syntax:
`set service snmp description` *desc*

Syntax:
`delete service snmp description`

Syntax:
`show service snmp description`

desc
Optional. A brief description of the system. This description is stored as MIB-2 system information. Letters, numbers, and hyphens are supported.

When set, this description is stored as the object ID sysDescr. By default, sysDescr is set to Vyatta [*version-string*], where *version-string* is the version of AT&T Vyatta vRouter software.

Configuration mode

```
service {
 snmp {
  description desc
 }
}
```

Use this command to record a brief description of the system.

Use the `set` form of this command to record a brief description of the system.

Use the `delete` form of this command to remove the system description.

Use the `show` form of this command to view the system description.

service snmp listen-address <addr>

Specifies the IP address on which the SNMP agent listens for requests.

Syntax:
`set service snmp listen-address` *addr* [**`port`** *port*]

Syntax:
`delete service snmp listen-address` *addr* [**`port`**]

Syntax:
`show service snmp listen-address` *addr* [**`port`**]

The SNMP agent listens on all addresses on port 161.

addr

Optional. Multi-node. The IPv4 or IPv6 address on which the SNMP agent listens for requests.

You can specify multiple listening addresses for SNMP by creating multiple **`listen-address`** configuration nodes.

port

The UDP port used for listening. The default port is 161.

Configuration mode

```
service {
 snmp {
  listen-address addr {
   port port
  }
 }
}
```

Use this command to specify the IPv4 or IPv6 address and port on which the SNMP agent listens for requests.

Use the `set` form of this command to specify listen-address parameters.

Use the `delete` form of this command to remove listen-address parameters.

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

service snmp location <location>

Records the location of the system.

Syntax:
`set service snmp location` *location*

Syntax:
`delete service snmp location`

Syntax:
`show service snmp location`

location
Optional. The location of the system. This location is stored as MIB-2 system information. Letters, numbers, and hyphens are supported.

Configuration mode

```
service {
 snmp {
  location location
 }
}
```

Use this command to record the location of the system.

Use the `set` form of this command to record the location of the system.

Use the `delete` form of this command to remove the system location.

Use the `show` form of this command to view the system location.

service snmp notification

Enables all protocol-specific SNMP traps for the BFD, BGP, or OSPF protocol.

Syntax:
`set service snmp notification [ bfd | bgp | ospf ] all`

Syntax:
`delete service snmp notification [ bfd | bgp | ospf ] all`

Syntax:
`show service snmp`

All protocol-specific SNMP traps are disabled by default on the system.

bfd
Specifies the BFD protocol.

bgp
Specifies the BGP protocol.

ospf
Specifies the OSPF protocol.

Configuration mode.

```
service {
       snmp {
           notification {
             { [ bfd | bgp | ospf ]
                                all
                        }

                }
        }
```

```
}
```

Use this command to enable protocol-specific SNMP traps.

Use the `set` form of this command to enable all protocol-specific SNMP traps for the BFD, BGP, or OSPF protocol.

Use the `delete` form of this command to delete all protocol-specific SNMP traps for the BFD, BGP, or OSPF protocol.

Use the `show` form of this command to display all protocol-specific SNMP traps for the BFD, BGP, or OSPF protocol.

service snmp trap-source <addr>

Specifies the IP address of the source of SNMP traps.

Syntax:
`set service snmp trap-source` *addr*

Syntax:
`delete service snmp trap-source` *addr*

Syntax:
`show service snmp trap-source`

By default, the system automatically selects the primary IP address of the interface facing the trap target.

addr

The IPv4 or IPv6 address of the source of SNMP traps.

This address is included as the source of SNMP traps in SNMP messages sent to an SNMP server. The address must be an address configured on a system interface.

Configuration mode

```
service {
 snmp {
  trap-source addr
 }
}
```

Use this command to specify the IPv4 or IPv6 address of the source of SNMP traps.

Use the `set` form of this command to specify the IP address of the source of SNMP traps.

Use the `delete` form of this command to remove a trap-source address and have the system select the source address automatically.

Use the `show` form of this command to view the trap-source addresses.

service snmp trap-target <addr>

Specifies the IP address and port of the destination for SNMP traps.

Syntax:
`set service snmp trap-target` *addr* [**`community`** *community* | **`port`** *port*]

Syntax:
`delete service snmp trap-target` *addr* [**`community`** *community* | **`port`**]

Syntax:
`show service snmp trap-target` *addr* [**`community`** *community* | **`port`**]

addr

Optional. Multi-node. The IPv4 or IPv6 address of the destination for SNMP traps.

You can specify multiple destinations for SNMP traps by creating multiple **trap-target** configuration nodes. Or, you can enter a space-separated list of IP addresses.

community

The community used when sending trap information. The default community is **public**.

port

The destination UDP port used for trap notification. The default port is 162.

Configuration mode

```
service {
 snmp {
  trap-target addr {
   community community
   port port
  }
 }
}
```

Use this command to specify the IPv4 or IPv6 address and port of the destination for SNMP traps as well as the community used when sending trap information.

Use the `set` form of this command to specify trap-target parameters.

Use the `delete` form of this command to remove trap-target parameters.

Use the `show` form of this command to view the trap-target configuration.

service snmp view <viewname> oid <oid>

Specifies a subtree to appear in the view.

Syntax:

`set service snmp view` *viewname* **oid** *oid* [**mask** | **exclude**]

Syntax:

`delete service snmp view` *viewname* **oid** *oid* [**mask** | **exclude**]

Syntax:

`show service snmp view` *viewname* **oid** *oid*

Not applicable

viewname

A view.

oid

Multi-node. The Object Identifier (OID) of a subtree to be included in or excluded from the view.

mask

A bit-mask that identifies a single row in a MIB table to be included or excluded. The bitmask is specified as hexadecimal digits delimited with a period (.). For example, ff.a0.

exclude

Exclude the identified subtree.

Configuration mode

```
service {
    snmp {
        view viewname{
            oid oid {
                mask mask
```

```
                exclude
            }
        }
    }
}
```

Use this command to specify a subtree to appear in the view.

Use the `set` form of this command to specify a subtree to appear in the view.

Use the `delete` form of this command to remove a specified subtree from the view.

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

service snmp v3 engineid <engineid>

Specifies the SNMP engine identifier (ID) of an SNMPv3 agent.

Syntax:
`set service snmp v3 engineid` *engineid*

Syntax:
`delete service snmp v3 engineid`

Syntax:
`show service snmp v3 engineid`

engineid
The engine ID of an SNMP agent. The engine ID consists of 2 to 32 hexadecimal digits.

Configuration mode

```
service {
 snmp {
  v3 {
   engineid engineid
  }
 }
}
```

Use this command to specify the SNMP engine ID of an SNMPv3 agent. This ID is a unique hexadecimal string that is used to identify the SNMP agent for administration purposes. The engine ID is used with a hashing function to generate keys for authentication and encryption of SNMPv3 messages.

Caution:

If you have SNMPv3 USM users associated with an SNMP engine ID, do not change or delete the value of the SNMP engine ID. The plaintext password that you enter for an SNMPv3 USM user is automatically encrypted using the Message Digest (MD5) encryption. The encrypted password is stored internally for use while the plaintext password is not saved or stored. The encrypted key is based on both the plaintext password and the engine ID. If the engine ID is changed or deleted, the stored encrypted keys for the SNMPv3 users become invalid. You will then be required to add these users to the SNMPv3 configuration once more to have these SNMPv3 users become valid in the AT&T Vyatta vRouter again.

Use the `set` form of this command to specify the SNMP engine ID of an SNMPv3 agent.

Use the `delete` form of this command to remove the SNMP engine ID of an SNMPv3 agent.

Use the `show` form of this command to view the configuration of the SNMP engine ID of an SNMPv3 agent.

service snmp v3 group <groupname>

Specifies the name of an SNMPv3 user group.

Syntax:
`set service snmp v3 group` *groupname*

Syntax:
`delete service snmp v3 group`

Syntax:
`show service snmp v3 group`

groupname
The name of an SNMPv3 user group. Only alphanumeric characters are supported.

Configuration mode

```
service {
    snmp {
        v3 {
            group groupname
        }
    }
}
```

Use this command to specify the name of an SNMPv3 user group. Use the `service snmp v3 user` *username* **`engineid`** *engineid* command to assign a user to a user group.

Use the `set` form of this command to specify the name of an SNMPv3 user group.

Use the `delete` form of this command to remove the name of an SNMPv3 user group.

Use the `show` form of this command to view the name of an SNMPv3 user group.

service snmp v3 group <groupname> mode <mode>

Defines the read/write access for an SNMPv3 user group.

Syntax:
`set service snmp v3 group` *groupname* **`mode`** *mode*

Syntax:
`delete service snmp v3 group` *groupname* **`mode`**

Syntax:
`show service snmp v3 group` *groupname* **`mode`**

The default mode is **`ro`**.

groupname
The name of an SNMPv3 user group.

mode
The mode for user group access rights. The mode is as follows:

`ro`
This mode allows users in the user group to view system information, but not change it.

`rw`
This mode provides users in the user group with read-write privileges.

The default mode is **`ro`**.

Configuration mode

```
service {
 snmp {
  v3 {
```

```
   group groupname {
    mode mode
   }
  }
 }
}
```

Use this command to specify the read/write access for an SNMPv3 user group.

Use the `set` form of this command to specify the read/write access for an SNMPv3 user group.

Use the `delete` form of this command to remove the read/write access for an SNMPv3 user group.

Use the `show` form of this command to view the read/write access for an SNMPv3 user group.

service snmp v3 group <groupname> seclevel <seclevel>

Defines the security level to apply to the users within an SNMPv3 user group.

Syntax:
`set service snmp v3 group` *groupname* **`seclevel`** *seclevel*

Syntax:
`delete service snmp v3 group` *groupname* **`seclevel`** *seclevel*

Syntax:
`show service snmp v3 group` *groupname* **`seclevel`** *seclevel*

groupname

The name of an SNMPv3 user group.

seclevel

The security level for user group. The security level is as follows:

`auth`

This security level requires users in the user group to use authentication as the security protocol to apply to the SNMP messages sent between an SNMP agent and SNMP manager

`priv`

This security level requires users in the user group to use encryption as the security protocol to apply to the SNMP messages sent between an SNMP agent and SNMP manager.

Configuration mode

```
service {
 snmp {
  v3 {
   group groupname {
    seclevel seclevel
   }
  }
 }
}
```

Use this command to specify the security level to apply to an SNMPv3 user group.

Use the `set` form of this command to specify the security level to apply to an SNMPv3 user group.

Use the `delete` form of this command to remove the security level specified for an SNMPv3 user group.

Use the `show` form of this command to view the security level for an SNMPv3 user group.

service snmp v3 group <groupname> view <viewname>

Associates a view with an SNMPv3 user group.

Syntax:
`set service snmp v3 group` *groupname* **`view`** *viewname*

Syntax:
`delete service snmp v3 group` *groupname* **`view`**

Syntax:
`show service snmp v3 group` *groupname* **`view`**

groupname

The name of an SNMPv3 user group.

viewname

The name of a view to be associated with the SNMPv3 user group. Only alphanumeric characters for a view name are allowed.

Configuration mode

```
service {
 snmp {
  v3 {
   group groupname {
    view viewname
   }
  }
 }
}
```

Use this command to associate a view with an SNMPv3 user group. The view must first be defined by using the `service snmp v3 view` *viewname* command.

Use the `set` form of this command to associate a view with an SNMPv3 user group.

Use the `delete` form of this command to remove the association between the view and an SNMPv3 user group.

Use the `show` form of this command to display the name of the view associated with an SNMPv3 user group.

service snmp v3 trap-target <addr>

Defines the SNMP target for informs or traps.

Syntax:
`set service snmp v3 trap-target` *addr*

Syntax:
`delete service snmp v3 trap-target` *addr*

Syntax:
`show service snmp v3 trap-target`

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr
  }
```

```
 }
}
```

Use this command to define the SNMP target for informs or traps.

Use the `set` form of this command to define the SNMP target for informs or traps.

Use the `delete` form of this command to remove the SNMP target for informs or traps.

Use the `show` form of this command to view the SNMP target for informs or traps.

service snmp v3 trap-target <addr> auth encrypted-key <passwd>

Defines the encrypted password to use for authentication at the trap target.

Syntax:

```
set service snmp v3 trap-target addr auth encrypted-key passwd
```

Syntax:

```
delete service snmp v3 trap-target addr auth encrypted-key
```

Syntax:

```
show service snmp v3 trap-target addr auth encrypted-key
```

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

passwd

The authentication password. Only hexadecimal passwords are allowed.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr {
    auth {
        encrypted-key passwd
    }
   }
  }
 }
}
```

Use this command to define the encrypted password to use for authentication at the trap target. Use the `service snmp v3 trap-target` *addr* **`auth plaintext-key`** *passwd* command to specify an unencrypted password for authentication. Only one of these two commands can be used to configure authentication for a given trap target.

Use the `set` form of this command to define the encrypted password for authentication.

Use the `delete` form of this command to remove the encrypted password for authentication.

Use the `show` form of this command to view the encrypted password for authentication.

service snmp v3 trap-target <addr> auth plaintext-key <passwd>

Defines the clear text password used for authentication at the trap target.

Syntax:

```
set service snmp v3 trap-target addr auth plaintext-key passwd
```

Syntax:

```
delete service snmp v3 trap-target addr auth plaintext-key
```

Syntax:

```
show service snmp v3 trap-target addr auth plaintext-key
```

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

passwd

The authentication password. The password must be eight or more characters. Only alphanumeric characters for a password are allowed.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr {
    auth {
        plaintext-key passwd
    }
   }
  }
 }
}
```

Use this command to define the clear text password used for authentication at the trap target. Use the `service snmp v3 trap-target` *addr* **`auth encrypted-key`** *passwd* command to specify an encrypted password for authentication. Only one of these two commands can be used to configure authentication for a given trap target.

Use the `set` form of this command to define the clear text password used for authentication.

Use the `delete` form of this command to remove the clear text password for authentication.

Use the `show` form of this command to view the clear text password for authentication.

service snmp v3 trap-target <addr> auth type <type>

Defines the protocol used for authentication at the trap target.

Syntax:

```
set service snmp v3 trap-target addr auth type type
```

Syntax:

```
delete service snmp v3 trap-target addr auth type
```

Syntax:

```
show service snmp v3 trap-target addr auth type
```

The default protocol is md5.

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

type

The protocol used for authentication. The protocol is as follows:

md5

Message Digest 5 (MD5) authentication

sha

Secure Hash Algorithm authentication.

The default protocol is md5.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr {
    auth {
        type type
    }
   }
  }
 }
}
```

Use this command to define the protocol used for authentication at the trap target.

Use the `set` form of this command to define the protocol used for authentication.

Use the `delete` form of this command to remove the protocol used for authentication.

Use the `show` form of this command to view the protocol used for authentication.

service snmp v3 trap-target <addr> engineid <engineid>

Specifies the SNMP engine identifier (ID) of the SNMPv3 trap target.

Syntax:

`set service snmp v3 trap-target` *addr* **`engineid`** *engineid*

Syntax:

`delete service snmp v3 trap-target` *addr* **`engineid`**

Syntax:

`show service snmp v3 trap-target` *addr* **`engineid`**

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

`engineid`

The engine ID of the SNMPv3 trap target. The *engineid* consists of 2 to 32 hexadecimal digits.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr {
    engineid engineid
   }
  }
 }
}
```

Use this command to specify the SNMP engine ID of the SNMPv3 trap target. This ID is a unique hexadecimal string that is used to identify the SNMP trap target for administration purposes. The engine ID is used with a hashing function to generate keys for authentication and encryption of SNMPv3 messages.

Note: If the `service snmp trap-target addr` *addr* command has been set to trap, you must also specify the engine ID of the SNMPv3 trap target using this command.

Use the `set` form of this command to specify the engine ID of the SNMPv3 trap target.

Use the `delete` form of this command to remove the engine ID of the SNMPv3 trap target.

Use the `show` form of this command to view the SNMP engine ID configuration.

service snmp v3 trap-target <addr> port <port>

Specifies the port on a trap target that SNMP traps and to which informs are sent.

Syntax:

```
set service snmp v3 trap-target addr port port
```

Syntax:

```
delete service snmp v3 trap-target addr port
```

Syntax:

```
show service snmp v3 trap-target addr port
```

The trap target uses port 162.

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

port

The port to which SNMPv3 traps and informs are sent. The range of values is 1 to 65535. The default value is 162.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr {
    port port
   }
  }
 }
}
```

Use this command to specify the port on a trap target that SNMP traps to which informs are sent.

Use the `set` form of this command to specify the port on a trap target that SNMP traps to which informs are sent.

Use the `delete` form of this command to remove the ports specified and return the system to using the default port.

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

service snmp v3 trap-target <addr> privacy encrypted-key <priv-key>

Defines the encrypted key for the privacy protocol used for traps and informs sent to the trap target.

Syntax:

```
set service snmp v3 trap-target addr privacy encrypted-key priv-key
```

Syntax:

```
delete service snmp v3 trap-target addr privacy encrypted-key
```

Syntax:

```
show service snmp v3 trap-target addr privacy encrypted-key
```

addr
The IPv4 or IPv6 address of the SNMPv3 trap target.

priv-key
The privacy key used to encrypt traps and informs sent to the trap target.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr {
    privacy {
        encrypted-key priv-key
    }
   }
  }
 }
}
```

Use this command to define the encrypted key for the privacy protocol used for traps and informs sent to the trap target.

Use the `set` form of this command to define the encrypted key for the privacy protocol used for traps and informs sent to the trap target.

Use the `delete` form of this command to remove the encrypted key for the privacy protocol used for traps and informs sent to the trap target.

Use the `show` form of this command to view the encrypted key for the privacy protocol used for traps and informs sent to the trap target.

service snmp v3 trap-target <addr> privacy plaintext-key <priv-key>

Defines the clear text key for the privacy protocol used for traps and informs sent to the trap target.

Syntax:
```
set service snmp v3 trap-target addr privacy plaintext-key priv-key
```

Syntax:
```
delete service snmp v3 trap-target addr privacy plaintext-key priv-key
```

Syntax:
```
show service snmp v3 trap-target addr privacy plaintext-key priv-key
```

addr
The IPv4 or IPv6 address of the SNMPv3 trap target.

priv-key
The privacy key used to encrypt traps and informs sent to the trap target. The key must be eight or more characters. Only alphanumeric characters for a privacy key are allowed.

Configuration mode

```
service {
 snmp {
  v3 {
   trap-target addr {
    privacy {
       plaintext-key priv-key
    }
```

```
    }
   }
  }
 }
```

Use this command to define the clear text key for the privacy protocol used for traps and informs sent to the trap target.

Use the `set` form of this command to define the clear text key for the privacy protocol used for traps and informs sent to the trap target.

Use the `delete` form of this command to remove the clear text key for the privacy protocol used for traps and informs sent to the trap target.

Use the `show` form of this command to view the clear text key for the privacy protocol used for traps and informs sent to the trap target.

service snmp v3 trap-target <addr> privacy type <type>

Defines the protocol used to encrypt traps and informs sent to the trap target.

Syntax:
`set service snmp v3 trap-target` *addr* **`privacy type`** *type*

Syntax:
`delete service snmp v3 trap-target` *addr* **`privacy type`**

Syntax:
`show service snmp v3 trap-target` *addr* **`privacy type`**

The default value is des.

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

type

The protocol used to encrypt traps and informs sent to the trap target. The protocol is as follows:

aes
Advanced Encryption Standard (AES) data encryption.

des
Data Encryption Standard (DES) data encryption

The default value is des.

Configuration mode

```
service {
    snmp {
        v3 {
            trap-target addr {
                privacy {
                    type type
                }
            }
        }
    }
}
```

Use this command to define the protocol used to encrypt traps and informs sent to the trap target.

Use the `set` form of this command to define the protocol used for privacy.

Use the `delete` form of this command to remove the protocol used for privacy.

Use the `show` form of this command to view the protocol used for privacy.

service snmp v3 trap-target <addr> protocol <protocol>

Defines the protocol for traps and informs sent to the trap target.

Syntax:
`set service snmp v3 trap-target` *addr* **`protocol`** *protocol*

Syntax:
`delete service snmp v3 trap-target` *addr* **`protocol`**

Syntax:
`show service snmp v3 trap-target` *addr* **`protocol`**

The system uses UDP.

addr
The IPv4 or IPv6 address of the SNMPv3 trap target.

protocol
The protocol used to send traps and informs to the trap target. The protocol is as follows:

`tcp`
Transmission Control Protocol (TCP).

`upd`
User Datagram Protocol (UDP).

The default protocol is udp.

Configuration mode

```
service {
    snmp {
        v3 {
            trap-target addr {
                protocol protocol
            }
        }
    }
}
```

Use this command to define the protocol for traps and informs sent to the trap target.

Use the `set` form of this command to define the protocol for traps and informs sent to the trap target.

Use the `delete` form of this command to remove the protocol for traps and informs sent to the trap target.

Use the `show` form of this command to view the protocol for traps and informs sent to the trap target.

service snmp v3 trap-target <addr> type <type>

Specifies the type of notifications to send to the trap target.

Syntax:
`set service snmp v3 trap-target` *addr* **`type`** *type*

Syntax:
`delete service snmp v3 trap-target` *addr* **`type`**

Syntax:
`show service snmp v3 trap-target` *addr* **`type`**

The system uses **`inform`**.

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

type

The notification type. The notification type is as follows:

inform
An SNMPv3 message sent to the trap target that requires acknowledgment.

trap
An SNMPv3 message sent to the trap target that does not require acknowledgment.

The default notification type is inform.

Configuration mode

```
service {
    snmp {
        v3 {
            trap-target addr {
                type type
            }
        }
    }
}
```

Use this command to specify the type of notifications to send to the trap target.

Note: If the notification type of trap is set using this command, then you must also specify the engine ID of the SNMPv3 trap target using the `service snmp v3 trap-target` *addr* **`engineid`** *engineid* command.

Use the `set` form of this command to specify the type of notifications sent to the trap target.

Use the `delete` form of this command to return the system to its default notification type.

Use the `show` form of this command to view the type of notifications sent to the trap target.

service snmp v3 trap-target <addr> user <username>

Defines an SNMPv3 username for authentication at the trap target.

Syntax:
`set service snmp v3 trap-target` *addr* **`user`** *username*

Syntax:
`delete service snmp v3 trap-target` *addr* **`user`**

Syntax:
`show service snmp v3 trap-target` *addr* **`user`**

addr

The IPv4 or IPv6 address of the SNMPv3 trap target.

username

The name of an SNMPv3 user at the trap target.

Configuration mode

```
service {
    snmp {
        v3 {
            trap-target addr {
                user username
            }
        }
    }
}
```

Use this command to define a username for authentication at the trap target.

Use the `set` form of this command to define a username for authentication at the trap target.

Use the `delete` form of this command to remove a username for authentication.

Use the `show` form of this command to view a username for authentication.

service snmp v3 tsm

Specifies that SNMPv3 uses the Transport Security Model (TSM).

Syntax:

```
set service snmp v3 tsm
```

Syntax:

```
delete service snmp v3 tsm
```

Syntax:

```
show service snmp v3 tsm
```

Configuration mode

```
service {
    snmp {
        v3 {
            tsm
        }
    }
}
```

Use this command to specify that SNMPv3 uses TSM for encryption.

Use the `set` form of this command to specify that SNMPv3 uses TSM encryption.

Use the `delete` form of this command to remove TSM encryption for SNMPv3.

Use the `show` form of this command to view that SNMPv3 uses TSM encryption.

service snmp v3 tsm local-key <local-key>

Specifies the fingerprint of a Transport Security Model (TSM) certificate for a server.

Syntax:

```
set service snmp v3 tsm local-key local-key
```

Syntax:

```
delete service snmp v3 tsm local-key
```

Syntax:

```
show service snmp v3 tsm local-key
```

local-key

The fingerprint of a TSM certificate or the filename of a key file.

Configuration mode

```
service {
    snmp {
        v3 {
            tsm {
                local-key local-key
            }
        }
```

```
    }
}
```

Use this command to specify the fingerprint of a TSM certificate for a server. The fingerprint can be specified either directly through a TSM certificate or indirectly through the name of the file containing the fingerprint.

Use the `set` form of this command to specify the fingerprint of a TSM certificate for a server.

Use the `delete` form of this command to remove the fingerprint of a TSM certificate for a server.

Use the `show` form of this command to view the fingerprint of a TSM certificate for a server.

service snmp v3 tsm port <port>

Defines the port used for TSM.

Syntax:

```
set service snmp v3 tsm port port
```

Syntax:

```
delete service snmp v3 tsm port
```

Syntax:

```
show service snmp v3 tsm port
```

The system uses port 10161.

port

The port used for TSM. The range of values is 1 to 65535. The default value is 10161.

Configuration mode

```
service {
    snmp {
        v3 {
            tsm {
                port port
            }
        }
    }
}
```

Use this command to define the port used for TSM.

Use the `set` form of this command to define the port used for TSM.

Use the `delete` form of this command to remove the port for TSM.

Use the `show` form of this command to view the port for TSM.

service snmp v3 user <username> auth encrypted-key <passwd>

Defines the encrypted password used to authenticate a user.

Syntax:

```
set service snmp v3 user username auth encrypted-key passwd
```

Syntax:

```
delete service snmp v3 user username auth encrypted-key
```

Syntax:

```
show service snmp v3 user username auth encrypted-key
```

username
The name of an SNMPv3 user.

passwd
The authentication password. Only hexadecimal passwords are allowed.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
                auth {
                    encrypted-key passwd
                }
            }
        }
    }
}
```

Use this command to define the encrypted password used to authenticate a user.

Use the `set` form of this command to define the encrypted password used to authenticate a user.

Use the `delete` form of this command to remove the encrypted password used to authenticate a user.

Use the `show` form of this command to view the encrypted password used to authenticate a user.

service snmp v3 user <username> auth plaintext-key <passwd>

Defines the clear text password used to authenticate a user.

Syntax:
`set service snmp v3 user` *username* **`auth plaintext-key`** *passwd*

username
The name of an SNMPv3 user.

passwd
The authentication password. The password must be eight or more characters. Only alphanumeric characters for a password are allowed.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
                auth {
                    plaintext-key passwd
                }
            }
        }
    }
}
```

Use this command to define the clear text password used to authenticate a user.

> **Note:** The plaintext password that you enter for an SNMPv3 user is automatically encrypted using the Message Digest (MD5) encryption. The encrypted password is stored internally for use while the plaintext password is not saved or stored.

Use the `set` form of this command to define the clear text password used to authenticate a user.

service snmp v3 user <username> auth type <type>

Defines the protocol used for user authentication.

Syntax:
`set service snmp v3 user` *username* **`auth type`** *type*

Syntax:
`delete service snmp v3 user` *username* **`auth type`**

Syntax:
`show service snmp v3 user` *username* **`auth type`**

The system uses MD5.

username

The name of an SNMPv3 user.

type

The protocol used for user authentication. The protocol is as follows:

`md5`
Message Digest 5 (MD5) authentication.

`sha`
Secure Hash Algorithm (SHA) authentication.

The default protocol is md5.

Configuration mode

Use this command to define the protocol used for user authentication.

Use the `set` form of this command to define the protocol used for user authentication.

Use the `delete` form of this command to remove the protocol used for user authentication.

Use the `show` form of this command to view the protocol used for user authentication.

```
service {
    snmp {
        v3 {
            user username {
                auth {
                    type type
                }
            }
        }
     }
}
```

service snmp v3 user <username> engineid <engineid>

Specifies the SNMP engine ID of an SNMPv3 user.

Syntax:
`set service snmp v3 user` *username* **`engineid`** *engineid*

Syntax:
`delete service snmp v3 user` *username* **`engineid`** *engineid*

Syntax:
`show service snmp v3 user` *username* **`engineid`** *engineid*

username

The name of an SNMPv3 user.

engineid

The engine ID of an SNMPv3 user. The engine ID consists of 2 to 32 hexadecimal digits.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
                engineid engineid
            }
        }
    }
}
```

Use this command to specify the SNMPv3 engine ID of an SNMPv3 user. This ID is a unique hexadecimal string that is used to identify the SNMPv3 user for administration purposes. The engine ID is used with a hashing function to generate keys for authentication and encryption of SNMP v3 messages.

Caution:

If you have SNMPv3 USM users associated with an SNMPv3 engine ID within your SNMPv3 configuration, do not change or delete the value of the SNMPv3 engine ID. The plaintext password that you enter for an SNMPv3 USM user is automatically encrypted using the Message Digest (MD5) encryption. The encrypted password is stored internally for use while the plaintext password is not saved or stored. The encrypted key is based on both the plaintext password and engine ID. If the engine ID is changed or deleted, the stored encrypted keys for the SNMPv3 USM users become invalid. You will then be required to add these users to the SNMPv3 configuration once more to have these SNMPv3 users become valid in the AT&T Vyatta vRouter again.

Use the `set` form of this command to specify the engine ID of an SNMPv3 user.

Use the `delete` form of this command to remove the engine ID of an SNMPv3 user.

Use the `show` form of this command to view the SNMPv3 engine ID configuration of SNMPv3 users.

service snmp v3 user <username> group <groupname>

Assigns an SNMPv3 user to a user group.

Syntax:

`set service snmp v3 user` *username* **`group`** *groupname*

Syntax:

`delete service snmp v3 user` *username* **`group`**

Syntax:

`show service snmp v3 user` *username* **`group`**

username

The name of an SNMPv3 user.

groupname

The name of a user group.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
```

```
                group groupname
            }
        }
    }
}
```

Use this command to assign an SNMPv3 user to a user group. The user group must first be created by using the `service snmp v3 group` *groupname* command.

Use the `set` form of this command to assign an SNMPv3 user to a user group.

Use the `delete` form of this command to remove an SNMPv3 user from a user group.

Use the `show` form of this command to view the user group to which an SNMPv3 user is assigned.

service snmp v3 user <username> mode <mode>

Specifies the mode for user access rights.

Syntax:
`set service snmp v3 user` *username* **`mode`** *mode*

Syntax:
`delete service snmp v3 user` *username* **`mode`**

Syntax:
`show service snmp v3 user` *username* **`mode`**

The default mode is **`ro`**.

username

The name of an SNMPv3 user.

mode

The mode for user access rights. The mode is as follows:

`ro`
This mode allows a user to view system information, but not change it.

`rw`
This mode provides a user with read-write privileges.

The default mode is ro.

Configuration mode

Use this command to specify the mode for user access rights.

Use the `set` form of this command to specify the mode for user access rights.

Use the `delete` form of this command to remove the mode for user access rights.

Use the `show` form of this command to view the mode for user access rights.

```
service {
    snmp {
        v3 {
            user username {
                mode mode
            }
        }
    }
}
```

service snmp v3 user <username> privacy encrypted-key <priv-key>

Defines the encrypted key used to encrypt user traffic.

Syntax:
`set service snmp v3 user` *username* **`privacy encrypted-key`** *priv-key*

Syntax:
`delete service snmp v3 user` *username* **`privacy encrypted-key`**

Syntax:
`show service snmp v3 user` *username* **`privacy encrypted-key`**

username
The name of an SNMPv3 user.

priv-key
The encrypted key. Only hexadecimal keys are supported.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
                privacy {
                    encrypted-key priv-key
                }
            }
        }
    }
}
```

Use this command to specify the encrypted key used to encrypt user traffic.

Use the `set` form of this command to specify the encrypted key used to encrypt user traffic.

Use the `delete` form of this command to remove the encrypted key.

Use the `show` form of this command to view the encrypted key used to encrypt user traffic.

service snmp v3 user <username> privacy plaintext-key <priv-key>

Defines the clear text key used to encrypt user traffic.

Syntax:
`set service snmp v3 user` *username* **`privacy plaintext-key`** *priv-key*

username
The name of an SNMPv3 user.

priv-key
The clear text key.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
```

```
                privacy {
                    plaintext-key priv-key
                }
            }
        }
    }
}
```

Use this command to define the clear text key used to encrypt user traffic.

Note: The plaintext password that you enter for an SNMPv3 user is automatically encrypted using the Message Digest (MD5) encryption. The encrypted password is stored internally for use while the plaintext password is not saved or stored.

Use the `set` form of this command to define the clear text key used to encrypt user traffic.

service snmp v3 user <username> privacy type <type>

Defines the protocol used to encrypt user traffic.

Syntax:

```
set service snmp v3 user username privacy type type
```

Syntax:

```
delete service snmp v3 user username privacy type
```

Syntax:

```
show service snmp v3 user username privacy type
```

The system uses **aes**.

username

The name of an SNMPv3 user.

type

The protocol used to encrypt user traffic. The protocol is as follows:

aes

Advanced Encryption Standard (AES) data encryption.

des

Data Encryption Standard (DES) data encryption.

The default protocol is aes.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
                privacy {
                    type type
                }
            }
        }
    }
}
```

Use this command to define the protocol used to encrypt user traffic.

Use the `set` form of this command to define the protocol used to encrypt user traffic.

Use the `delete` form of this command to remove the protocol used to encrypt user traffic.

Use the `show` form of this command to view the protocol used to encrypt user traffic.

service snmp v3 user <username> tsm-key <key>

Specifies the fingerprint of or the file containing the Transport Security Model (TSM) certificate.

Syntax:
`set service snmp v3 user` *username* **`tsm-key`** *key*

Syntax:
`delete service snmp v3 user` *username* **`tsm-key`**

Syntax:
`show service snmp v3 user` *username* **`tsm-key`**

username
The name of an SNMPv3 user.

key
The fingerprint of or the file containing the TSM certificate.

Configuration mode

```
service {
    snmp {
        v3 {
            user username {
                tsm-key key
            }
        }
    }
}
```

Use this command to specify the fingerprint of or the file containing the TSM certificate for a user.

Use the `set` form of this command to specify the fingerprint of or the file containing the TSM certificate.

Use the `delete` form of this command to remove the fingerprint or file name of the TSM certificate.

Use the `show` form of this command to view the fingerprint or file name of the TSM certificate.

service snmp v3 view <viewname>

Creates a view.

Syntax:
`set service snmp v3 view` *viewname*

Syntax:
`delete service snmp v3 view` *viewname*

Syntax:
`show service snmp v3 view`

viewname
The name of a view.

Configuration mode

```
service {
    snmp {
        v3 {
            view viewname
        }
    }
```

```
}
```

Use this command to create a view. A view specifies which objects a user can see.

Use the `set` form of this command to create a view.

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

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

service snmp v3 view <viewname> oid <oid>

Specifies a subtree to appear in the view.

Syntax:

```
set service snmp v3 view viewname oid oid [ mask mask | exclude ]
```

Syntax:

```
delete service snmp v3 view viewname oid oid [ mask | exclude ]
```

Syntax:

```
show service snmp v3 view view viewname oid oid
```

viewname
: The name of a view.

oid
: Multi-node. The Object Identifier (OID) of a subtree to be included in or excluded from the view.

mask
: A bit-mask that identifies a single row in a MIB table to be included or excluded. The bitmask is specified as hexadecimal digits delimited with a period (.). For example, ff.a0.

exclude
: Exclude the identified subtree.

Configuration mode

```
service {
    snmp {
        v3 {
            view viewname {
                oid oid {
                    mask mask
                    exclude
                }
            }
        }
    }
}
```

Use this command to specify a subtree to appear in the view.

Use the `set` form of this command to specify a subtree to appear in the view.

Use the `delete` form of this command to remove a specified subtree from the view.

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

show snmp

Displays SNMP statistics.

Syntax:

```
show snmp [ community | mib | v3 ]
```

Operational mode

community

Status of SNMP community.

mib

SNMP MIB information.

v3

Status of SNMP v3 on local host.

Use this command to display SNMP statistics.

The following example shows the output for `show snmp`.

```
vyatta@R1:~$ show snmp
[UDP: [127.0.0.1]:161->[0.0.0.0]]=>[Vyatta 999.larkspurse.06200031] Up: 0:02:40.80
Interfaces: 5, Recv/Trans packets: 545097/179020 | IP: 202587/89811
vyatta@R1:~$
```

show snmp community-mapping

Displays the SNMP version 1 and version 2 community and context mapping.

Syntax:

```
show snmp community-mapping
```

Operational mode

Use this command to display the SNMP version 1 and version 2 community and context mapping.

The following example shows the output for `show snmp community-mapping` .

```
vyatta@R1:~$ show snmp community-mapping
SNMPv1/v2c Community/Context Mapping:
Community                   Context
---------                   -------
commA                       'vrf1'
commB                       'vrf2'
deva                        'default'
deva2                       'vrf2'
test                        'vrf3'
test2                       'vrf4'
vyatta@R1:~$
```

show snmp trap-target

Displays the SNMP version 1 and version 2 trap targets.

Syntax:

```
show snmp trap-target
```

Operational mode

Use this command to display the SNMP version 1 and version 2 trap targets.

The following example shows the output for `show snmp trap-target`.

```
vyatta@R1:~$ show snmp trap-target
SNMPv1/v2c Trap-targets:
Trap-target                   Port   Routing-Instance Community
-----------                   ----   ---------------- ---------
1.1.1.1                              'vrf3'           'test'
vyatta@R1:~$
```

show snmp v3 certificates

Displays TSM certificates.

Syntax:

```
show snmp v3 certificates
```

Operational mode.

Use this command to display TSM certificates.

The following example shows the output for `show snmp v3 certificates`.

```
vyatta@R1:~$ show snmp v3 certificates
/etc/snmp/tls:

certs/snmpd.crt:
subject=
/C=US/ST=CA/L=Davis/O=Net-SNMP/OU=Development/CN=vyatta@debian/emailAddress=vyatta@debian
SHA1 Fingerprint=33:F2:92:24:E8:1A:D4:99:10:91:F5:A8:84:2A:2E:AD:96:C7:FE:C0

certs/usertsmro.crt:
subject=
/C=US/ST=CA/L=Davis/O=Net-SNMP/OU=Development/CN=vyatta@debian/emailAddress=vyatta@debian
SHA1 Fingerprint=15:6B:82:AB:FA:27:1F:E0:1C:1D:5B:F4:0E:2E:41:A0:C6:38:3E:11

certs/usertsmrw.crt:
subject=
/C=US/ST=CA/L=Davis/O=Net-SNMP/OU=Development/CN=vyatta@debian/emailAddress=vyatta@debian
SHA1 Fingerprint=CE:4A:F4:48:D7:44:B6:9E:F5:1D:05:F9:66:C7:C0:DE:9D:98:08:9E

vyatta@R1:~$
```

show snmp v3 group

Displays a list of configured groups.

Syntax:

```
show snmp v3 group
```

Operational mode

Use this command to display a list of configured groups.

The following example shows the output for `show snmp v3 group`.

```
vyatta@R1:~$ show snmp v3 group
SNMPv3 Groups:
Group               View
```

```
-----               ----
group1              view1(ro)
group2              view2(ro)
group3              view3(ro)
vyatta@R1:~$
```

show snmp v3 trap-target

Displays the list of SNMP version 3 trap targets.

Syntax:

```
show snmp v3 trap-target
```

Operational mode

Use this command to display the list of SNMP version 3 trap targets.

The following example shows the output for `show snmp v3 trap-target`.

```
vyatta@R1:~$ 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                        'vrf4'
         'test'
vyatta@R1:~$
```

show snmp v3 user

Displays a list of configured users.

Syntax:

```
show snmp v3 user
```

Operational mode

Use this command to display a list of configured users.

The following example shows the output for `show snmp v3 user`.

```
vyatta@R1:~$ show snmp v3 user
SNMPv3 Users:
User                Auth Priv Mode Group
----                ---- ---- ---- -----
user1               md5       ro   group1
user2               md5  aes  ro   group2
user3               sha       ro   group3
user4               md5       rw
vyatta@R1:~$
```

show snmp v3 view

Displays a list of configured views.

Syntax:
`show snmp v3 view`

Operational mode

Use this command to display a list of configured views.

The following example shows the output for `show snmp v3 view`.

```
vyatta@R1:~$ show snmp v3 view
SNMPv3 Views:
View : view1
OIDs :
       .1.1.1.1.1.1
       .1.1.1.1.1.2
       .1.1.1.1.1.3 mask ff.a0
View : view2
OIDs :
       .2.1.1.1.1
       .2.1.1.1.1.2 exclude
View : view3
OIDs :
       .3.1.1.1.1
       .3.1.1.1.1.1 exclude
vyatta@R1:~$
```

show snmp routing-instance

Displays the routing instance used to reach the host.

Syntax:
`show snmp routing-instance`

Operational mode

Use this command to display the routing instance used to reach the host.

The following example shows the output for `show snmp routing-instance`.

```
vyatta@R1:~$ show snmp routing-instance
Routing Instance SNMP Agent is Listening on for Incoming Requests:
Routing-Instance            RDID
-----------------           ----
vrf1                        5
vyatta@R1:~$
```

VRF Support

VRF support for SSH

You can configure SSH on any routing instance. If you configure SSH without specifying a routing instance, the default routing instance is used.

The following example shows how to configure SSH for the default routing instance.

```
vyatta@R1# set service ssh listen-address 10.0.0.1
vyatta@R1# set service ssh port 21
vyatta@R1# run show configuration
service {
    ssh {
        listen-address 10.0.0.1
        port 21
                        }
}
```

The following example shows the same configuration sequence for the BLUE routing instance.

```
vyatta@R1# set routing routing-instance BLUE service ssh listen-address 10.0.0.1
vyatta@R1# set routing routing-instance BLUE service ssh port 21
vyatta@R1# commit
vyatta@R1# run show configuration
routing {
        routing-instance BLUE {
                service {
                        ssh {
                           listen-address 10.0.0.1
                           port 21
                        }

                }
        }
}
```

For more information about SSH and configuring SSH, see AT&T Vyatta Network Operating System Basic System Configuration Guide.

VRF support for Telnet

You can configure Telnet on any routing instance. If you configure Telnet without specifying a routing instance, the default routing instance is used.

When you configure Telnet service in a routing instance, the external user can connect to the vRouter through a Telnet session by using the configuration parameters for that instance.

The Telnet service can be started with parameters that are specified in the configuration. If parameters are not specified, Telnet service starts on the default port (port 23).

The following example shows how to configure Telnet for the default routing instance.

```
vyatta@R1# set service telnet listen-address 42.42.42.42
vyatta@R1# set service telnet port 1234
vyatta@R1# commit
vyatta@R1# run show configuration
service {
        telnet {
                listen-address 42.42.42.42
                port 1234
        }
```

```
}
```

The following example shows the same configuration sequence for the BLUE routing instance.

```
vyatta@R1# set routing routing-instance BLUE service telnet listen-address 42.42.42.42
vyatta@R1# set routing routing-instance BLUE service telnet port 1234
vyatta@R1# commit
vyatta@R1# run show configuration
routing {
        routing-instance BLUE {
                service {
                        telnet {
                                listen-address 42.42.42.42
                                port 1234
                        }
                }
        }
}
```

VRF support for SNMP

The AT&T Vyatta vRouter supports the implementation of SNMP on a routing instance, which allows the following associations and configurations:

- An SNMP client to be associated with a specific routing instance and handle context-based access to MIBs.
- An SNMP trap target to be associated with a routing instance for sending SNMP notifications that are specific to the routing instance.
- An SNMP agent to be configured to listen for incoming requests from a specific routing instance.

The SNMP V2 clients are associated with a routing instance by mapping the SNMP community strings with a routing instance, as shown in the following command:

- `set service snmp community <comm-string> [context <routing-instance>]`

When a V2 request with a community string that is mapped to a routing instance is received, an SNMP agent retrieves MIB information that is specific to the routing instance.

The SNMP V3 clients are associated with a routing instance by specifying the routing instance as context in their requests. An SNMP agent returns context-based MIB information for these requests.

The SNMP V2 and V3 trap targets can be configured to receive routing instance-specific SNMP notifications. Traps to these targets are sent out on the configured routing instance, as shown in the following sample:

- `set service snmp trap-target <ip-addr> [routing-instance <name>]`
- `set service snmp v3 trap-target <ip-addr> [routing-instance <name>]`

When no routing instance is configured for a trap target, traps are sent over a default routing instance.

An SNMP agent can be configured to accept client requests from a specific routing instance:

- `set service snmp [routing-instance <name>]`

When no routing instance is configured, an SNMP agent listens for client requests on a default routing instance.

Configuring SNMP on a routing instance

The following sections provide examples of configuration mode commands.

Associating an SNMP client on a routing instance

The following configuration associates the commA community string with the RED routing instance and the commB community string with the BLUE routing instance. Only one context name can be mapped to a community, but multiple communities can be mapped to the same context name. A community string that is mapped to a context must have a defined view.

Table 20: Associating an SNMP client on a routing instance

Step	Command
Set the SNMP version 1 and version 2 community as commA and context as RED.	`vyatta@R1# set service snmp community commA context red`
Set the SNMP version 1 and version 2 community as commB and context as BLUE.	`vyatta@R1# set service snmp community commB context blue` `vyatta@R1# set service snmp view all oid 1`
Associate all views with the commA SNMP community.	`vyatta@R1# set service snmp community commA view all`
Associate all views with the commB SNMP community.	`vyatta@R1# set service snmp community commB view all`

View the configuration.	`vyatta@R1# show service snmp community`
	`community commA {`
	`  context red`
	`  view all`
	`}`
	`community commB {`
	`  context blue`
	`  view all`
	`}`
	`vyatta@vyatta#`

Associating a trap target with a routing instance

The following configuration associates an SNMPv2 trap target with the 1.1.1.1 IP address on the RED routing instance.

Table 21: Associating SNMPv2 trap targets on a routing instance

Step	Command
Set the SNMPv2 trap target with the 1.1.1.1 IP address on the RED routing instance.	`vyatta@R1# set service snmp v2 trap-target 1.1.1.1 routing-instance red`
Define the test community configuration node.	`vyatta@R1# set service snmp v2 trap-target 1.1.1.1 community test`
Commit the configuration.	`vyatta@R1# commit`

Step	Command

View the configuration.	`vyatta@R1# show service snmp v2 trap-target`
	` trap-target 1.1.1.1 {`
	`  community test`
	`  routing-instance red`
	` }`
	`vyatta@vyatta#`

The following configuration associates an SNMPv3 trap target with IP address 2.2.2.2 on the RED routing instance.

Table 22: Associating an SNMPv3 trap target on a routing instance

Step	Command
Set the SNMPv3 trap target with the 2.2.2.2 IP address on the RED routing instance.	`vyatta@R1# set service snmp v3 trap-target 2.2.2.2 routing-instance red`
Define the usr2 SNMPv3 user.	`vyatta@R1# set service snmp v3 trap-target 2.2.2.2 user usr2`
Define a cleartext password to authenticate a user.	`set service snmp v3 trap-target 2.2.2.2 auth plaintext-key "usr2usr2"`
Commit the configuration.	`vyatta@R1# commit`

View the configuration.	`vyatta@R1# show service snmp v3 trap-target`
	` trap-target 2.2.2.2 {`
	`  auth {`
	`  plaintext-key "********"`
	`  }`
	`  user usr2`
	`  routing-instance red`
	` }`
	`vyatta@vyatta#`

Configuring an SNMP agent to listen on a routing instance

The following configuration shows how to configure an SNMP agent to listen for incoming requests from the RED routing instance.

Table 23: Configuring an SNMP agent on a routing instance

Step	Command
Set an SNMP agent to listen for incoming requests from the RED routing instance.	`vyatta@R1# set service snmp routing-instance red`

Step	Command

View the configuration.	`vyatta@R1# show service snmp routing-instance`
	`  routing-instance red`
	`vyatta@vyatta#`

Supported VRF-aware SNMP MIBs

The following table lists the VRF aware SNMP MIBs and traps that are supported by the AT&T Vyatta vRouter.

Table 24: Supported VRF aware SNMP MIBs

MIB Name	Document Title	OIDs	Notes
IP-MIB	RFC 4113, *Management Information Base for the User Datagram Protocol (UDP)*	1.3.6.1.2.1.4	The following tables are VRF aware: • ipAddrTable (1.3.6.1.2.1.4.20) • inetCidrRoute (1.3.6.1.2.1.4.21); • ipIfStatsTable (1.3.6.1.2.1.4.31) • ipAddressPrefixTable (1.3.6.1.2.1.4.32) • ipAddressTable (1.3.6.1.2.1.4.34) • ipDefaultRouteTable (1.3.6.1.2.1.4.37)
IPV6-MIB	RFC 2465, *Management Information Base for IP Version 6*	1.3.6.1.2.1.55.1	The following tables are VRF aware: • ipv6IfTable (1.3.6.1.2.1.55.1.5)
IP-FORWARD-MIB	RFC 4292, *IP Forwarding Table MIB* RFC 2096, *IP Forwarding Table MIB*	1. .3.6.1.2.1.4.24	The following tables are VRF aware: • inetCidrRouteTable (1.3.6.1.2.1.4.24.7) • ipCidrRouteTable (1.3.6.1.2.1.4.24.4)

Note: For non-default VRF context, the above MIBs are read-only, and the variables of these MIBs cannot be set or modified.

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'
```

VRF-Aware SNMP Commands

show snmp community-mapping

Displays the SNMP version 1 and version 2 community and context mapping.

Syntax:

```
show snmp community-mapping
```

Operational mode

Use this command to display the SNMP version 1 and version 2 community and context mapping.

The following example shows the output for `show snmp community-mapping` .

```
vyatta@R1:~$ show snmp community-mapping
SNMPv1/v2c Community/Context Mapping:
Community                   Context
---------                   -------
commA                       'vrf1'
commB                       'vrf2'
deva                        'default'
deva2                       'vrf2'
test                        'vrf3'
test2                       'vrf4'
vyatta@R1:~$
```

show snmp trap-target

Displays the SNMP version 1 and version 2 trap targets.

Syntax:

```
show snmp trap-target
```

Operational mode

Use this command to display the SNMP version 1 and version 2 trap targets.

The following example shows the output for `show snmp trap-target`.

```
vyatta@R1:~$ show snmp trap-target
SNMPv1/v2c Trap-targets:
Trap-target                   Port   Routing-Instance Community
-----------                   ----   ---------------- ---------
1.1.1.1                              'vrf3'           'test'
vyatta@R1:~$
```

show snmp v3 trap-target

Displays the list of SNMP version 3 trap targets.

Syntax:

```
show snmp v3 trap-target
```

Operational mode

Use this command to display the list of SNMP version 3 trap targets.

The following example shows the output for `show snmp v3 trap-target`.

```
vyatta@R1:~$ 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                        'vrf4'
         'test'
vyatta@R1:~$
```

show snmp routing-instance

Displays the routing instance used to reach the host.

Syntax:

```
show snmp routing-instance
```

Operational mode

Use this command to display the routing instance used to reach the host.

The following example shows the output for `show snmp routing-instance`.

```
vyatta@R1:~$ show snmp routing-instance
Routing Instance SNMP Agent is Listening on for Incoming Requests:
Routing-Instance            RDID
-----------------           ----
vrf1                        5
vyatta@R1:~$
```

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
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

Acronym	Description
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
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
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

Acronym	Description
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
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