VYATTA, INC. | Vyatta OFR

Vyatta OFR Configuration Guide

Vyatta
Suite 160
One Waters Park Drive
San Mateo, CA 94403
vyatta.com

ISSUE DATE: February 2007

DOCUMENT REVISION NO. Rel VC2 v02.

DOCUMENT PART NO. A0-0075-00-02

Table of Contents

Quick List of Examples

Use this list to help you locate examples you'd like to try or look at.

Preface

This guide explains how to use the Vyatta OFR router, and how to use Vyatta OFR router commands in the command-line interface. It provides an overview of the router's functionality, highlighting core concepts, and a detailed description of each available command.

This preface provides information about using this guide. The following topics are covered:

- Intended Audience
- Organization of This Guide
- Document Conventions
- Vyatta Publications

Intended Audience

This guide is intended for experienced system and network administrators. Depending on the functionality to be used, readers should have specific knowledge in the following areas:

- Networking and data communications
- TCP/IP protocols
- General router configuration
- Routing protocols
- Network administration
- Network security

Organization of This Guide

This guide has the following aids to help you find the information you are looking for:

- **Quick List of Examples**

  Use this list to help you locate examples you'd like to try or look at.

- **Quick Guide to Configuration Statements**
- Use this section to quickly see the complete syntax of configuration statements.

This guide has the following chapters and appendixes:

Chapter	Description	Page
Chapter 1: System Management	This chapter explains how to configure basic system information for the router such as host name, DNS information, and date and time.	1
Chapter 2: Ethernet and VLAN Interfaces	This chapter explains how to configure Ethernet interfaces, the loopback interface, and VLAN interfaces.	16
Chapter 3: Serial Interfaces	This chapter describes how to configure serial interfaces on the Vyatta OFR.	27
Chapter 4: Basic Services	This chapter explains how to configure system services such as DHCP, HTTP, Telnet, and SSH.	33
Chapter 5: Forwarding and Routing	This chapter provides a brief overview of basic routing topics: traffic fowarding, unicast routing, multicast routing, and multicast topology discovery.	44

Chapter 6: Routing Policies	This chapter describes the policy framework, which you can use to apply policies that influence routing behavior.	55
Chapter 7: Static Routes	This chapter describes how to configure static routes on the Vyatta OFR.	71
Chapter 8: RIP	This chapter describes how to configure the Routing Information Protocol on the Vyatta OFR.	75
Chapter 9: OSPF	This chapter describes how to configure the OSPF routing protocol on the router.	75
Chapter 10: BGP	This chapter describes how to configure the Border Gateway Protocol on the Vyatta OFR.	93
Chapter 11: VRRP	This chapter describes how to configure the Virtual Router Redundancy Protocol on the Vyatta OFR.	109
Chapter 12: NAT	This chapter describes how to configure NAT on the Vyatta OFR.	116
Chapter 13: Firewall	This chapter describes how to configure the Firewall on the Vyatta OFR.	125
Chapter 14: User Authentication	This chapter describes how to set up user accounts and user authentication.	135
Chapter 15: Logging	This chapter explains how the Vyatta OFR generates and manages system logging messages, and describes how to configure logging.	140
Chapter 16: SNMP	This chapter describes how to configure Simple Network Management Protocol on the Vyatta OFR.	146
Chapter 17: Software Upgrades	This chapter describes how to use the Vyatta OFR's package mechanism to update your software.	153

Document Conventions

This guide contains advisory paragraphs and uses typographic conventions.

Advisory Paragraphs

This guide may use the following advisory paragraphs:

Warnings alert you to situations that may pose a threat to personal safety, as in the following example:

WARNING *Risk of injury. Switch off power at the main breaker before attempting to connect the remote cable to the service power at the utility box.*

Cautions alert you to situations that might cause harm to your system or damage to equipment, or that may affect service, as in the following example:

CAUTION *Risk of loss of service. Restarting a running router will interrupt service.*

Notes provide information you might need to avoid problems or configuration errors:

NOTE *You must create and configure network interfaces before enabling them for routing protocols.*

Tip: *Use tips to save time and effort.*

Tips (see left) provide helpful information for doing something in a faster or easier way, or for optimizing the performance of your system.

Typographic Conventions

In addition to advisory paragraphs, this document may use the following typographic conventions:

`Courier`	Courier font is used in command syntax sections and in special example paragraphs.
`boldface Courier`	Boldface Courier font is used to show something you enter at a command line.
boldface	Boldface font is used to represent commands or keywords inside a paragraph of ordinary text.
italics	Italic font is used to show arguments and variables, where you supply the value.

<key>	Angle brackets are used to indicate a key on your keyboard. Combinations of keys are joined by plus signs (“+”). An example is **<Ctrl>+<Alt>+<Del>**.
[*arg1*|*arg2*]	Square brackets enclose enumerated options for completing a syntax. The options are separated by a vertical bar. An example is **[enable|disable]**.
num1–numN	The typographic convention at left indicates a range of numbers. An example is 1–65535, which means 1 through 65535 inclusive.
arg1..argN	The typographic convention at left indicates a range of enumerated values. An example is **eth0..eth23**, which means **eth1**, **eth2**, **eth3**, and so on through **eth23**.
arg [*arg* …]	The typographic convention at left indicates a value that can optionally represent a space-separated list of the same kind of element (for example, a space-separated list of IP addresses).

Vyatta Publications

The Vyatta technical library includes the following publications:

Vyatta OFR Quick Start Guide	Explains how to install the router software, and provides some basic configuration to get you started.
Vyatta OFR Configuration Guide	Explains router functions, and steps through sample configurations for every function.
Vyatta OFR Command Reference	Provides a complete description of each command in the CLI.

Chapter 1: System Management

This chapter explains how to configure basic system information for the router such as host name, DNS information, and date and time.

The following topics are covered:

- System Configuration Overview
- Configuring Host Information
- Configuring DNS
- Configuring Date and Time
- Viewing System Information

System Configuration Overview

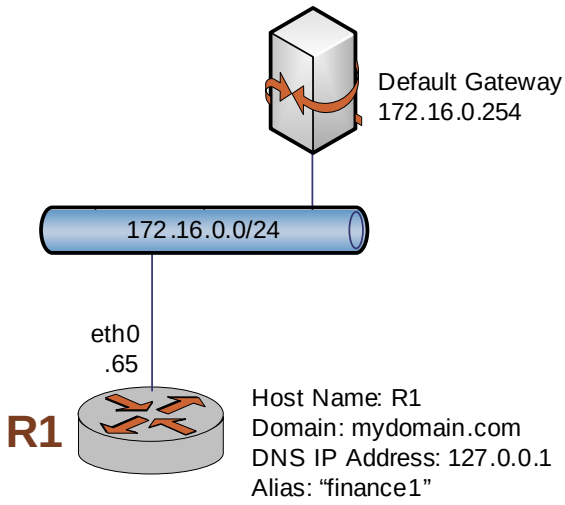
The commands in this chapter allow you to change and view basic IP system information. You can configure the following:

- Host and basic network information
- DNS information
- Date and time information

Configuring Host Information

In this section, sample configurations are presented for the router's host information. The sample configuration used is shown in Figure 1-1.

Figure 1-1 Host information



This section includes the following examples:

- Example 1-1 Setting the router's host name
- Example 1-2 Setting the router's domain
- Example 1-3 Mapping the router's IP address to its host name
- Example 1-4 Setting the default gateway
- Example 1-5 Creating an alias for the router
- Example 1-6 Showing host information

Host Name

The router's name is set using the **system host-name** command. Router names can include letters, numbers, and hyphens ("-").

Example 1-1 sets the router's host name to R1. To set the router host name, perform the following steps in configuration mode:

Example 1-1 Setting the router's host name

Step	Command

Set the router's host name.	`root@vyatta# set system host-name R1`
	`OK`
	`[edit]`

Commit the change. The command prompt changes to reflect the change	`root@vyatta# commit`
	`OK`
	`[edit]`
	`root@R1>`

Domain

The router's domain is set using the **system domain-name** command. Domain names can include letters, numbers, hyphens, and periods.

Example 1-2 sets the router's domain to **mydomain.com**.

To set the router's domain, perform the following steps in configuration mode:

Example 1-2 Setting the router's domain

Step	Command

Set the domain name.	`root@R1# set system domain-name mydomain.com`
	`OK`
	`[edit]`

Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

IP Address

The router's IP address can be statically mapped to its host name for local DNS purposes, using the **system static-host-mapping** command.

IP networks are specified in CIDR format—that is, in *ip-address*/*prefix* notation such as 192.168.12.0/24. For single addresses, use dotted quad format, that is, *a.b.c.d*. For network prefixes, enter a decimal number from 1 through 32.

A good practice is to map the router's host name to the loopback address, as the loopback interface is the most reliable on the router. By convention, the loopback interface is configured with the address 127.0.0.1. This is the address configured for the loopback interface in the sample topology used in this guide.

Example 1-3 creates a static mapping between the router's host name R1 and IP address 127.0.0.1. This is the IP address the DNS server will use to resolve DNS requests for **R1.mydomain.com**.

To map the host name to the IP address, perform the following steps in configuration mode:

Example 1-3 Mapping the router's IP address to its host name

Step	Command

Create a mapping between the host name and the IP address of the loopback interface.	`root@R1# set system static-host-mapping host-name R1 inet 127.0.0.1`
	`OK`
	`[edit]`

Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

Default Gateway

Example 1-4 specifies a default gateway for the router at 172.16.0.254. This is done by creating a static route to the gateway on network 0.0.0.0/0.

To specify the default gateway, perform the following steps in configuration mode:

Example 1-4 Setting the default gateway

Step	Command

Specify the default gateway.	`root@R1# set protocols static route 0.0.0.0/0 next-hop 172.16.0.254`
	`OK`
	`[edit]`

Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

Aliases

You can define one or more aliases for the router by mapping the router's IP address to more than one host name.

Example 1-5 creates the alias **finance1** for the router.

To create an alias for the router, perform the following steps in configuration mode:

Example 1-5 Creating an alias for the router

Step	Command

Define an alias.	`root@R1# set system static-host-mapping host-name R1 alias finance1`
	`OK`
	`[edit]`

Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

Showing Host Information

You can display host information using the **show host** command.

Example 1-7 displays the host information configured so far.

To show host information, perform the following steps in operational mode:

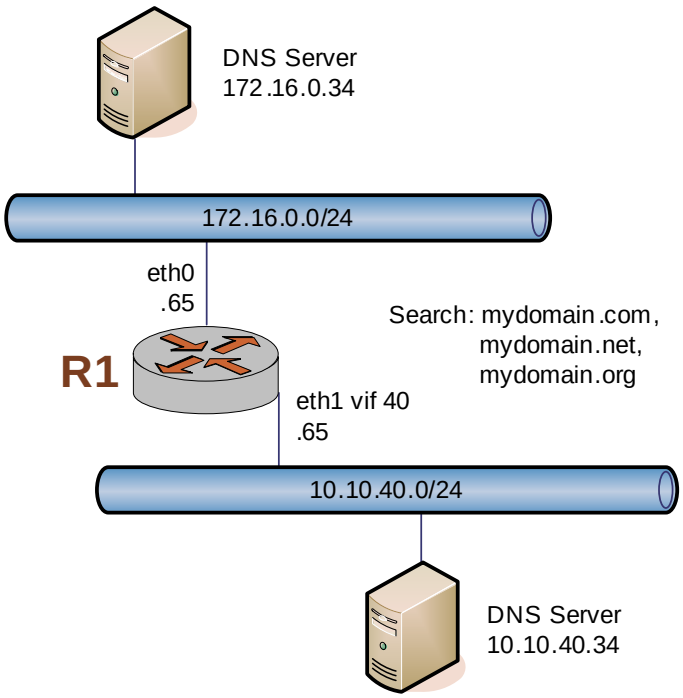
Example 1-6 Showing host information

Step	Command
See the name configured for the host.	`root@R1> show host name` `R1`
Show host by name, and see which IP address has been used for DNS resolution.	`root@R1> show host R1` `R1.mydomain.com has address 172.16.0.65`

Configuring DNS

In this section, sample configurations are presented for DNS information. The DNS configuration used is shown in Figure 1-2.

Figure 1-2 DNS



This section includes the following examples:

- Example 1-7 Specifying DNS name servers
- Example 1-8 Setting search order for domain completion

DNS Name Servers

DNS name servers are specified using the **system name-server** command.

Example 1-7 specifies two DNS servers for the router: one at 172.16.0.34, and the other at 10.10.40.34.

To specify DNS servers, perform the following steps in configuration mode:

Example 1-7 Specifying DNS name servers

Step	Command
Specify the first DNS server.	`root@R1# set system name-server 172.16.0.34` `[edit]`
Specify the second DNS server.	`root@R1# set system name-server 10.10.40.34` `[edit]`

Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

Domain Search Order

You can specify a list of domains for the router to use to complete an unqualified host name. To define this list specify the order in which domains are searched, use the **system domain-search** command.

This command requires a space-separated list of domain names, specified in the order you want them searched. A domain name can include letters, numbers, hyphens (“-”), and periods (“.”).

Example 1-8 directs the router to attempt domain completion in the following order: first, mydomain.com; second, mydomain.net; and last mydomain.org.

To specify domain search order, perform the following steps in configuration mode:

Example 1-8 Setting search order for domain completion

Step	Command

Specify the search order.	`root@R1# set system domain-search mydomain.com mydomain.net mydomain.org`
	`OK`
	`[edit]`

Example 1-8 Setting search order for domain completion

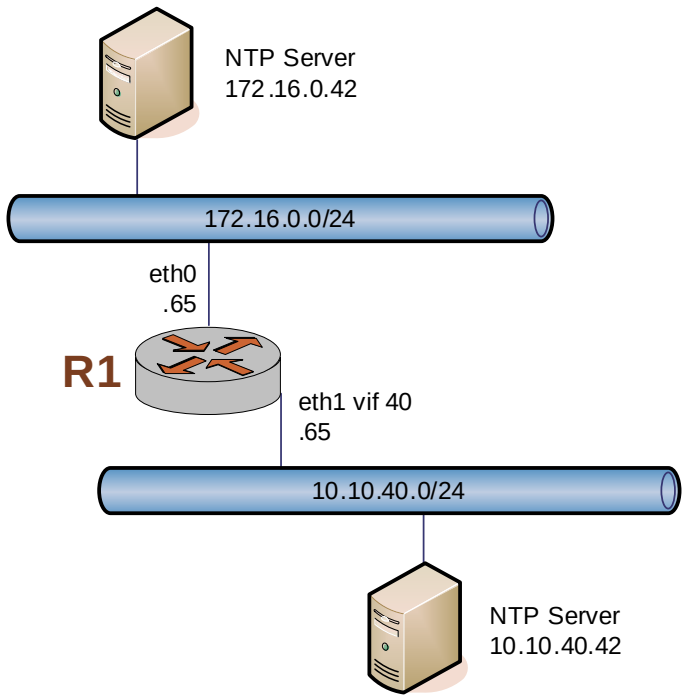
Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

Configuring Date and Time

Date and time can either be set manually, or obtained by manually or automatically synchronizing the router with one or more Network Time Protocol (NTP) servers. Time zone must be manually set, and may be specified as an offset from Universal Coordinated Time (UTC) or as one of a number of supported literal time zones.

In this section, sample configurations are presented for maintaining date and time information. The sample configuration used is shown in Figure 1-3.

Figure 1-3 Date and Time



This section includes the following examples:

- Example 1-9 Setting the date and time manually
- Example 1-10 Manually synchronizing the router with an NTP server
- Example 1-11 Showing the system date and time

- Example 1-12 Setting the time zone as an offset from UTC
- Example 1-13 Setting the time zone as a time zone name
- Example 1-14 Using NTP for automatic synchronization
- Example 1-15 Showing NTP server configuration and connection status

Setting the Date

Example 1-9 manually sets the date to 1:15 PM exactly on October 30, 2005. The format is *MMDDHHMMYYYY*, and must be enclosed in double quotes.

To manually set the date, perform the following steps in operational mode:

Example 1-9 Setting the date and time manually

Step	Command

Specify the date. The format is *MMDDHHMMYYYY*, enclosed in double quotes.	`root@R1> date “103013152005”`
	`Sun Oct 30 13:15:00 UTC 2005`
	`root@R1>`

Manually Synchronizing with an NTP Server

Example 1-10 manually synchronizes the system clock with the NTP server at 172.16.0.42.

Note that this does not set up an ongoing association with the NTP server; it merely performs a one-time synchronization. For information about setting up automatic synchronization, please see “Using NTP for automatic synchronization” on page 12.

To manually synchronize the router with an NTP server, perform the following steps in operational mode:

Example 1-10 Manually synchronizing the router with an NTP server

Step	Command

Specify the location of the NTP server.	`root@R1> date ntp 172.16.0.42`
	`Sun Nov 13 14:04:29 UTC 2005`
	`root@R1>`

Showing the Date and Time

To view the time according to the system clock, use the **show host date** command in operational mode, as shown in Example 1-11:

Example 1-11 Showing the system date and time

```
root@vyatta> show host date
Mon Oct 31 10:34:41 PST 2005
root@vyatta>
```

Setting the Time Zone

Time zone must be configured, using **system time-zone** command. To do this, you specify the amount by which your time zone is offset from UTC (coordinated universal time). UTC has the same time as the Greenwich time zone. The string giving the offset is enclosed in quotes.

The offset you specify is added to UTC to produce the local time. You can also use one of the support time zone name to indicate time zone. Again, the string supplying the time zone name must be enclosed in quotes.

Note that the router uses POSIX-style offsets. The POSIX specification uses positive signs west of Greenwich—not positive signs east of Greenwich, which some other systems use. For example, an offset of "**GMT +4**" corresponds to 4 hours behind UTC (that is, west of Greenwich).

Example 1-12 sets the time zone to 8 hours west of Greenwich, which is Pacific Standard Time.

To set the time zone using an offset from UTC, perform the following steps in configuration mode:

Example 1-12 Setting the time zone as an offset from UTC

Step	Command

Set the time zone.	`root@R1# set system time-zone "GMT+8"`
	`OK`
	`[edit]`
	`root@R1#`

Commit the information.	`root@R1# commit`
	`OK`
	`[edit]`

The following time zone names, enclosed in double quotes, are also accepted:

"Los Angeles": Sets the time zone to Los Angeles time.

"New York": Sets the time zone to New York time.

"Denver": Sets the time zone to Denver time.

“Chicago”: Sets the time zone to Chicago time.

“Anchorage”: Sets the time zone to Anchorage time.

“Honolulu”: Sets the time zone to Honolulu time.

“Phoenix”: Sets the time zone to Phoenix time.

The default is “**GMT**”, which uses UTC time exactly.

Example 1-13 sets the time zone to New York time.

To set the time zone using a supported time zone name, perform the following steps in configuration mode:

Example 1-13 Setting the time zone as a time zone name

Step	Command

Specify the time zone as a time zone name.	`root@R1# set system time-zone “New York”`
	`OK`
	`[edit]`
	`root@R1#`

Commit the information.	`root@R1# commit`
	`OK`
	`[edit]`

Using NTP for Automatic Synchronization

To use NTP for automatic synchronization, you must create associations with the NTP servers. To create an association with an NTP server, use the **system ntp-server** command and specify the IP address of the server.

Example 1-14 configures two NTP servers: one at 172.16.0.42, and one at 10.10.40.42.

To specify NTP servers, perform the following steps in configuration mode:

Example 1-14 Using NTP for automatic synchronization

Step	Command
Specify a server at 176.16.0.42.	`root@R1# set system ntp-server 172.16.0.42` `[edit]`
Specify a server at 10.10.40.42.	`root@R1# set system ntp-server 10.10.40.42` `[edit]`

Commit the information.	`root@R1# commit`
	`OK`
	`[edit]`

Showing NTP Server Configuration

To see the NTP servers configured for the router and see their connection status, use the **show ntp associations** command in operational mode, as shown in Example 1-15:

Example 1-15 Showing NTP server configuration and connection status

```
root@vyatta> show ntp associations no-resolve
remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
176.16.0.42   0.0.0.0     16 u    - 1024    0    0.000    0.000 4000.00
10.10.40.42   0.0.0.0     16 u    - 1024    0    0.000    0.000 4000.00
```

Viewing System Information

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view system configuration by using the **show system** command, as shown in Example 1-16.

To show the information in the **system** configuration node, perform the following step in configuration mode:

Example 1-16 Viewing the “system” configuration node

Step	Command

Show the contents of the **system** configuration node.	root@R1# **show system**
	host-name: “R1”
	domain-name: “mydomain.com”
	time-zone: “GMT+8”
	ntp-server: “ntp.vyatta.com”
	ntp-server: “172.16.0.42”
	ntp-server: “10.10.40.42”
	static-host-mapping {
	host-name R1 {
	inet: 172.16.0.65
	}
	login {
	user root {
	authentication {
	encrypted-password: “$1$$Ht7gBYnxI1xCd0/JOnodh.”
	}
	}
	user vyatta {
	authentication {
	encrypted-password: “$1$$Ht7gBYnxI1xCd0/JOnodh.”
	}
	}
	}
	package {
	repository “vyatta/1.0” {
	host-name: “archive.vyatta.com”
	component main {
	description: “1.0-MainPackages”
	}
	component security {
	description: “1.0-SecurityUpdates”
	}
	}
	}
	}
	
	[edit]
	root@R1#

Chapter 2: Ethernet and VLAN Interfaces

This chapter explains how to configure Ethernet interfaces, the loopback interface, and VLAN interfaces.

The following topics are covered:

- Ethernet Interfaces Overview
- Configuring Physical Ethernet Interfaces
- Configuring the Loopback Interface
- Configuring VLANs
- Monitoring Ethernet Interfaces

Ethernet Interfaces Overview

A router receives packets via its network interfaces from its neighboring routers. Some of those packets will be destined for the router itself, but most of them will normally be forwarded on via another network interface to another router or to locally connected hosts.

There are many different types of interfaces. This section deals with defining configuration for Ethernet interfaces.

Virtual Interfaces (Vifs)

VLANs are identified by a 4-byte tag that is inserted in the front of the Layer 2 Ethernet header. Having this additional tag means that interfaces configured for 802.1q are not compatible with standard Ethernet packets.

Like a physical Ethernet interface, each vif can have multiple addresses assigned to it. If you are using 802.1q VLANs, create vif configuration nodes beneath the physical interface and assign the IP address to the vif. If you are not using 802.1q, but you want to have multiple networks on the same physical interface (that is, you want to use multinetting, but not VLANs), simply create multiple **address** configuration nodes directly under the physical interface, without using vifs.

In the Vyatta OFR, an Ethernet interface may be used simultaneously as a standard port and an 802.1q port. To do this, configure a vif for the interface, and assign the VLAN ID for the interface to the vif. On Ethernet interfaces, a vif is always a VLAN interface, and its identifier is the VLAN ID.

This feature may not be compatible with all Ethernet switches: some switches require a physical Ethernet interface to be exclusively either a 802.1q interface or a standard Ethernet interface.

Enabling Interfaces

The Vyatta OFR can only use interfaces that are available to the operating system kernel (that is, interfaces that physically exist on the system) and have been created in the configuration tree.

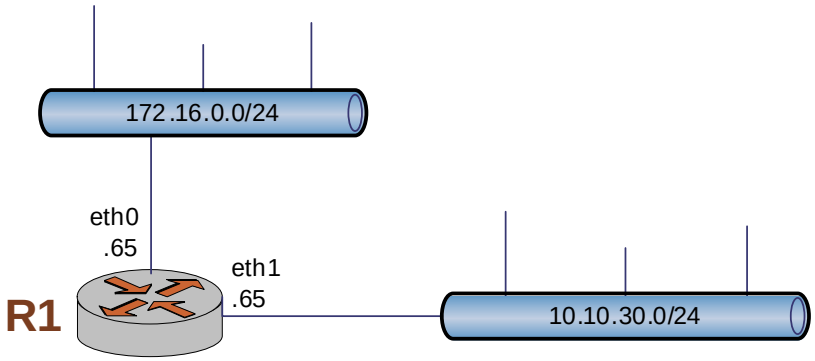
The Vyatta OFR automatically creates configuration nodes for all available physical interfaces on startup.

If you want to use an interface with a specific function (say, BGP) the interface must be enabled within the configuration node for that function (for example, within the BGP configuration node.

Configuring Physical Ethernet Interfaces

In this section, sample configurations are presented for Ethernet interfaces. The sample configuration is shown in Figure 2-1.

Figure 2-1 Ethernet interfaces



This section includes the following examples:

- Example 2-1 Viewing available system interfaces
- Example 2-2 Creating and configuring Ethernet interfaces

Viewing Available Interfaces

You can only configure interfaces that actually are available to the operating system on the hardware you are using. To view all the interfaces known to the operating system, use the **show interfaces system** command in operational mode, as shown in Example 2-1:

Example 2-1 Viewing available system interfaces

```
root@vyatta> show interfaces system
```

Configuring Ethernet Interfaces

In the Vyatta OFR router, most configuration can be applied either directly to the physical interface, or to a *virtual interface* (vif), which is a logical interface created for the physical interface. When the router starts up, it automatically detects the physical interfaces available on your device and creates configuration nodes for them. For example, on a system with two Ethernet interfaces, the router automatically creates configuration nodes for **eth0** and **eth1**.

Ethernet vifs are used only when 802.1Q VLANs are to be supported. In a basic Ethernet configuration, such as that for trial or evaluation or for a simple network topology, it will often be simplest and adequate to apply IP addresses directly to the physical interface.

Each physical interface can have multiple IP addresses assigned to it.If you want to have multiple networks on the same physical interface (that is, if you want to use multinetting, but not VLANs), simply create multiple **address** configuration nodes directly under the primary interface.

This sequence applies IP addresses directly to the two Ethernet interfaces already configured for the system—eth0 and eth1. These interfaces were automatically created by the system on startup, when the system detected the physical interfaces. Each IP address is applied directly to the interface.

To create and configure Ethernet interfaces, perform the following steps in configuration mode:

Example 2-2 Creating and configuring Ethernet interfaces

Step	Command
Create the configuration node for eth0. You can create the node completely down to the first IP address, and set the prefix length, with the same **set** statement.	`root@R2# set interfaces ethernet eth0 address 172.16.0.65 prefix-length 24` `[edit]`
Assign an IP address to interface eth1.	`root@R1# set interfaces ethernet eth1 address 10.10.30.65 prefix-length 24` `[edit]`

Example 2-2 Creating and configuring Ethernet interfaces

Commit and view the configuration.	root@R1# **commit**
	OK
	[edit]
	root@R1# **show interfaces**
	loopback lo {
	}
	ethernet eth0 {
	address 172.16.0.65 {
	prefix-length: 24
	}
	}
	ethernet eth1 {
	address 10.10.30.65 {
	prefix-length: 24
	}
	}
	
	[edit]
	root@R1#

Configuring the Loopback Interface

The loopback interface is a special software-only interface that emulates a physical interface and allows the router to “connect” to itself. Packets routed to the loopback interface are rerouted back to the router and processed locally. Packets routed out the loopback interface but not destined for the loopback interface are dropped.

The loopback interface provides a number of advantages:

- As long as the router is functioning, the loopback interface is always up, and so is very reliable. As long as there is even one functioning link to the router, the loopback interface can be accessed. The loopback interface thus eliminates the need to try each IP address of the router until you find one that is still up.
- Because the loopback interface is always up, a routing session (such as a BGP session) can continue even if the outbound interface fails.
- You can simplify collection of management information by specifying the loopback interface as the interface for sending and receiving management information such as logs and SNMP traps.
- The loopback interface can be used as to increase security, by filtering incoming traffic using access control rules that specify the local interface as the only acceptable destination.

- In OSPF, you can advertise a loopback interface as an interface route into the network, regardless of whether physical links are up or down. This increases reliability, since the the routing traffic is more likely to be received and subsequently forwarded.
- In BGP, parallel paths can be configured to the loopback interface on a peer device. This provides improved load sharing.

The router automatically creates the loopback interface on startup, with an interface name of **lo**. You must configure an IP address for the interface. The IP address for the loopback interface must be unique, and must not be used by any other interface.

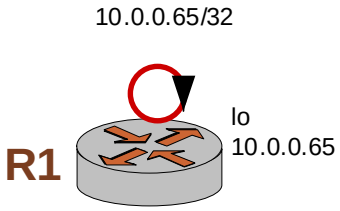
When configuring the router, it is good practice to take advantage of the loopback interface's reliability:

- The router's hostname should be mapped to the loopback interface address, rather than a physical interface.
- In OSPF and BGP, the router ID should be set to the loopback address.
- The network for the loopback interface can be small, since IP address space is not a consideration in this cse. Often a prefix of /32 is assigned.

NOTE *In some systems, the IP address 127.0.0.0 is assigned to the loopback interface by convention. However, in the Vyatta OFR the network 127.0.0.0/8 is reserved for XORP to communicate between processes. As a result, no IP address on this reserved network may be configured on any interface. Any other network may be assigned to the loopback interface.*

This sequence assigns the address 10.0.0.65 to the loopback interface. The subnet is set to /32. When you have finished, the interface will be configured as in Figure 2-3.

Figure 2-2 Configuring the loopback interface



To create and configure the loopback interface, perform the following steps in configuration mode:

Example 2-3 Configuring the loopback interface

Step	Command
Assign the IP address to the loopback interface	`root@R1# set interfaces loopback lo address 10.0.0.65 prefix-length 32` `[edit]`

Commit and view the configuration.	`root@R1# commit`
	`OK`
	`[edit]`
	`root@R1# show interfaces`
	`loopback lo {`
	`address 10.0.0.65 {`
	`prefix-length: 32`
	`}`
	`}`
	`ethernet eth0 {`
	`address 172.16.0.65 {`
	`prefix-length: 24`
	`}`
	`}`
	`ethernet eth1 {`
	`address 10.10.30.65 {`
	`prefix-length: 24`
	`}`
	`}`
	
	`[edit]`
	`root@R1#`

Configuring VLANs

VLANs are identified by a 4-byte tag that is inserted in the front of the Layer 2 Ethernet header. Having this additional tag means that interfaces configured for 802.1q are not compatible with standard Ethernet packets.

Like a physical Ethernet interface, each vif can have multiple addresses assigned to it. If you are using 802.1q VLANs, create vif configuration nodes beneath the physical interface and assign the IP address to the vif. If you are not using 802.1q, but you want to have multiple networks on the same physical interface (that is, you want to use multinetting, but not VLANs), simply create multiple **address** configuration nodes directly under the physical interface, without using vifs.

In the Vyatta OFR, an Ethernet interface may be used simultaneously as a standard port and an 802.1q port. To do this, configure a vif for the interface, and assign the VLAN ID for the interface to the vif. On Ethernet interfaces, a vif is always a VLAN interface, and its identifier is the VLAN ID.

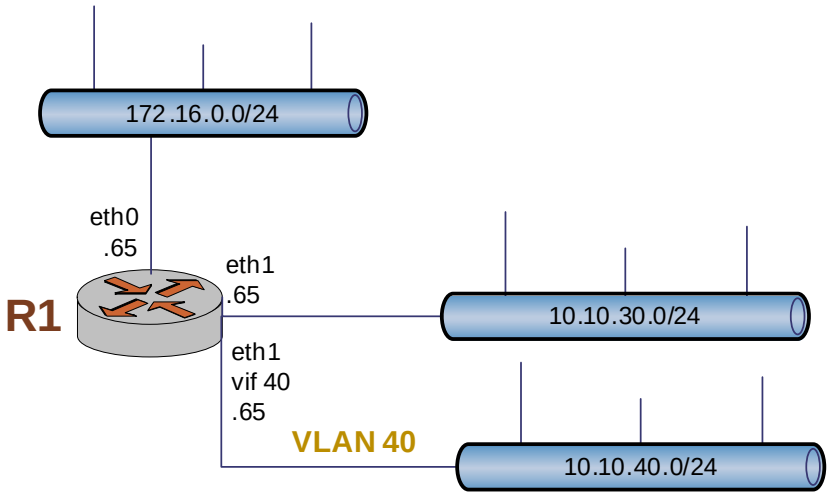
This feature may not be compatible with all Ethernet switches: some switches require a physical Ethernet interface to be exclusively either a 802.1q interface or a standard Ethernet interface.

This sequence configures a VLAN interface on router R1—vif 40 on eth1. The vif identifier is the VLAN ID, and this vif connects to VLAN 40. After configuring this VLAN, router R1 will have:

- One interface (eth0) that is configured as only a standard Ethernet interface
- One interface (eth1) that is configured as both a standard interface (IP address 10.10.30.65 applied directly to the interface) and as an 802.1q interface with one logical VLAN interface (IP address 10.10.40.65 applied to vif 40).

When you have finished, the interfaces will be configured as in Figure 2-3.

Figure 2-3 VLAN configuration



To create and configure a VLAN interface, perform the following steps in configuration mode:

Example 2-4 Creating and configuring Ethernet interfaces

Step	Command
Create the configuration node for vif 40 on eth1 and assign an IP address.	`root@R1# set interfaces ethernet eth1 vif 40 address 10.10.40.65 prefix-length 24` `[edit]`

Commit and view the configuration.	`root@R1# commit`
	`OK`
	`[edit]root@R1# show interfaces`
	`loopback lo {`
	`address 10.0.0.65 {`
	`prefix-length: 32`
	`}`
	`}`
	`ethernet eth0 {`
	`address 172.16.0.65 {`
	`prefix-length: 24`
	`}`
	`}`
	`ethernet eth1 {`
	`address 10.10.30.65 {`
	`prefix-length: 24`
	`}`
	`vif 40 {`
	`address 10.10.40.65 {`
	`prefix-length: 24`
	`}`
	`}`
	`}`
	
	`[edit]`
	`root@R1#`

When you refer to a vif within an **interfaces ethernet** command (such as **set interfaces ethernet** or **show interfaces ethernet**) you refer to it as **ethernet** *int-name* **vif** *vif-id*, as shown in the following example:

```
show interfaces ethernet eth1 vif 40
```

When you refer to the same vif within other commands, you refer to it as *int.vif*, as shown in bold in the following example:

```
set protocols rip interface eth1.40 address 10.10.40.65
```

Monitoring Ethernet Interfaces

You can use the following operational commands to monitor Ethernet interfaces. Please see the *Vyatta OFR Command Reference* for details on these commands and their options.

Command	Description
`show interfaces ethernet`	Shows information for Ethernet interfaces.
`show interfaces system`	Shows information for all interfaces available to the Linux kernel.

Chapter 3: Serial Interfaces

This chapter describes how to configure serial interfaces on the Vyatta OFR.

The following topics are covered:

- Serial Interfaces Overview
- Configuring Serial Interfaces
- Monitoring Serial Interfaces

Serial Interfaces Overview

A router receives packets via its network interfaces from its neighboring routers. Some of those packets will be destined for the router itself, but most of them will normally be forwarded on via another network interface to another router or to locally connected hosts.

There are many different types of interface. This section deals with defining configuration for serial interfaces.

Virtual Interfaces (Vifs)

The Vyatta OFR router distinguishes between physical interfaces (*interfaces*), and logical interfaces (*virtual interfaces*, or *vifs*).

Every physical network device in the system is considered to be an "interface." An example of a interface is a physical port on a serial card. Every serial interface has zero or more corresponding vifs.

On serial interfaces, physical line characteristics are specific for the interface, but encapsulation (Cisco HDLC, Frame Relay, or Point-to-Point Protocol) is specified for vifs.

Unlike Ethernet interfaces, a physical serial interface cannot directly have a configured IP address. Instead, the IP address must be assigned to the vif.

Note that each serial vif can support exactly one IP address

Enabling Interfaces

The Vyatta OFR can only use interfaces that are available to the operating system kernel (that is, interfaces that physically exist on the system) and have been created in the configuration tree and configured with an IP address.

The Vyatta OFR automatically creates configuration nodes for all available physical interfaces on startup.

If you want to use an interface with a specific function (say, BGP) the interface must be enabled within the configuration node for that function (for example, within the BGP configuration node.

Configuring Serial Interfaces

The system will automatically discover any available physical serial interfaces on startup. Before you can apply any configuration to a serial interface, a vif must be “created” for the interface and its encapsulation specified in the configuration tree.

For serial interfaces, physical line characteristics are applied to the interface as a whole. Encapsulation characteristics are applied to the vif, as shown in the configuration hierarchy below:

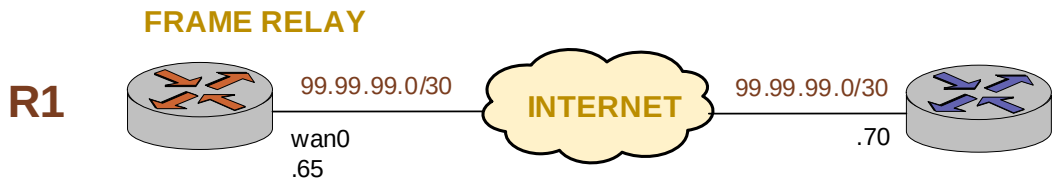
```
interfaces {
   serial wan0 {
      ppp {
         vif 1 {
         }
      }
   }
}
```

The current implementation supports Cisco HDLC, Frame Relay, and Point-to-Point Protocol encapsulation.

- Cisco HDLC and point-to-point interfaces support only one vif, and this vif must have the identifier “1”.
- The identifier for Frame Relay vifs is the DLCI number. This can range from 16 through 991.
- Currently, any vif on a serial interface can support exactly one IP address.

In this section, a sample Frame Relay configuration is presented for a serial interface on router R1. The sample configuration is shown in Figure 3-1.

Figure 3-1 Network interfaces



This section includes the following examples:

- Example 3-1 Viewing available system interfaces

- Example 3-2 Creating and configuring a Frame Relay interface

Viewing Available Interfaces

You can only configure interfaces that actually are available to the operating system on the hardware you are using.

To view all the interfaces known to the operating system, use the **show interfaces system** command in operational mode, as shown in Example 3-1:

Example 3-1 Viewing available system interfaces

```
root@R1> show interfaces system
```

Configuring a Frame Relay Interface

Example 3-2 sets up a Frame Relay interface on interface wan0. In this example:

- A Sangoma A101 T1/E1 serial card is connected to the interface.
- The physical line is a T1 line.
- The interface has one vif, which has DLCI number 16.
- The local IP is 99.99.99.65. This is in the public IP range, since this interface will connect over the wide-area network.
- The IP address of the far end is 99.99.99.70. This is on the same network (prefix-length 24) as this interface.

To create and configure this serial interface, perform the following steps in configuration mode:

Example 3-2 Creating and configuring a Frame Relay interface

Step	Command
Specify the kind of physical line the interface will be using.	`root@R1# set interfaces serial wan0 t1-options` `[edit]`
Set the line encapsulation to Frame Relay.	`root@R1# set interfaces serial wan0 encapsulation frame-relay` `[edit]`
Create a vif with DCLI 16.	`root@R1# set interfaces serial wan0 frame-relay vif 16` `[edit]`

Example 3-2 Creating and configuring a Frame Relay interface

Assign the local IP address to the vif.	`root@R1# set interfaces serial wan0 frame-relay vif 16 address local-address 99.99.99.65` `[edit]`
Set the network mask (prefix length) for the vif.	`root@R1# set interfaces serial wan0 frame-relay vif 16 address prefix-length 24` `[edit]`
Set the IP address of the far end of the connection.	`root@R1# set interfaces serial wan0 frame-relay vif 16 address remote-address 99.99.99.70` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

View the configuration.	`root@R1# exit`
	`[edit]`
	`root@R1> show interfaces serial wan0`
	`encapsulation: "frame-relay"`
	`description: "sangoma-t1/e1"`
	`t1-options {`
	`}`
	`frame-relay {`
	`vif 16 {`
	`address {`
	`local-address: 99.99.99.65`
	`prefix-length: 24`
	`remote-address: 99.99.99.70`
	`}`
	`}`
	`}`
	`root@R1>`

Monitoring Serial Interfaces

You can use the following operational commands to monitor serial interfaces. Please see the *Vyatta OFR Command Reference* for details on these commands and their options.

Command	Description
`show interfaces serial`	Shows information for serial interfaces.
`show interfaces system`	Shows information for interfaces available to the Linux kernel.

Chapter 4: Basic Services

This chapter explains how to configure system services such as DHCP, HTTP, Telnet, and SSH.

The following topics are presented:

- Basic Services Overview
- DHCP
- Access Protocols: HTTP, Telnet, and SSH
- Viewing Service Information

Basic Services Overview

The commands in this chapter allow you to enable basic protocol services. You can configure the following:

- DHCP
- Access Protocols: HTTP, Telnet, and SSH

DHCP

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

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

To use the DHCP service, you define a pool of IP addresses for each subnet assigned by the DHCP server. Each DHCP address pool is associated with an Ethernet interface on the router. For each address pool, you can specify the length of time an address will be valid (its lease duration). The default lease duration is 24 hours. You can also specify the DNS and WINS servers available to clients on the subnet.

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

The router supports DHCP relay.

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

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

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

Examples in This Section

This section includes the following examples:

- Example 4-1 Configuring DHCP address pools
- Example 4-2 Setting up DHCP relay
- Example 4-3 Displaying DHCP lease information
- Example 4-4 Viewing DHCP statistics

Configuring DHCP Address Pools

Configure DHCP address pools if you want the router to act as a DHCP server for the network.

Example 4-1 creates three address pools:

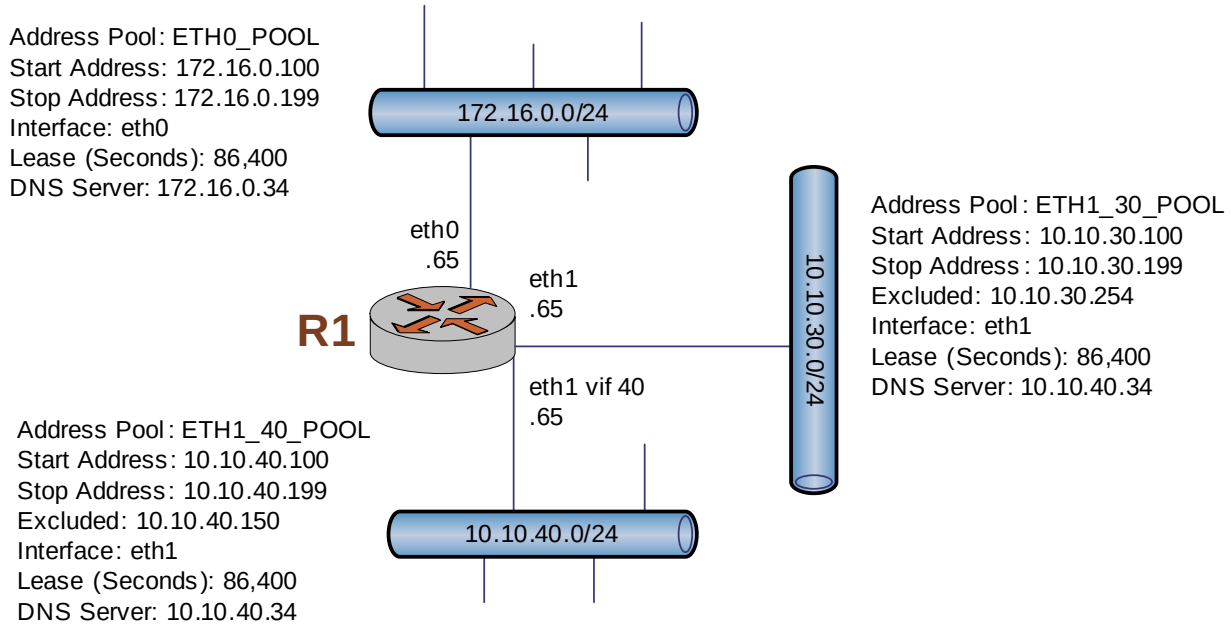
- **ETH0_POOL**. This address pool serves subnet 172.16.0.0/24, which is connected to the router interface eth0. The lease time will remain at the default, 24 hours (86,400 seconds). This address pool is able to use the DNS name server at 172.16.0.34.
- **ETH1_30_POOL**. This address pool serves subnet 10.10.30.0/24, which is connected directly to interface eth1. The lease time will remain at the default, 24 hours (86,400 seconds). This address pool will use the DNS name server at 10.10.40.34, which is directly connected to eth1.40 (that is, eth1 vif 40).
- **ETH1_40_POOL**. This address pool serves subnet 10.10.40.0/24, which is also connected to interface eth1.40. The lease time will remain at the default, 24 hours (86,400 seconds). This address pool will use the DNS name server at 10.10.40.34, which is connected to eth1.40.

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

- The addresses for the network routers in the sample topology, and for the servers configured in “Chapter 1: System Management,” fall below this range.
- The address for the default gateway (.254) and the broadcast address (.255) fall above it.
- As an example of specific address exclusion, the address pool ETH1_40_POOL excludes one additional address: 10.10.40.150.

Figure 4-1 shows the sample address pool configuration.

Figure 4-1 DHCP



To configure DHCP address pools, perform the following steps in configuration mode:

Example 4-1 Configuring DHCP address pools

Step	Command
Create the configuration node for ETH0_POOL. Specify the start and stop IP addresses for the pool.	`root@R1# set service dhcp-server name ETH0_POOL start 172.16.0.100 stop 172.16.0.199` `[edit]`
Specify the size of the network served by ETH0_POOL.	`root@R1# set service dhcp-server name ETH0_POOL network-mask 24` `[edit]`
Bind ETH0_POOL to interface eth0.	`root@R1# set service dhcp-server name ETH0_POOL interface eth0` `[edit]`
Specify a DNS server for ETH0_POOL.	`root@R1# set service dhcp-server name ETH0_POOL dns-server 172.16.0.34` `[edit]`

Example 4-1 Configuring DHCP address pools

Create the configuration node for ETH1_30_POOL. Specify the start and stop IP addresses for the pool.	`root@R1# set service dhcp-server name ETH1_30_POOL start 10.10.30.100 stop 10.10.30.199` `[edit]`
Specify the size of the network served by ETH1_30_POOL.	`root@R1# set service dhcp-server name ETH1_30_POOL network-mask 24` `[edit]`
Bind ETH1_30_POOL to interface eth1.	`root@R1# set service dhcp-server name ETH1_30_POOL interface eth1` `[edit]`
Specify a DNS server for ETH1_30_POOL.	`root@R1# set service dhcp-server name ETH1_30_POOL dns-server 10.10.40.34` `[edit]`
Create the configuration node for ETH1_40_POOL. Specify the start and stop IP addresses for the pool.	`root@R1# set service dhcp-server name ETH1_40_POOL start 10.10.40.100 stop 10.10.40.199` `[edit]`
Specify the size of the network served by ETH1_40_POOL.	`root@R1# set service dhcp-server name ETH1_40_POOL network-mask 24` `[edit]`
Exclude address 10.10.40.150 from the address pool.	`root@R1# set service dhcp-server name ETH1_40_POOL exclude 10.10.40.150` `[edit]`
Bind ETH1_40_POOL to interface eth1.	`root@R1# set service dhcp-server name ETH1_40_POOL interface eth1` `[edit]`
Specify a DNS server for ETH1_40_POOL.	`root@R1# set service dhcp-server name ETH1_40_POOL dns-server 10.10.40.34` `[edit]`

Commit the information.	`root@R1# commit`
	`OK`
	`[edit]`

Setting Up DHCP Relay

Configure DHCP relay if you want the router to forward DHCP relay to another DHCP server.

Example 4-2 does the following:

- Directs the router to forward client-to-server DHCP messages out through interface eth0 to the DHCP server at 172.16.1.52.
- Enables relay options. This directs the router to add the Relay Agent Information option (option 82) to the DHCP message before forwarding, as specified by RFC 3046.
- Re-forwarding of DHCP messages will not be permitted by this router. If a packet is received that already contains relay information, the packet is discarded.
- Other relay option parameters are left at default values. This means that the router will use port 67 for DHCP messaging, will allow a maximum DHCP packet size of at most 576 bytes, and will have a maximum hop count of 10 hops.

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

Example 4-2 Setting up DHCP relay

Step	Command
Enable DHCP relay on interface eth0, and specify the IP address of the DHCP server.	`root@R1# set service dhcp relay interface eth0 server 172.16.1.52` `[edit]`
Enable relay options.	`root@R1# set service dhcp relay interface eth0 relay-options` `[edit]`
Set the router to discard messages containing relay information. Leave other parameters at default values.	`root@R1# set service dhcp relay interface eth0 relay-options relay-agents-packets discard` `[edit]`

Viewing DHCP Lease Information

To view DHCP lease information, use the **show dhcp lease** command in operational mode, as shown in Example 4-3. Example 4-3 uses the **pool** option, and will display lease information for address pool ETH1_40_POOL.

Example 4-3 Displaying DHCP lease information

```
root@vyatta> show dhcp lease pool ETH1_40_POOL
```

Viewing DHCP Statistics

To view DHCP lease information, use the **show dhcp statistics** command in operational mode, as shown in Example 4-3:

Example 4-4 Viewing DHCP statistics

```
root@vyatta> show dhcp statistics
```

Access Protocols: HTTP, Telnet, and SSH

The Vyatta OFR router supports access from remote systems by means of HTTP, Telnet, and SSH.

- HTTP provides remote web access to the Vyatta OFR.
- The Secure Shell (SSH) protocol provides secure encrypted communications between two hosts communicating over an insecure network. The Vyatta OFR supports both the SSH v1 and v2 protocols.
  - For SSH v1, the software supports encrypted communication using ciphers such as 3DES or Blowfish, as selected by the SSH client. The default is 3DES.
  - For SSH v2, the software supports 128-bit AES, Blowfish, 3DES, CAST128, Arcfour, 192-bit AES, or 256-bit AES, as selected by the SSH client.
- Telnet provides unencrypted communications between the router and another host. If you use SSH, we recommend that you disable Telnet access, which is not secure.

You can leave the port as the default, which is the well-known port for the protocol, or configure a non-standard port.

For security reasons, remote access to the router is disabled by default. You must configure the router explicitly (by creating the SSH and Telnet service configuration nodes) so that users on remote systems can access it.

In this section, sample configurations are presented for HTTP, Telnet, and SSH. This section includes the following examples:

- Example 4-5 Enabling HTTP access
- Example 4-6 Enabling Telnet access
- Example 4-7 Enabling SSH access
- Example 4-8 Viewing the “service” configuration node

HTTP

Configuring HTTP is optional, but creating the HTTP service will provide remote access to the router.

By default, HTTP is enabled on port 80. Example 4-5 enables HTTP on port 8080, as shown in Figure 4-2.

Figure 4-2 Enabling HTTP access

To enable the HTTP service on the router, perform the following steps in configuration mode:

Example 4-5 Enabling HTTP access

Step	Command
Create the configuration node for the HTTP service and specify the port number.	`root@R1# set service http port 8080` `[edit]`

Commit the information.	`root@R1# commit`
	`OK`
	`[edit]`

Telnet

Configuring Telnet is optional, but creating the Telnet service will allow you to access the router remotely. Example 4-6 enables Telnet on the default port (port 23), as shown in Figure 4-3.

Figure 4-3 Enabling Telnet access

To enable the Telnet service on the router, perform the following steps in configuration mode:

Example 4-6 Enabling Telnet access

Step	Command
Create the configuration node for the Telnet service.	`root@R1# set service telnet` `[edit]`

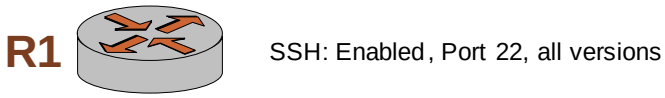
Commit the information.	`root@R1# commit`
	`OK`
	`[edit]`

SSH

Configuring SSH is optional, but creating the SSH service will provide secure remote access to the router.

Example 4-7 enables SSH on the default port (port 22), as shown in Figure 4-3. By default, only SSH version 2 is enabled, but Example 4-7 enables SSH for all versions of SSH.

Figure 4-4 Enabling SSH access



To enable the SSH service on the router, perform the following steps in configuration mode:

Example 4-7 Enabling SSH access

Step	Command
Create the configuration node for the SSH service.	`root@R1# set service ssh protocol-version all` `[edit]`

Commit the information.	`root@R1# commit`
	`OK`
	`[edit]`

Viewing Service Information

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view service configuration by using the **show service** command, as shown in Example 4-8.

To show the information in the **service** configuration node, perform the following step in configuration mode:

Example 4-8 Viewing the “service” configuration node

Step	Command
Show the contents of the **service** configuration node.	`root@R1# show service`

Chapter 5: Forwarding and Routing

This chapter provides a brief overview of basic routing topics: traffic fowarding, unicast routing, multicast routing, and multicast topology discovery.

The following topics are covered:

- Forwarding
- Unicast Routing Overview
- Multicast Routing Overview
- Multicast Topology Discovery
- Route Selection Process
- Configuring Forwarding
- Displaying Route Information

Forwarding

The forwarding engine of a router is that part that receives packets from one interface and forwards them through another interface. On the Vyatta OFR, the configuration interface for the forwarding engine is called the Forwarding Engine Abstraction (**fea**) for unicast packets, and the Multicast Forwarding Engine Abstraction (**mfea**) for multicast packets.

You might want a router in one situation to have a different forwarding policy from that in another. For example, you might want the router to forward unicast packets but not multicast packets. Configuring the forwarding engine allows you to selectively enable or disable forwarding for unicast and multicast.

On the Vyatta OFR:

- Unicast packet forwarding is enabled by default for each protocol when the protocol configuration node is created.
- Multicast packet forwarding must be explicitly enabled.

Unicast Routing Overview

To forward packets, a router maintains a forwarding table containing routes indicating to which neighboring router a packet for a particular destination should be forwarded. At the minimum, then, a route then consists of a destination subnet and a next hop.

The destination subnet is usually represented as a base IP address and a prefix-length in bits. For example, the subnet 128.16.64.0/24 has a prefix length of 24 bits, indicating that the first 24 bits of this address identify the network in question, and the last 8 bits identify hosts on this subnet. Thus, a route for this subnet would be used to forward packets for addresses 128.16.64.0 to 128.16.64.255 inclusive. The next hop can be the IP address of a neighboring router, or it might indicate that the route is for a subnet that is directly connected to this router.

Prefix Matching

IP routers perform longest prefix match forwarding. This means that a router might have more than one route that matches a destination address, and under such circumstances, it will use the route that has the longest prefix. For example, suppose a router has the following two routes:

- Subnet: 128.16.0.0/16, next hop: 10.0.0.1
- Subnet: 128.16.64.0/24, next hop: 10.0.0.2

A packet destined for 128.16.0.1 would match the first route only, and so would be forwarded to 10.0.0.1. However, a packet destined for 128.16.64.1 would match both routes, and so would be forwarded to 10.0.0.2 because the second route has a longer prefix (24 is longer than 16).

Populating the Forwarding Table

To be useful, a router needs to populate its forwarding table. It does this in three ways:

- Routes for directly connected subnets are automatically entered into the forwarding table.
- Routes may be configured via the router's configuration file or command line interface. Such routes are known as *static routes*.
- Routes may be learned from another router via a routing protocol. Such routes are known as *dynamic routes*.

Static Routes

Static routes are discussed in detail in "Chapter 7: Static Routes."

Dynamic Routing

Many different routing protocols can supply dynamic routes. The following unicast dynamic routing protocols are supported by the Vyatta OFR:

- **Routing Information Protocol (RIP).** This is probably the simplest intra-domain routing protocol, and is often used on small networks. RIP is discussed in detail in "Chapter 8: RIP"
- **Open Shortest Path First (OSPF).** Used for intra-domain routing, often on large ISP networks. OSPF is discussed in more detail in "Chapter 9: OSPF"
- **Border Gateway Protocol (BGP).** BGP is used for inter-domain routing. BGP is discussed in detail in "Chapter 10: BGP"

In addition, there are also multicast routing protocols. Of these, the Vyatta OFR supports PIM-SM and IGMP.

Route Redistribution

A common requirement is to redistribute routes between routing protocols. Examples of situations where routes might be redistributed are as follows:

- When interconnecting some statically routed subnets with some subnets using IP for dynamic routing.

Rather than configure the static routes, and additionally informing RIP to originate route advertisements for the same subnets, it is simpler and less error-prone to configure the router to redistribute all the static routes into RIP.

- When a network uses RIP internally, but also uses BGP to peer with the rest of the Internet.

  One solution would be to configure BGP at the border routes to originate route advertisements for the internal subnets. However, if a new subnet is added internally, then the border routers also need to be correctly modified. Instead, you can simply configure the border routers to redistribute RIP routes into BGP.

The Vyatta OFR redistributes routes using routing policies. Routing policies are discussed in detail in "Chapter 6: Routing Policies."

Multicast Routing Overview

IP multicast is a technology that allows one-to-many and many-to-many distribution of data on the Internet. Senders send their data to a multicast IP destination address, and receivers express an interest in receiving traffic destined for such an address. The network then determines how to get the data from senders to receivers.

If both the sender and receiver for a multicast group are on the same local broadcast subnet, the routers do not need to intervene in the process, and communication can take place directly. If, however, the sender and receiver are on different subnets, then a multicast routing protocol needs to be involved in setting up multicast forwarding state on the tree between the sender and the receivers.

Multicast Routing Protocols

Broadly speaking, there are two different types of multicast routing protocols:

- Dense-mode protocols

  In these, traffic from a new multicast source is delivered to all possible receivers. Then, subnets where there are no members request to be pruned from the distribution tree.

- Sparse-mode protocols

  In these, explicit control messages are used to ensure that traffic is only delivered to the subnets where there are receivers that have requested to receive it.

Examples of dense-mode protocols are DVMRP and PIM Dense Mode. Examples of sparse-mode protocols are PIM Sparse Mode, CBT, and MOSPF. Most of these protocols are largely historic at this time, with the exception of PIM Sparse Mode (PIM-SM) and PIM Dense Mode (PIM-DM). At this time, PIM-DM is not very widely used.

In addition to the routing protocols used to set up forwarding state between subnets, a way is needed for the routers to discover that there are local receivers on a directly attached subnet. For IPv4 this role is served by the Internet Group Management Protocol (IGMP).

Service Models: ASM vs. SSM

There are two different models for IP multicast:

- Any Source Multicast (ASM), in which a receiver joins a multicast group, and receives traffic from any senders that send to that group.
- Source-Specific Multicast (SSM), in which a receiver explicitly joins to a (source, group) pairing.

Traditionally IP multicast used the ASM model, but problems deploying inter-domain IP multicast resulted in the much simpler SSM model being proposed. In the future it is likely that ASM will continue to be used within intranets and enterprises, but SSM will be used when multicast is used inter-domain. The two models are compatible, and PIM-SM can be used as a multicast routing protocol for both. The principal difference is that ASM only requires IGMPv2 or MLDv1, whereas SSM requires IGMPv3 or MLDv2 to permit the receivers to specify the address of the sending host.

Multicast Addresses

For IPv4, multicast addresses are in the range 224.0.0.0 to 239.255.255.255 inclusive. Addresses within 224.0.0.0/24 are considered link-local and should not be forwarded between subnets. Addresses within 232.0.0.0/8 are reserved for SSM usage. Addresses in 239.0.0.0/8 are ASM addresses defined for varying sizes of limited scope.

Supported Protocols

Vyatta supports the following multicast protocols:

- PIM Sparse Mode for both ASM and SSM multicast routing for IPv4.
- IGMPv1 and IGMPv2 for IPv4 local multicast membership.

Multicast Topology Discovery

Multicast routing protocols such as PIM-SM (Protocol Independent Multicast Sparse-Mode) and PIM-DM (Protocol Independent Multicast Dense-Mode) build the multicast delivery tree by using the RPF (Reverse-Path Forwarding) information toward the root of the tree. The root could be the so-called Rendezvous Point (RP) (in case of PIM-SM) or the source itself (in case of PIM-SM or PIM-DM).

The RPF information in each router is per multicast distribution tree and is basically the next-hop neighbor router information toward the root of the tree. In other words, the RPF router is the next-hop router toward the root. In case of PIM-SM, the RPF neighbor is typically the router that a Join message is sent to.

Obviously, all multicast routers must have consistent RPF state, otherwise a Join message might never reach the root of the tree. Typically, the unicast path forwarding information is used to create the RPF information, because under normal circumstances the unicast routing provides the necessary information to all routers.

Note that the unicast-based RPF creates multicast distribution trees where each branch of the tree follows the unicast path from each leaf of the tree toward the root. Usually this is the desired behavior, but occasionally someone may want the unicast and the multicast traffic to use different paths. For example, if a site has two links to its network provider, one of the links may be used for unicast only, and the other one only for multicast.

To provide for such flexibility in the configuration, the PIM-SM and PIM-DM specifications use the Multicast Routing Information Base (MRIB) for obtaining the RPF information. Typically, the MRIB may be derived from the unicast routing table, but some protocols such as MBGP may carry multicast-specific topology information. Furthermore, the MRIB may be modified locally in each site by taking into account local configuration and preferences. A secondary function of the MRIB is to provide routing metrics for destination addresses. Those metrics are used by the PIM-SM and PIM-DM Assert mechanism.

Configuring the MRIB

The Vyatta RIB module contains a table with the MRIB. That table is propagated to the PIM-SM module and is used by PIM-SM in the RPF computation. The MRIB table inside the RIB module is completely independent from the Unicast Routing Information Base (URIB) table. The URIB table is created from the unicast routes calculated by unicast routing protocols such as BGP, OSPF and RIP. The MRIB table is created similarly, but only by those protocols that are explicitly configured to add their routes to the MRIB.

For example, if Multi-protocol BGP is enabled, then the BGP module will add multicast-specific routes to the MRIB.

Currently, Vyatta supports the following methods for adding routing entries to the MRIB:

- Multi-protocol BGP.

The BGP module can be configured to negotiate multiprotocol support with its peers. Then, the BGP multicast routes will be installed in the MRIB. See Chapter 10: BGP" for information how to configure BGP.

- Static Routes.

  The Static Routes module can be used to configure static unicast and multicast routes. The unicast routes are added to the unicast RIB, while the multicast routes are added to the MRIB. See "Chapter 7: Static Routes" for information how to configure static routes.

- FIB2MRIB.

  If there are no unicast routing protocols configured in the router to supply the MRIB routes, then the FIB2MRIB module can be used to populate the MRIB. If the FIB2MRIB module is enabled, it will register with the FEA to read the whole unicast forwarding table from the underlying system, and to receive notifications for all future modifications of that table. In other words, the FIB2MRIB's task is to replicate the unicast forwarding information on that router into the MRIB.

Route Selection Process

A router can run multiple routing protocols simultaneously. For example, you might use RIP to distribute routes within your network, but BGP to learn external routes. In some situations, this can lead to a router learning the same route from more than one routing protocol. For example, the router might learn the following two routes:

- Subnet: 128.16.64.0/24, nexthop: 192.150.187.1, learned from BGP via an external peering. AS Path: 123 567 987.
- Subnet: 128.16.64.0/24, nexthop: 10.0.0.2, learned from RIP with metric 13

The problem for the router is to choose the best of these two routes for the given address.

The longest prefix match rule does not help the router choose between these two routes, because the prefix lengths are the same and the metric used for RIP is not directly comparable against the AS path length or any other attribute attached to a BGP route.

the Vyatta OFR uses the concept of *administrative distance* to determine which route is the best—that is, which route "wins." Essentially, each routing protocol is configured with a "distance." If a route is learned from two protocols, then the version with the smallest distance wins.

The Vyatta router uses a predefined set of administrative distances. Currently, these are not configurable. These are as follows.

Table 5-1 Vyatta router predefined administrative distances

Type of Route	Distance
Directly connected subnets	0
Static routes	1
BGP, heard from external peer	20
OSPF	110
RIP	120
BGP, heard from internal peer	200
FIB2MRIB routes (Vyatta-specific, in MRIB only)	254

Configuring Forwarding

On the Vyatta OFR, unicast packet forwarding is enabled by default when the unicast protocol configuration node is created. Multicast packet forwarding must be explicitly enabled.

Example 5-1 enables multicast forwarding on interface eth0, which permits IGMP (a multicast protocol) to be configured on that interface.

Example 5-1 Enabling multicast forwarding

```
root@R1# set multicast mfea4 interface ethernet eth0
[edit]
root@R1# set protocols igmp interface eth0
[edit]
root@R1# show igmp interface
Interface    State    Querier         Timeout Version Groups
eth0         UP       10.1.0.100         None       2      3
eth1         DISABLED 0.0.0.0            None       2      0
eth2         DISABLED 0.0.0.0            None       2      0
lo           DISABLED 0.0.0.0            None       2      0
root@R1# show mfea interface
Interface    State    Vif/PifIndex Addr            Flags
eth0         UP                0/2 10.1.0.100     MULTICAST BROADCAST KERN_UP
```

```
eth1          DISABLED          1/3                MULTICAST BROADCAST
eth2          DISABLED          2/4                MULTICAST BROADCAST KERN_UP
lo            DISABLED          3/1                LOOPBACK KERN_UP
register_vif DISABLED           4/2 10.1.0.100     PIM_REGISTER KERN_UP
```

Displaying Route Information

Example 5-2 shows all routes in the RIB using the default output format (brief).

Example 5-2 “show route”: Displaying routes

```
root@vyatta> show route
Total routes: 13, Total paths: 13
10.0.0.0/8      [static(1)]     > to 192.168.2.1  via eth2/eth2
10.0.0.0/24     [connected(0)]  > to 10.0.0.50  via eth0/eth0
25.0.0.0/8      [ebgp(0)]       > to 10.0.0.100 via eth0/eth0
25.25.0.0/16    [ebgp(0)]       > to 10.0.0.100 via eth0/eth0
25.25.25.0/24   [ebgp(0)]       > to 10.0.0.100   via eth0/eth0
26.0.0.0/8      [ospf(1)]       > to 10.0.0.100 via eth0/eth0
26.26.0.0/16    [ospf(1)]       > to 10.0.0.100 via eth0/eth0
26.26.26.0/24   [ospf(1)]       > to 10.0.0.100 via eth0/eth0
27.0.0.0/8      [rip(2)]        > to 10.0.0.100 via eth0/eth0
27.27.0.0/16    [rip(2)]        > to 10.0.0.100 via eth0/eth0
27.27.27.0/24   [rip(2)]        > to 10.0.0.100 via eth0/eth0
172.16.0.0/14   [connected(0)]  > to 172.16.0.50  via eth1/eth1
192.168.2.0/24  [connected(0)]  > to 192.168.2.31 via eth2/eth2
```

The Vyatta OFR returns the longest matching unique dotted decimal entry for the specified prefix, even if the entry does not exactly match the dotted decimal subnet. I the entry does not match, only the closest (longest) matching prefix is returned.

Example 5-3 shows the longest prefix matching mechanism.

Example 5-3 “show route”: Longest prefix matching

```
root@R1> show route
Total routes: 9, Total paths: 9
0.0.0.0/0               [static(1)]     > to 10.0.0.1           via eth0
10.0.0.0/24             [connected(0)]  > to 10.0.0.234         via eth0
172.0.0.0/8             [static(1)]     > to 10.0.0.1           via eth0
172.16.0.0/16           [static(1)]     > to 10.0.0.1           via eth0
172.16.0.0/24           [static(1)]     > to 10.0.0.1           via eth0
172.16.0.0/25           [static(1)]     > to 10.0.0.1           via eth0
```

```
172.17.0.0/16           [static(1)]     > to 10.0.0.1           via eth0
172.32.0.0/24           [static(1)]     > to 10.0.0.1           via eth0
172.64.0.0/25           [static(1)]     > to 10.0.0.1           via eth0
root@R1> show route 172.16.0.255
Total routes: 1, Total paths: 1
172.16.0.0/24           [static(1)]     > to 10.0.0.1           via eth0
```

Example 5-4 displays static routes.

Example 5-4 "show route": Displaying static routes

```
root@vyatta> show route protocol static
Total routes: 13, Total paths: 13
Routes in this view: 1, Paths in this view: 1

10.0.0.0/8      [static(1)]     > to 192.168.2.1  via eth2/eth2
```

Example 5-5 displays routes with a prefix length of 16.

Example 5-5 "show route": Displaying routes of a specified prefix length

```
root@vyatta> show route prefix-length 16
Total routes: 13, Total paths: 13
Routes in this view: 2, Paths in this view: 2

25.25.0.0/16    [ebgp(0)]       > to 10.0.0.100 via eth0/eth0
26.26.0.0/16    [ospf(1)]       > to 10.0.0.100 via eth0/eth0
27.27.0.0/16    [rip(2)]        > to 10.0.0.100 via eth0/eth0
```

Example 5-6 displays routes with a next hop of 10.0.0.100.

Example 5-6 "show route": Displaying routes with a specified next hop

```
root@vyatta> show route next-hop 10.0.0.100
Total routes: 13, Total paths: 13
Routes in this view: 9, Paths in this view: 9

25.0.0.0/8      [ebgp(0)]       > to 10.0.0.100 via eth0/eth0
25.25.0.0/16    [ebgp(0)]       > to 10.0.0.100 via eth0/eth0
25.25.25.0/24   [ebgp(0)]      > to 10.0.0.100   via eth0/eth0
26.0.0.0/8      [ospf(1)]       > to 10.0.0.100 via eth0/eth0
26.26.0.0/16    [ospf(1)]       > to 10.0.0.100 via eth0/eth0
```

```
26.26.26.0/24   [ospf(1)]       > to 10.0.0.100 via eth0/eth0
27.0.0.0/8      [rip(2)]        > to 10.0.0.100 via eth0/eth0
27.27.0.0/16    [rip(2)]        > to 10.0.0.100 via eth0/eth0
27.27.27.0/24   [rip(2)]        > to 10.0.0.100 via eth0/eth0
```

Chapter 6: Routing Policies

This chapter describes the policy framework, which you can use to apply policies that influence routing behavior.

The following topics are presented:

- Policy Overview
- Creating Policies
- Policy Objects
- Policy Evaluation
- Import and Export Routing Policies
- Specifying Policy Criteria
- Regular Expressions

Policy Overview

A routing policy is a mechanism that allows you to configure criteria for comparing a route against, and for specifying the actions that will be performed on the route if the criteria are met. For example, a routing policy can be used to filter out (block) specific prefixes that are being announced by a BGP neighbor. Routing policies are also used to export routes learned via one protocol (for instance, OSPF) into another protocol (for instance, BGP). This is commonly called *route redistribution*.

Once a routing policy has been defined, then in order for the policy to take effect it must be applied to a specific routing protocol. Policies can be applied either as an *import* policy or as an *export* policy.

Import policies are evaluated for updates that are *received* via the routing protocol to which the policy has been applied. For example, if you configure an import policy for BGP, all BGP announcements received by the OFR are compared against first, prior to being added to the BGP and routing tables.

Export policies are evaluated for updates that are *transmitted* via the routing protocol to which the policy has been applied. For example, if you configure an export policy for BGP, all BGP updates originated by the OFR are compared against the policy prior to being sent to any BGP peers.

Creating Policies

Policies are configured in the CLI using the **set policy policy-statement...** command.

- Each policy statement must contain at least one term.
- Within each term are three configuration options: **from**, **to**, and **then**.

This basic policy configuration syntax results in the configuration structure shown in Example 6-1. In this structure, user-defined values are shown in italics. (These user-defined items are described below, in Table 6-1.

Example 6-1 Structure of policy statements

```
policy {
   policy-statement policy-name {
      term term-name {
         from {
            criteria operator value
         }
         to {
            criteria operator value
         }
```

```
            then {
               action value
            }
         }
      }
   }
```

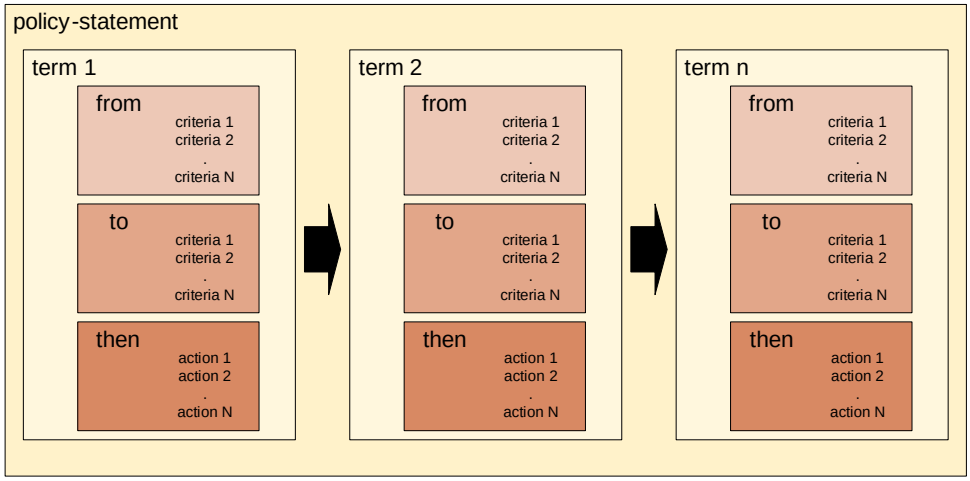


Table 6-1 describes the items that must be defined when you create a policy. (These are the user-defined items that are in italics in Example 6-2.)

Table 6-1 Configurable items in policies

Item	Description
policy-name	The policy name defines an identifier that can be used to reference the policy in other configuration sections, like routing protocols.
term-name	Every policy must have at least one term, but typically a policy will have multiple terms. The term name defines a specific term within a policy. The term contains a grouping that defines a set of criteria, together with a set of actions to be taken if all the criteria are met. A common practice is to use an integer for the term name; even so, terms are evaluated in the order in which they are entered into configuration, not by the term value.

Table 6-1 Configurable items in policies

Item	Description
criteria	Criteria are used in combination with the from and to configuration keywords to define the conditions that should be used to identify a specific route that the policy term is attempting to match. For example, you could specify a term with the **from network4** criterion defined, in order to match a specific route or prefix. If more than one match criterion is defined, the route must match all the criteria in order for the defined action(s) to be taken.
operator	Some criteria defined in terms can accept operators in addition to the criterion value. For example, a **from** option within a term could be expressed as **from prefix-length4 > 24**. In this case, the operator is the greater-than sign (“>”). This criterion would match any routes with a prefix length greater than 24.
action	Actions are uses within the **then** configuration option to define what action(s) should be performed if all the criteria for the term are matched. For example, an action could be as simple as accepting the route, or it might also include changing the next hop of the routing update.

To create a policy, you enter the appropriate set of CLI commands. Example 6-2 shows a policy that has all three configuration options: **from**, **to**, and **then**. In this example, the user-defined values are bolded.

Example 6-2 Policy with “from”, “to”, and “then” options

```
root@R1# set policy policy-statement EXAMPLE term 1 from network4
<= 10.0.0.0/8

root@R1# set policy policy-statement EXAMPLE term 1 to nexthop4
== 10.0.0.1

root@R1# set policy policy-statement EXAMPLE term 1 then action
accept
```

Policy Objects

To minimize the number of terms that have to be created within a policy statement, the Vyatta OFR supports the creation of *policy objects*. Policy objects allow you group a set of items, such as IP addresses, that should all have the same policy action performed for matching networks. For example, you can create a policy object that groups a list of networks, and then reference that policy object in a single term that directs the router to accept routes matching any of those networks from a BGP neighbor.

Policy Evaluation

A policy statement is evaluated term by term, exiting on the first completely matched term. If none of the match conditions in the terms is matched, the default action is taken:

- For import policies, the default action is to accept the route.
- For export policies, the default action is to ignore any route that does not match a term. That is, there is an implicit **reject all** action as the last term of any export policy.

Import and Export Routing Policies

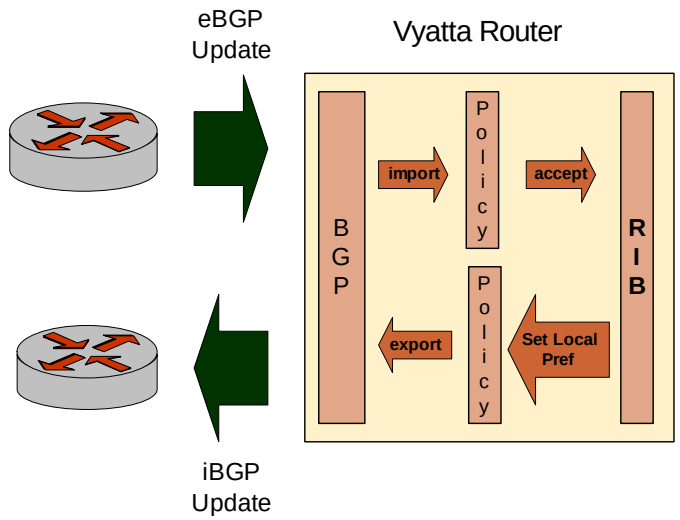
Import routing policies control the behavior of routing updates that are received by a routing protocol. For example, if you want to change the metric of routes learned via OSPF from its OSPF neighbors, you would configure an OSPF import policy.

Export routing policies control the behavior of routing updates that are sent by a routing protocol. For example, if you wanted to prevent a specific prefix from being sent to BGP neighbors, you would configure a BGP export policy.

Figure 6-1 shows the behavior for BGP when both an import and export policy-statement are configured. In this example:

- The import policy is filtering for allowed prefixes. These are the prefixes that the router should accept from its BGP neighbors.
- The export policy is setting the Local Pref for updates sent to its BGP neighbors.

Figure 6-1 BGP import and export policies



All export policies require a defined **from protocol** statement in the policy-statement term to identify from which protocol table the routes are originating. For example, if you configure a policy that includes **from protocol ospf**, the routes will come from the OSPF database. In addition to providing actions for the routes within a protocol, such as BGP, export policies can also be used to redistribute routes from one protocol into another.

Specifying Policy Criteria

Criteria are the expressions that define a match condition for a route. The match conditions specify the conditions under which a route is a candidate to have an action performed.

Criteria Operators

Some of the match criteria defined in **from** and **to** policy-statement terms can use operators in addition to the criteria value. For example, a **from** policy-statement term could include a **prefix-length4 > 24** statement. This would match routes with a prefix length greater than 24. In this case, the greater-than sign (“>”) is the operator.

If no operator is explicitly defined, each criterion has a default operator value. For example, by default, the operator for **prefix-length4** is equals (“==”).

Table 6-2 shows the definitions for policy operators.

Table 6-2 Operator Definitions

Operator	Example	Description
:	10.10.35.0**:**10.10.35.254	Specifies a range of values, such as a range of numbers or IP addresses. Example: “Is an IPv4 address between 10.10.35.0 and 10.10.35.254, inclusive.”
==	==15	Is equal to. Example: “is equal to 15.”
!=	!=0	Is not equal to. Example: “Is not equal to 0.”
<	<15	Is less than. Example: “Is less than 15.”
>	>12	Is greater than. Example: “Is greater than 12.”
<=	<=12	Is less than or equal to. Example: “Is less than or equal to 12.”
>=	>=12	Is greater than or equal to. Example: “Is greater than or equal to 12.”

The following criteria allow operators.

Table 6-3 Matching criteria allowing operators

Criterion	Matching Operators Allowed
localpref	all
med	all
metric	all
neighbor	all
network4	all
network4-list	all
nexthop	all
origin	all
prefix-length	all
tag	all

Protocol-Specific Criteria

Not every criterion in the **from**, **to**, and **then** parts of the term can be applied to every routing protocol; the applicable criteria vary with the protocol. Table 6-4 shows which options apply to which protocols.

Table 6-4 Policy Options Applicable per Protocol

from	BGP	RIP	RIPng	OSPF	Static
protocol	×	×	×	×	×
network4	×	×	×	×	×
network6	×	×	×	×	×
network4-list	×	×	×	×	×
network6-list	×	×	×	×	×
prefix-length4	×	×	×	×	×
prefix-length6	×	×	×	×	×
nexthop4	×	×		×	
nexthop6	×		×		
as-path	×				
as-path-list	×				
community	×				
community-list	×				
neighbor	×				
origin	×				
med	×				
localpref	×				
metric		×	×	×	×
external				×	
tag		×	×	×	
to	BGP	RIP	RIPng	OSPF	Static
network4	×	×	×	×	×
network6	×	×	×	×	×

Table 6-4 Policy Options Applicable per Protocol

network4-list	×	×	×	×	×
network6-list	×	×	×	×	×
prefix-length4	×	×	×	×	×
prefix-length6	×	×	×	×	×
nexthop4	×	×		×	
nexthop6	×		×		
as-path	×				
as-path-list	×				
community	×				
community-list	×				
neighbor	×				
origin	×				
med	×				
localpref	×				
was-aggregated	×				
metric		×	×	×	
external				×	
tag		×	×	×	
then	BGP	RIP	RIPng	OSPF	Static
action	×	×	×	×	×
trace	×	×	×	×	×
nexthop4	×	×		×	
nexthop6	×		×		
as-path-prepend	×				
as-path-expand	×				
community	×				
community-add	×				
community-del	×				

Table 6-4 Policy Options Applicable per Protocol

origin	×				
med	×				
med-remove	×				
localpref	×				
aggregate-prefix-len	×				
aggregate-brief-mode	×				
metric		×	×	×	
external				×	
tag		×	×	×	

Regular Expressions

Regular expressions provide the ability to perform pattern matching are used to parse data sets within AS path lists and community lists. In general, a regular expression takes the following form:

<*regex-term*><*operator*>

where <*regex-term*> is a string to be matched, and <*operator*> is one of the operators shown in Table 6-2.

Note that operators must occur immediately after <*regex-term*> with no intervening space, with the following exceptions:

- The vertical bar operator (“|”) and hyphen (“-”) operator, both of which are placed between two terms
- Parentheses, which enclose <*regex-term*>s.

Table 6-5 shows the regular expression operators supported in policy statements.

Table 6-5 Regular expression operators

Operator	Description
{*m*,*n*}	At least *m* and at most *n* repetitions of *regex-term.* Both *m* and *n* must be positive integers, and *m* must be smaller than *n*.
{*m*}	Exactly *m* repetitions of *regex-term*. *m* must be a positive integer.

Table 6-5 Regular expression operators

Operator	Description
{*m*,}	*m* or more repetitions of *regex-term*. *m* must be a positive integer.
*	Zero or more repetitions of *regex-term*. This is equivalent to {0,}.
+	One or more repetitions of *regex-term*. This is equivalent to {1,}.
?	Zero or one repetition of *regex-term*. This is equivalent to {0,1}.
|	One of the two *regex-term* on either side of the vertical bar.
-	Between a starting and ending range, inclusive.
^	Character at the beginning of an AS path regular expression. This character is added implicitly; therefore, the use of it is optional.
$	Character at the end of an AS path regular expression. This character is added implicitly; therefore, the use of it is optional.
()	A group of *regex-term*s that are enclosed in the parentheses. If enclosed in quotation marks with no intervening space (“()”), indicates a null. Intervening space between the parentheses and the *regex-term* is ignored.
[]	Set of characters. One character from the set can match. To specify the start and end of a range, use a hyphen (-).
^	NOT operator.

Configuring Policies

In this section, sample policies are configured for redistributing static and connected routes. This section includes the following examples:

- Example 6-3 Redistributing static routes
- Example 6-4 Redistributing directly connected routes

Redistributing Static Routes

Tip: *It's a good convention to distinguish user-defined names from system-defined names. In this example, the policy statement names are all capitals.*

If you want to redistribute static routes you can also configure a policy to do that. Note that once defined, the policy must be applied to specific routing protocols using the **import** or **export** directive within that protocol.

Example 6-3 creates the policy statement EXPORT_STATIC. This policy directs the routing protocol to redistribute all static routes.

To create a policy for redistributing static routes, perform the following steps in configuration mode:

Example 6-3 Redistributing static routes

Step	Command
Create the policy match conditions. In this rule, static routes are a match.	`root@R1# set policy policy-statement EXPORT_STATIC term 10 from protocol static` `[edit]`
Specify the action for matching rules. In this rule, accept (and redistribute) static routes.	`root@R1# set policy policy-statement EXPORT_STATIC term 10 then action accept` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Redistributing Directly Connected Routes

In this release if you want a routing protocol, such as RIP, to announce connected interfaces (including those with RIP configured) you must define a policy for redistributing connected routes.

Note that once defined, the policy must be applied to specific routing protocols using the **import** or **export** directive within that protocol.

Example 6-3 creates the policy EXPORT_CONN. This policy directs the routing protocol to redistribute all directly connected routes.

To create a policy for redistributing directly connected routes, perform the following steps in configuration mode:

Example 6-4 Redistributing directly connected routes

Step	Command
Create the policy match conditions. In this rule, directly connected routes are a match.	`root@R1# set policy policy-statement EXPORT_CONN term 10 from protocol connected` `[edit]`
Specify the action for matching rules. In this rule, accept (and redistribute) directly connected routes.	`root@R1# set policy policy-statement EXPORT_CONN term 10 then action accept` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Defining a BGP AS Path List

An AS path is a list of the ASs that a routing update traversed to a destination in the Border Gateway Protocol (BGP). The path is represented as a sequence of AS numbers, which are the numbers uniquely identifying BGP autonomous systems. Each AS number represents a network that a packet traverses if it takes the associated route to the destination.

For a packet to reach a destination using this route, it traverses the listed ASs from the leftmost AS number to the rightmost, where the rightmost is the AS immediately preceding its destination, also known as the “origin” AS.

Using policies, match conditions can be defined based on all or portions of the AS path. To do this, you create a named AS path regular expression and then include it in a routing policy.

The AS can be specified in either of the following ways:

- Using the AS number. The AS number as a whole counts as a single term. That is, you cannot reference individual characters within an AS number.
- Using a group of AS numbers enclosed in double quotes. When you group AS numbers, the operator acts on the entire group. The grouped path itself can include operators.

For example, the following AS path list:

Example 6-5 AS path list

```
“3 2 1”
```

matches all AS paths using this exact AS sequence, and no others. It does not match, for instance the AS sequence **“4 3 2 1”**.

You can use wildcards from Table 6-5 to create more powerful AS path list specifications. For example, to match **“***n* **3 2 1”**, where *n* is any AS, you could use the following community list:

Example 6-6 AS path list using “?” wildcard

```
“? 3 2 1”
```

This list matches all of the following:

```
“4 3 2 1”
“100 3 2 1”
“55 3 2 1”
```

To match **“***list* **3 2 1”**, where *list* is any list of AS numbers, you could use the following community list:

Example 6-7 AS path list using “*” wildcard

```
“* 3 2 1”
```

This list matches all of the following:

```
“4 3 2 1”
“100 3 2 1”
“55 3 2 1”
“4 55 3 2 1”
“4 100 55 3 2 1”
```

Defining a BGP Community List

A BGP community is a tag that can be used to mark routes. This tag is carried in BGP update messages, allowing ASs not directly connected to share information about a route.

The community tag categorizes prefixes. Based on the category, the router can make quick forwarding decisions based on the community attribute. There are two types of BGP communities: “well-known” communities (such as “no-export” and “no-advertise”) and “user-defined” or private communities.

A single community identifier is just the AS number of the community, from 0 to 65535. You can specify more than one community as a space-separated list enclosed in double quotes.

The router recognizes the following BGP well-known communities as per RFC 1997:

NO_EXPORT: All routes received carrying a communities attribute containing this value are not advertised outside a BGP confederation boundary (a stand-alone autonomous system that is not part of a confederation should be considered a confederation itself).

NO_ADVERTISE: All routes received carrying a communities attribute containing this value are not advertised to other BGP peers.

NO_SUBCONFED: All routes received carrying a communities attribute containing this value are not advertised to external BGP peers (this includes peers in other members autonomous systems inside a BGP confederation).

Viewing Policy Information

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view policy configuration by using the **show policy** command, as shown in Example 6-8.

To show the information in the **policy** configuration node, perform the following step in configuration mode:

Example 6-8 Viewing the “policy” configuration node

Step	Command

Show the contents of the **policy** configuration node.	`root@R1# show policy`
	`policy-statement "EXPORT_CONN" {`
	`term 1 {`
	`from {`
	`protocol: "connected"`
	`}`
	`then {`
	`action: accept`
	`}`
	`}`
	`}`
	`policy-statement "EXPORT_STATIC" {`
	`term 1 {`
	`from {`
	`protocol: "static"`
	`}`
	`then {`
	`action: accept`
	`}`
	`}`
	`}`

Chapter 7: Static Routes

This chapter describes how to configure static routes on the Vyatta OFR.

The following topics are covered:

- Static Routes Overview
- Configuring Static Routes
- Viewing Static Route Information

Static Routes Overview

A static route is a manually configured route, which in general cannot be updated dynamically from information the router learns about the network topology. However, if a link fails, the router will remove routes, including static routes, from the RIB that used that interface to reach the next hop.

In general, static routes should only be used for very simple network topologies, or to override the behavior of a dynamic routing protocol for a small number of routes.

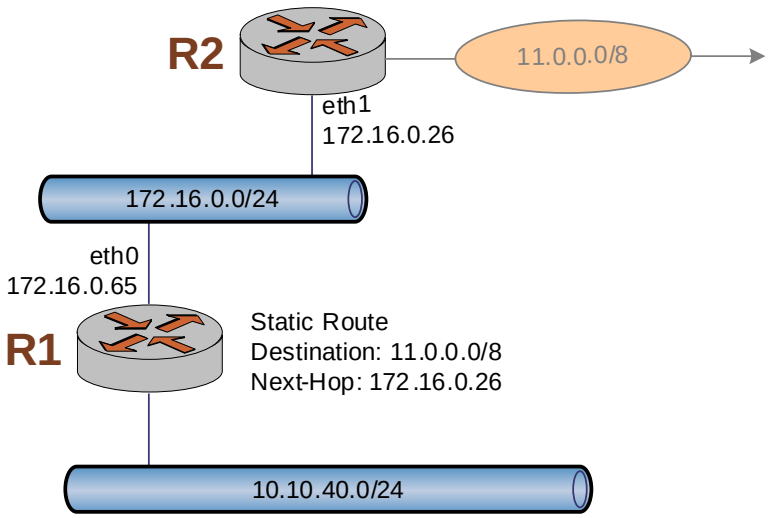
The collection of all routes the router has learned from its configuration or from its dynamic routing protocols is stored in its Routing Information Base (RIB).

Unicast routes are directly used to determine the forwarding table used for unicast packet forwarding.

Configuring Static Routes

In this section, sample configurations are presented for static routes. When you are finished, the router will be configured as shown in Figure 7-1.

Figure 7-1 Static routes



This section includes the following examples:

- Example 7-1 Creating a static route

Example 7-1 creates a static route to network 11.0.0.0/8 directed towards 172.16.0.26.

To create a static route, perform the following steps in configuration mode:

Example 7-1 Creating a static route

Step	Command
Create a mapping between the host name and the IP address.	`root@R1# set protocols static route 11.0.0.0/8 next-hop 172.16.0.26` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Viewing Static Route Information

This section includes the following examples:

- Example 7-2 Showing static routes
- Example 7-3 Viewing the “protocols static” configuration node

Showing Static Routes in the Routing Table

To display route information, use the **show route** command. To show just static routes, use the **show route protocols static** option, as shown in Example 7-2.

Example 7-2 Showing static routes

```
root@vyatta> show route protocol static
Total routes: 1, Total paths: 1
11.0.0.0/8      [static(1)]     > to 172.16.0.26  via eth0
   physical index 3
   ether 00:80:c8:b9:61:0b
```

Showing Static Route Configuration

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view service configuration by using the **show protocols static** command, as shown in Example 7-3.

To show the information in the **protocols static** configuration node, perform the following step in configuration mode:

Example 7-3 Viewing the “protocols static” configuration node

Step	Command
Show the contents of the **protocols static** configuration node.	`root@R1# show protocols static`

Chapter 8: RIP

This chapter describes how to configure the Routing Information Protocol on the Vyatta OFR.

The following topics are covered:

- RIP Overview
- Supported Standards
- Configuring RIP
- Viewing RIP Information

RIP Overview

The Routing Information Protocol (RIP) is the simplest unicast routing protocol in widespread use today. RIP is very simple, both in configuration and protocol design, so it is widely used in simple topologies. However, RIP does not scale well to larger networks, where OSPF or IS-IS are generally more appropriate.

There have been two versions of the RIP protocol.

- RIP version 1 dates back to the early days of the Internet. It is now historic, primarily because it does not support classless addressing, which is necessary in today's Internet. The Vyatta OFR does not support RIPv1.
- RIP version 2 introduces a subnet mask, which allows classless addressing.The Vyatta OFR completely supports RIPv2, as specified in RFC 2453.

RIP is a distance vector protocol, which means that when a router receives a route from a neighbor, that route comes with a distance metric indicating the cost associated with reaching the destination via that neighbor. For RIP, this cost is the number of "hops" (that is, intermediate routers) to the destination.

The router adds its metric for the link on which the route was received to the metric in the received route, and then compares the route against its current best path to that destination. If the new route wins against all other factors, it is installed in the router's routing table. If the route is simply an update of the previous best route, then the stored metric is updated, and the route's deletion timer is restarted. Otherwise the route is ignored. Periodically, the router's routing table is sent to each of its neighbors. Additionally, if a route changes, then the new route is sent to each neighbor.

One reason why RIP is not good for large networks is that in complex topologies it is rather slow to conclude that a route is no longer usable. This is because routers in a loop will learn a route from each other all the way around the loop, and so when a destination becomes unreachable, the routing change will have to propagate around the loop multiple times, increasing the metric each time until the metric reaches infinity, when the route is finally removed. RIP uses a low value of 16 as infinity to reduce the time it takes to remove old information.

A simple case of such a loop is two routers talking to each other. After a destination becomes unreachable, two routers may each believe the other has the best route. *Split horizon* is a scheme for avoiding problems caused by including routes in updates sent to the router from which they were learned. The simple split horizon scheme omits routes learned from one neighbor in updates sent to that neighbor. *Split horizon with poisoned reverse* includes such routes in updates, but sets their metrics to infinity. In general, it is advisable to use split-horizon with poisoned reverse when using RIP, as this significantly speeds convergence in many scenarios.

Supported Standards

The Vyatta implementation of RIP complies with the following standard:

- RFC 2453: RIP version 2.

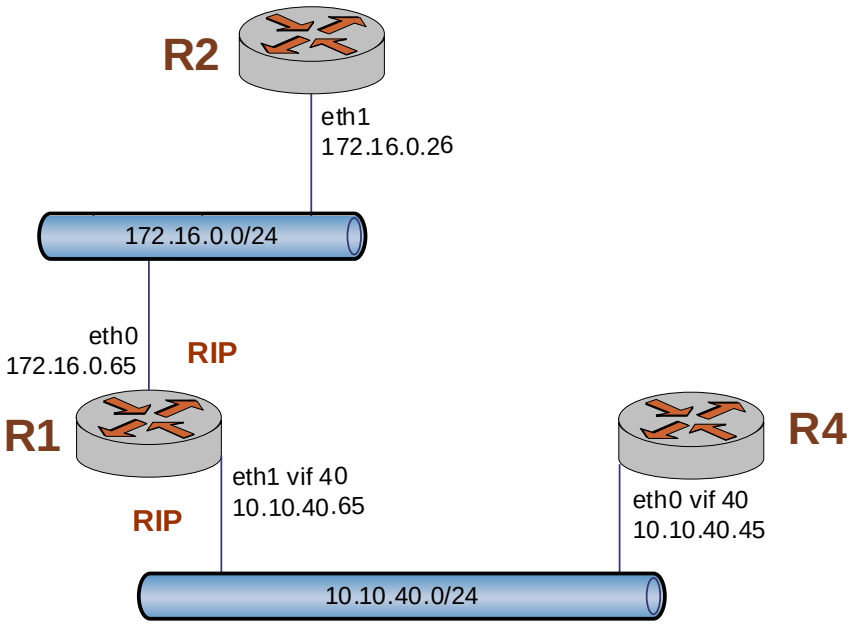
Configuring RIP

To run RIP, you specify the set of interfaces, vifs and addresses on which RIP is to be enabled. Each address to be used by RIP must be explicitly added to the RIP configuration. Typically a metric will also be configured.

In addition, to originate routes via RIP, you must use the **export** parameter to export routes from the routing table via RIP. Also, unlike some other router implementations, on the Vyatta OFR you must export both static routes and connected routes into RIP in order announce these routes.

In this section, sample configurations are presented for RIP. The configuration used is shown in Figure 8-1.

Figure 8-1 RIP



This section includes the following examples:

- Example 8-1 Enabling RIP
- Example 8-2 Redistributing routes in RIP

Basic RIP Configuration

Tip: *Note the notation for referring to vif 40 of interface eth1:* ***eth1.40***.

Example 8-1 enables RIP on 172.16.0.65 on interface eth0 and 10.10.40.65 (on interface eth1.40.

To enable RIP, perform the following steps in configuration mode:

Example 8-1 Enabling RIP

Step	Command
Create the RIP configuration node for 172.16.0.65. This enables RIP on that address on eth0.	`root@R1# set protocols rip interface eth0 address 172.16.0.65` `[edit]`
Create the RIP configuration node for 10.10.40.65. This enables RIP on that address on eth1 vif 40.	`root@R1# set protocols rip interface eth1.40 address 10.10.40.65` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Once RIP is configured, you must define policies to redistribute connected and static routes into RIP, as described in the next section.

Redistributing Static and Connected Routes into RIP

Directly connected routes must be explicitly redistributed by applying a routing policy using the **export** directive within RIP configuration. You can optionally also redistribute static routes.

Example 8-2 applies the policy statements defined in Example 6-3 (“Redistributing static routes,” page 66) and Example 6-4 (“Redistributing directly connected routes,” page 67) as export policies in RIP.

To redistribute routes in RIP, perform the following steps in configuration mode:

Example 8-2 Redistributing routes in RIP

Step	Command
Create the export configuration node, and list the policies you want as export policies. Note there is no space between list items.	`root@R1# set protocols rip export EXPORT_CONN,EXPORT_STATIC` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Viewing RIP Information

This section includes the following examples:

- Example 8-3 Showing RIP routes
- Example 8-4 Showing RIP peer information
- Example 8-5 Viewing the “protocols rip” configuration node

Showing RIP Routes

To display route information, use the **show route** command. To show just RIP routes, use the **show route protocol rip** option, as shown in Example 8-3.

Example 8-3 Showing RIP routes

```
root@vyatta> show route protocol rip
```

Showing RIP Peers

To view information about RIP peers, use the **show rip peer** command in operational mode. Example 8-4 uses the **show rip peer statistics all** option of this command, showing RIP statistics for peer 172.16.0.26 on eth0 172.16.0.65.

Example 8-4 Showing RIP peer information

```
root@R1> show rip peer statistics all
Last Active at Fri Jun  9 12:04:53 2006

  Counter                                 Value
  ------------------------------- ----------------
  Total Packets Received                  25966
  Request Packets Received                  167
  Update Packets Received                 25799
  Bad Packets Received                        0
  Authentication Failures                     0
  Bad Routes Received                         0
  Routes Active                               2
```

Showing RIP Configuration

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view RIP configuration by using the **show protocols rip command**, as shown in Example 8-5.

To show the information in the **protocols rip** configuration node, perform the following step in configuration mode:

Example 8-5 Viewing the “protocols rip” configuration node

Step	Command

Show the contents of the **protocols rip** configuration node.	`root@R1# show protocols rip`
	`  interface eth0 {`
	`  address 172.16.0.65 {`
	`  }`
	`  }`
	`  interface eth1.40 {`
	`  address 10.10.40.65 {`
	`  }`
	`  }`
	`  export: "EXPORT_CONNECTED,EXPORT_STATIC"`

Chapter 9: OSPF

This chapter describes how to configure the OSPF routing protocol on the router.

The following topics are covered:

- OSPF Overview
- Configuring OSPF
- Monitoring OSPF

OSPF Overview

The Open Shortest Path First (OSPF) routing protocol uses a link state algorithm (Dijkstra), as opposed to protocols such as RIP that use a distance vector algorithm. In OSPF, each router advertises the state of its own links, or connections, in a link state advertisement (LSA). It then multicasts the LSA to other routers on the network.

In addition, each router uses the LSAs it receives from other routers to construct a graph that represents the network topology. To build its routing table, the router applies Dijkstra's Shortest Path First algorithm to find the best path through the graph to each network in the topology. This "shortest path tree" becomes the basis for the routes that are elected by OSPF for inclusion in the routing table.

OSPF Areas

OSPF is hierarchical. In OSPF, the network is broken up into "areas." Within each area, routers possess only local routing information. Routing information about other areas is calculated using routes exchanged between areas. This reduces the amount of network topology information routers have to generate and maintain, making OSPF a better choice for larger networks.

No interface can belong to more than one area, and you should take care to avoid assigning overlapping address ranges for different areas. If all the interfaces on a router belong to the same area, the router is said to be an internal router. If the router has OSPF-enabled interfaces that belong to more than one area, it is said to be an Area Border Router (ABR). If a router imports routes from another protocol into OSPF, the router is said to be an Autonomous System Boundary Router (ASBR).

Note that an Area Border Router does not automatically summarize routes between areas. To summarize routes, you must explicitly configure router summarization using the **area-range** command.

An Area Border Router must be connected to the backbone area (0.0.0.0).

Specifying OSPF Areas

OSPF areas are defined by the interfaces that belong to them. Areas are defined in an IPv4 format and interfaces are configured to reside in a particular area.

```
root@R1# set protocols ospf4 area 0.0.0.0 interface eth0 address
172.16.0.65
```

OSPF Area Types

This section discusses four types of OSPF areas:

- Normal Areas
- Stub Areas
- Not-So-Stubby-Areas
- The Backbone Area

Normal Areas

A normal area is an OSPF area that is not “special” in any way. It is not the backbone area, it is not a stub area, and it is not a not-so-stubby area (NSSA).

Stub Areas

A stub area is an OSPF area where no external link-state advertisements (type 5 LSAs) are allowed. A stub area contains an Area Border Router (ABR), but never contains an Autonomous System Boundary Router (ASBR). Therefore, from a stub area, destinations external to the AS can only be reached using the default summary route.

Stub areas are useful when a large part of the topology database consists of AS external advertisements. Routers in a stub area maintain a link state database that represents local links but no external links (type 4 or type 5 LSAs). Instead, external links are summarized into a default route and advertised using a single type 3 LSA generated by the ABR. This greatly reduces the size of the topology database. To replace externally-advertised routes, a stub area defines a default external route, called a summary route, through which traffic can reach external destinations.

You cannot configure an area as being both a stub area and an NSSA.

Not-So-Stubby-Areas

A Not-So-Stubby Area (NSSA) has some characteristics of a stub area, and some of a normal area. NSSAs are similar to stub areas in that they have only default routes to Autonomous System Boundary Routers (ASBRs). However, NSSAs can contain an ASBR (although they need not), while stub areas cannot.

You can configure an NSSA when you want to control the number of external routes in an area, but also allow an ASBR, or where you want OSPF traffic to be able to transit through.

Configuring an area as an NSSA allows a certain limited number of external routes to be redistributed into the network. Normally, ASBRs generate type 5 link-state advertisements (LSAs) for external routes, which advertise destinations external to the OSPF network

being redistributed into OSPF, and which are flooded to the entire autonomous system (AS). NSSAs generate Type 7 LSAs for flooding within the NSSA. These are then translated into Type 5 LSAs by the ABR for flooding into the backbone area.

You cannot configure an area to be both a stub area and an NSSA.

The Backbone Area

If you define more than one area in your OSPF network, one of the areas must be designated as the backbone. The backbone area must be of type “normal”, and it must be area 0 (that is, it must have an area ID of 0.0.0.0).

The backbone area must be contiguous, that is, each area in the OSPF autonomous system must have a direct physical connection to the backbone.

An Area Border Router (ABR) must have at least one interface in the backbone area.

Virtual Links

OSPF requires a single contiguous backbone area (area 0.0.0.0). All areas must connect to the backbone, and all inter-area traffic passes through it. If you cannot design your network to have a single contiguous backbone, or if a failed link splits the backbone, you can “repair” or extend a backbone area by creating a virtual link between two non-contiguous areas.

Wherever possible, configure a contiguous backbone and avoid virtual links, because they add complexity to the network. Also, keep the following in mind when creating virtual links:

- A virtual link can traverse one area.
- Virtual links can be created only between Area Border Routers (ABRs).
- Virtual links cannot traverse stub areas or not-so-stubby areas (NSSAs).

OSPF Costs

The OSPF cost is the link-state metric that you want advertised in the link-state advertisement (LSA) as the cost of sending packets over this interface. The cost of reaching any destination is the sum of the costs of the individual hops. You can only assign one cost per interface. The Vyatta OFR assigns every interface a default cost of 1, which can be modified through configuration.

OSPF Priority

On broadcast interfaces, OSPF uses a router priority value. This priority value is used to determine which routers are selected as the area's Designated Router (DR) and Backup Designated Router (BDR).

The DR and BDR are used to reduce the amount of OSPF traffic on broadcast networks, by reducing the number of adjacent routers to which a router must flood its topology information. In broadcast networks (such as Ethernet), each router establishes an adjacency with only the DR and the BDR, rather than with every router in its area. The DR and the BDR then flood updates from these routers to to all other routers on the network segment.

Priority can range from 0 to 255. In general, the router with the highest priority is elected as the DR, and the router with the second-highest priority is elected as the BDR. The higher the number, the higher the priority. If all routers have an equal priority, the first OSPF router activated on the network becomes the DR and the second router activated becomes the BDR.

Routers with a priority of 0 are ineligible for election.

Route Summaries (Area Ranges)

Area ranges consolidate and summarize internal routes in an area into a single route, for advertisement into an OSPF area border. These are used for route advertisement as type 3 LSAs into other areas of the OSPF domain.

Inter-area route summarization takes place at Area Border Routers (ABRs), and are advertised by ABRs into the backbone area. Therefore, area ranges are configured only on ABRs.

Route summarization helps keep the size of the link state topology database to a minimum, which conserves router memory, minimizes the processing overhead of SPF calculations, reduces the use of network bandwidth by the protocol, and helps speed network convergence.

To summarize the routes in an area into a range, the IP addresses in the area must be assigned so that they are contiguous. This requires careful design of the hierarchical network structure. If there are breaks in the contiguity of IP addresses of an area, you must define more than one route summary for the ranges.

Monitoring the Network

OSPF dynamically discovers the state of the network using LSA updates. It learns about neighbor relationships using hello packets and a router dead interval. The transmit delay introduced by the physical interface itself is also taken into account.

Hello Packets

A hello packet is an OSPF packet used to detect and maintain relationships with neighbors on the same network (directly connected routers). The greater the interval between hello packets, the less router traffic occurs, but the longer it takes for certain topology changes to be detected.

The hello interval must be the same for all routers that are to establish two-way communication within a network. If two routers do not agree on these parameters, they will not consider themselves “neighbors,” and will disregard one another’s communications.

The default hello interval is 10 seconds.

Router Dead Interval

The router dead interval is the maximum time that a router should wait to receiving a hello packet from its neighbor. This value is typically four times the hello interval. If the dead interval passes without the interface receiving a hello packet from the neighbor, the neighbor’s status is changed to out-of-service.

The dead interval must be the same for all routers that are to establish two-way communication within a network. If two routers do not agree on these parameters, they will not consider themselves “neighbors,” and will disregard one another’s communications.

The default router dead interval is 40 seconds.

Interface Transit Delay

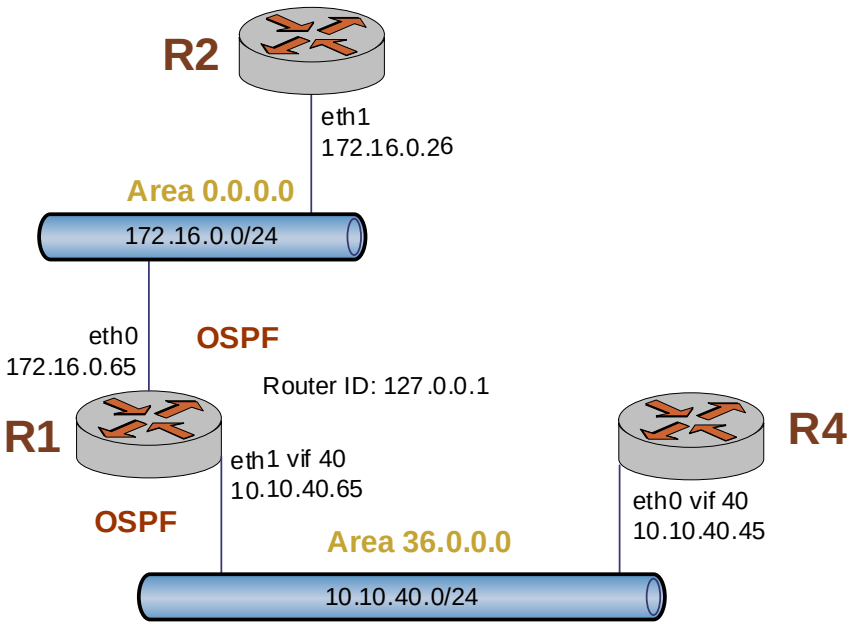
The interface transit delay specifies the estimated time in seconds required to send a link-state advertisement on this interface. The transmit delay is added to the age of the LSA. The LSA age is used to help the network sequence LSAs, so that it can determine which of competing LSAs is the more recent and trustworthy.

LSAs are numbered in sequence, but the sequence numbers are finite, and so cannot be used as the sole determinant of the most recent LSA. Therefore, OSPF also tracks the age of LSAs. Each time the LSA is forwarded to another router, its current age is incremented by the transmit delay. The packet’s age, together with its sequence number, helps the receiving router to determine which version of a received LSA is more recent, and therefore to be used.

Configuring OSPF

In this section, sample configurations are presented for OSPF. When you have finished, the router will be configured as shown in Figure 9-1.

Figure 9-1 OSPF



This section includes the following examples:

- Example 9-1 Basic OSPF Configuration
- Example 9-2 Redistributing routes into OSPF

Basic OSPF Configuration

Example 9-1 establishes one interface in the backbone area and one interface in area 36.0.0.0.

- The router ID is set to the address of the loopback interface. This is considered good practice, as the loopback interface is the most reliable on the router. In this scenario, the loopback address is 127.0.0.1.
- The hello interval for both interfaces is left at the default (10 seconds), as is the dead interval (40 seconds), and the priority (128).

Note the notation for referring to vif 40 of interface eth1: **eth1.40**.

To create a basic OSPF configuration, perform the following steps in configuration mode:

Example 9-1 Basic OSPF Configuration

Step	Command
Create the OSPF configuration node, giving the router ID. This enables OSPF on the router.	`root@R1# set protocols ospf4 router-id 127.0.0.1` `[edit]`
Place 172.16.0.65 on eth0 in the backbone area.	`root@R1# set protocols ospf4 area 0.0.0.0 interface eth0 address 172.16.0.65` `[edit]`
Place 10.10.40.65 on eth1 vif 40 in area 36.0.0.0	`root@R1# set protocols ospf4 area 36.0.0.0 interface eth1.40 address 10.10.40.65` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Redistributing Static and Connected Routes into OSPF

Directly connected routes must be explicitly redistributed by applying a routing policy using the **export** directive within OSPF configuration. You can optionally also redistribute static routes.

Example 9-2 applies the policy statements defined in Example 6-3 (“Redistributing static routes,” page 66) and Example 6-4 (“Redistributing directly connected routes,” page 67) as export policies in OSPF.

To redistribute routes in OSPF, perform the following steps in configuration mode:

Example 9-2 Redistributing routes into OSPF

Step	Command
Create the export configuration node, and list the policies you want as export policies. Note there is no space between list items.	`root@R1# set protocols ospf4 export EXPORT_CONN,EXPORT_STATIC` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Monitoring OSPF

This section includes the following examples:

- Example 9-3 Showing OSPF routes
- Example 9-4 Showing OSPF neighbor information
- Example 9-5 Showing the OSPF LSA database
- Example 9-6 Viewing the “protocols ospf4” configuration node
- Example 9-7 Enabling logging for OSPF

Showing OSPF Routes

To display route information, use the **show route** command, as in Example 9-3.

Example 9-3 Showing OSPF routes

```
root@R1> show route
Total routes: 6, Total paths: 6
0.0.0.0/0       [static(1)]     > to 10.1.0.1           via eth0
10.1.0.0/24     [connected(0)]  > to 10.1.0.50          via eth0
10.10.10.49/32  [ospf(2)]       > to 10.1.0.49          via eth0
24.0.0.0/8      [ospf(1)]       > to 10.1.0.49          via eth0
172.16.0.0/24   [connected(0)]  > to 172.16.0.50        via eth1
192.168.2.0/24  [ospf(2)]       > to 10.1.0.49          via eth0
```

Showing OSPF Neighbors

To view information about OSPF neighbors, use the **show ospf4 neighbor** command in operational mode.

Example 9-4 Showing OSPF neighbor information

```
root@R1> show ospf4 neighbor
Address      Interface State     ID           Pri  Dead
10.10.30.46  eth0      Full      192.168.2.44 3    39
10.1.0.49    eth0      Full      10.10.10.49  1    37
172.16.0.26  eth0      TwoWay    172.16.0.26  128  36
```

Showing the OSPF LSA Database

To view the OSPF Link State Advertisement (LSA) database, use the **show ospf4 database** command in operational mode.

Example 9-5 Showing the OSPF LSA database

```
root@R1> show ospf4 database
OSPF link state database, Area 0.0.0.0
Type      ID           Adv Rtr      Seq        Age Opt  Cksum  Len
Router    *172.16.0.65 172.16.0.65  0x80000002352 0x2  0x6b5836
Network   10.1.0.49    10.10.10.49  0x800002d0359 0x22 0x923440
Router    10.10.30.46  10.10.30.46  0x800002d0510 0x22 0x518f48
Router    172.16.0.26  172.16.0.26  0x80000005485 0x2  0x5e6348
ASExt-2   24.0.0.0     10.1.0.2     0x800001e2839 0x2  0x66d636
```

Showing OSPF Configuration

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view OSPF configuration by using the **show protocols ospf4** command, as shown in Example 9-6.

To show the information in the **protocols ospf4** configuration node, perform the following step in configuration mode:

Example 9-6 Viewing the “protocols ospf4” configuration node

Step	Command

Show the contents of the **protocols ospf4** configuration node.	`root@R1# show protocols ospf4`
	`router-id: 127.0.0.1`
	`area 0.0.0.0 {`
	`interface eth0 {`
	`address 172.16.0.65 {`
	`}`
	`}`
	`interface eth1 {`
	`address 10.10.30.65 {`
	`}`
	`}`
	`}`

Sending OSPF Messages to Syslog

The OSPF process can be configured to produce OSPF-specific log messages, by creating and enabling the **traceoptions** configuration node. When OSPF logging is enabled, the log messages are sent to whatever destinations are configured for syslog.

- By default, log messages are sent to the main log file at **/var/log/messages**.
- You can configure syslog to send all log messages to a file you specify in the **/var/user** directory.

Example 9-7 enables logging for OSPF.

To enable logging for OSPF, perform the following steps in configuration mode:

Example 9-7 Enabling logging for OSPF

Step	Command

Enable logging within the **ospf** configuration node.	`root@R1# set protocols ospf traceoptions flag all disable false`
	`OK`
	`[edit]`

Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

Chapter 10: BGP

This chapter describes how to configure the Border Gateway Protocol on the Vyatta OFR.

The following topics are covered:

- BGP Overview
- Supported Standards
- Configuring BGP
- Monitoring BGP

BGP Overview

Border Gateway Protocol (BGP) is the principal inter-domain routing protocol on the Internet. BGP version 4 is specified in RFC 1771.

The principal concept of BGP is that of the Autonomous System (AS). An AS is a routing domain that is under one administrative authority, and which implements its own routing policies. BGP is used to convey information about AS path topology between ASs.

BGP selects the best path for each prefix from all available paths, which can include paths learned through either internal BGP (iBGP) or external BGP (eBGP). It then installs the best path for the prefix into the IP routing table.

Each BGP route carries with it an AS path, which records the ASs through which the route has passed between the AS where the route was originally advertised and the current AS. When a BGP router passes a route to a router in a neighboring AS, it prepends its own AS number to the AS path.

The AS path is used to prevent routes from looping, and also can be used in policy filters to determine what actions to perform on the prefix, including whether or not to accept the announcement. AS paths are read from right to left, with the right most AS being the origin AS.

BGP best paths are installed in the routing table pursuant to any BGP import policies. Similarly, selected routes may be exported from the routing table to BGP peers, again pursuant to BGP policies.

eBGP and iBGP

BGP is used in two different ways:

- iBGP is used to exchange routing information between routers that are in the same AS. Typically, these routes are originally learned from eBGP. A peering connection between routers configured in the same AS is an iBGP peering.
- eBGP is used to exchange routing information between routers that are in different ASs. A peering connection between routers that are not configured in the same AS is an eBGP peering.

When a route is learned from another router over an eBGP connection, the router first decides if this is the best path to the destination, based on a standard decision process and local policy configuration. If the route is the best path, the route is passed on to all the other BGP routers in the same domain using iBGP connections, as well as on to all the eBGP peers (as allowed by policy).

Scalability of BGP

The Border Gateway Protocol 4 specification (RFC 1771) requires that iBGP speakers be fully meshed. Such a full mesh of configured BGP peerings does not scale well to large domains, so two techniques can be used to improve scaling:

- BGP Confederations (RFC 3065)
- Route Reflection (RFC 2796)

Confederations

Confederations enable you to reduce the size and complexity of the iBGP mesh. In a BGP confederation, a single AS is divided into multiple internal sub-ASs. The sub-ASs are grouped as a confederation, which advertises as a single AS to external peers.

The sub-ASs are only visible within the defined confederation. From outside the confederation, they appear as a single AS. The sub-ASs exchange routing information as if they are iBGP peers.

Sub-ASs use different BGP AS numbers and connections between sub-ASs are eBGP connections. The system is configured to know that the eBGP is another confederation within the same AS, as true eBGP neighbors are identified differently.

Route Reflection

In a fully meshed BGP configuration, any route learned from an iBGP peer is not re-advertised to other iBGP peers. In BGP Route Reflection, iBGP re-advertisement restrictions are relaxed. An iBGP router speaker that is configured as a “Route Reflector” (RR) can, under certain conditions, re-advertise routes learned internally to other internal peers. The re-advertised routes are known as “reflected routes”.

In route reflection, internal peers of an RR are categorized into two types:

- Client peers. The RR and its client peers form a cluster. Within a cluster, client peers need not be fully meshed, but must have an iBGP connection to the RR(s) in the cluster. From the client’s perspective these iBGP connections look the same as any other iBGP connection; that is, the client has no awareness that it is in a Route Reflector cluster.
- Non-client peers. Non-client peers, including the RR, must be fully meshed.

When an RR receives a route from an iBGP peer, it selects the best path based on its path selection rule. After the best path is selected, the RR chooses its action depending on the type of the peer from which it learned the best path.

- If the route was learned from a client peer, the RR reflects the route to both client and non-client peers. All iBGP updates from client peers are reflected to all other client peers in the cluster. This is done regardless of whether the update was the best path for the RR itself.

- If the route was learned from a non-client peer, it is reflected out to all client peers.

To prevent looping, clients must not peer with RRs outside of the cluster.

Route Flapping and Flap Damping

Route flap is a situation where a route fluctuates repeatedly between being announced, then withdrawn, then announced, then withdrawn, and so on. In this situation, a BGP system will send an excessive number of update messages advertising network reachability information.

Route dampening minimizes the propagation of update messages between BGP peers for flapping routes. This reduces the load on these devices without unduly impacting the route convergence time for stable routes.

When route damping is enabled, a route is assigned a penalty each time it "flaps." If the penalty exceeds a configured threshold (its *suppress* value) the route is suppressed.

After the route has been stable for a configured interval (its *half-life*) the penalty is reduced by half. Subsequently, the penalty is reduced every 5 seconds. When the penalty falls below a configured value (its *reuse* value), the route is unsuppressed.

The penalty applied to a route will never exceed the *maximum penalty*, which is computed from configured attributes as follows:

```
Maximum penalty = reuse * 2^(suppress/half-life)
```

AS Paths

An AS path is a path to a destination in the Border Gateway Protocol (BGP). The path is represented as a sequence of AS numbers, which are the numbers uniquely identifying BGP autonomous systems. Each AS number represents an autonomous system (which may be comprised of multiple networks) that a packet traverses if it takes the associated route to the destination.

For a packet to reach a destination using this route, it traverses the listed ASs from the leftmost AS number to the rightmost, where the rightmost is the AS immediately preceding its destination.

Using policies, match conditions can be defined based on all or portions of the AS path. To do this, you can either specify the AS path directly in a policy command using a regular expression in the **as-path** attribute, or create a named AS path regular expression using the **as-path-list** attribute and including the name in a policy command.

For information on specifying AS paths using regular expressions, please see the section "Defining a BGP AS Path List" in "Appendix 6: Routing Policies."

BGP Communities

A BGP community is a tag that can be used to mark routes. This tag is carried in BGP update messages, allowing ASs not directly connected to share information about a route.

The community tag categorizes prefixes. Based on the category, the router can make quick forwarding decisions based on the community attribute. There are two types of BGP communities: “well-known” communities (such as “no-export” and “no-advertise”) and “user-defined” or private communities.

A single community identifier is just the AS number of the community, from 0 to 65535. You can specify more than one community as a space-separated list enclosed in double quotes.

The router recognizes the following BGP well-known communities as per RFC 1997:

NO_EXPORT: All routes received carrying a communities attribute containing this value are not advertised outside a BGP confederation boundary (a stand-alone autonomous system that is not part of a confederation should be considered a confederation itself).

NO_ADVERTISE: All routes received carrying a communities attribute containing this value are not advertised to other BGP peers.

NO_SUBCONFED: All routes received carrying a communities attribute containing this value are not advertised to external BGP peers (this includes peers in other members autonomous systems inside a BGP confederation).

Supported Standards

The Vyatta implementation of BGP fully complies with the following standards:

- draft-ietf-idr-bgp4-22.txt: BGP-4 Specification (obsoletes RFC 1771)
- RFC 1657: Definitions of Managed Objects for the Fourth Version of the Border Gateway Protocol (BGP- 4) using SMIv2.

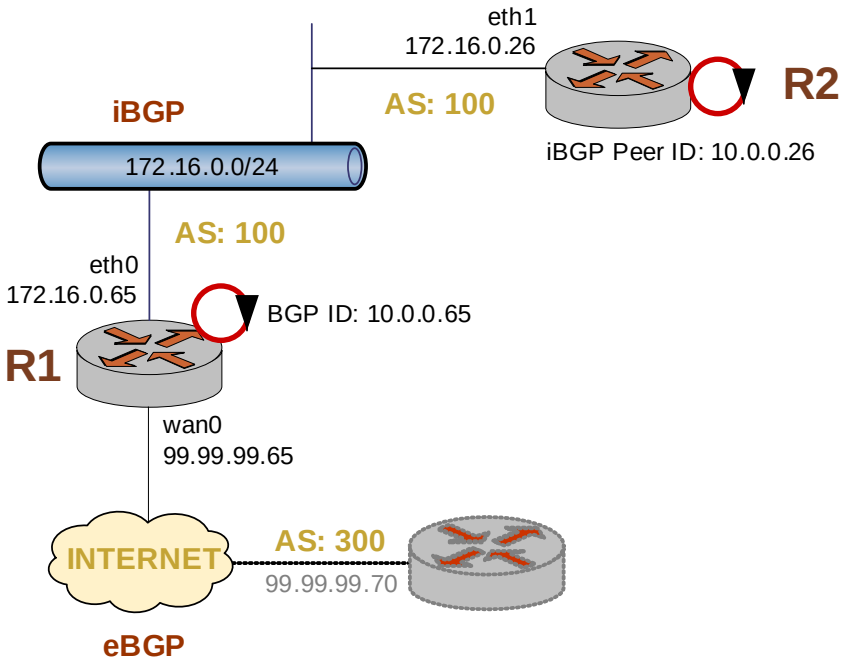
The Vyatta implementation of BGP also provides limited support for the following:

- BGP MIB
- RFC 1997: BGP Communities Attribute.
- RFC 3065: BGP Confederations RFC 3065
- RFC 2796: Route Reflection RFC 2796

Configuring BGP

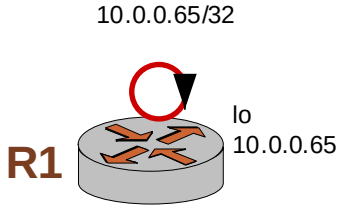
In this section, sample configurations are presented for BGP. When you have finished, the router will be configured as shown in Figure 10-1.

Figure 10-1 BGP



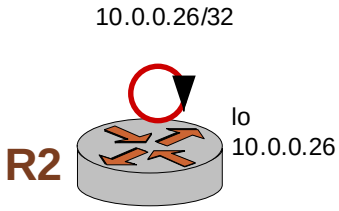
In this set of examples, you should assume that the loopback interface on router R1 has been configured with IP address 10.0.0.65, as shown in Figure 10-2. (The loopback address was configured on router R1 in “Chapter 2: Ethernet and VLAN Interfaces”. See “Configuring the Loopback Interface” on page 20 for this information.)

Figure 10-2 The loopback interface on router R1



In addition, you should assume that the loopback interface on router R2 has been configured with IP address 10.0.0.26, as shown in Figure 10-3. (This configuration was not shown in an example.)

Figure 10-3 The loopback interface on router R2



This section includes the following examples:

- Example 10-1 Enabling BGP
- Example 10-2 Configuring an iBGP peer
- Example 10-3 Configuring an eBGP peer

Enabling BGP

To enable BGP on the router, you must specify the router’s BGP ID and the autonomous system in which the router will reside.

The BGP ID is normally given the loopback address of the router. Note that the BGP ID does not actually provide any reachability information, but just gives the BGP speaker a unique identifier.

Even so, it is typically set to one of the router's IP addresses, and it is normally required that this be globally unique. It is considered good practice to set the BGP router ID to the address of the loopback interface, as this is the most reliable interface on the router.

Example 10-1 enables BGP on router R1. In this example:

- Router R1 is given the router ID 10.10.10.65 in AS 100, because 10.10.10.65 is the IP address of the loopback interface.
- This router resides within AS 100.

To enable BGP on router R1, perform the following steps in configuration mode:

Example 10-1 Enabling BGP

Step	Command
Set the BGP ID to the loopback address. This also creates the BGP configuration node, enabling BGP on the router.	`root@R1# set protocols bgp bgp-id 10.10.10.65` `[edit]`
Specify the local AS.	`root@R1# set protocols bgp local-as 100` `[edit]`

Commit and view the configuration.	`root@R1# commit`
	`OK`
	`[edit]`
	`root@R1# show protocols bgp`
	`  bgp-id: 10.0.0.65`
	`  local-as: 100`
	
	`[edit]`
	`root@R1#`

Configuring an iBGP Peer

For iBGP peerings, the peer identifier is normally the IP address bound to that router's loopback interface. An iBGP session is usually contained within a local LAN, with multiple redundant physical links between the iBGP devices. For iBGP routes, reachability is all that is necessary, and the loopback interface is reachable so long as at least one physical interface is operational. For this situation, therefore, peering on the loopback interface works well.

Since BGP does not provide reachability information, you must sure that each iBGP peer knows how to reach other peers. To be able to reach one another, each peer must have some sort of Interior Gateway Protocol (IGP) route, such as a connected route, a static route, or a route through a dynamic routing protocol such as RIP or OSPF, which tells them how to reach the opposite router.

Note that, in the sample scenario, BGP will not be able to determine the path from the loopback address of R1 to the loopback address of R2, unless an IGP route is added. For this reason, we add a static route from R1 to R2 within the example.

Example 10-2 creates router R2 as an iBGP peer. In this example:

- IP address 10.0.0.26 is used as R2's peer ID, since that is the IP address of the loopback interface on R2.
- For maximum reliability, R1's loopback interface is used as the local IP and next-hop for this iBGP peer.
- A static route is defined between R1 and the loopback interface on R2. Although R2's interface 172.16.0.26 is on a directly connected network, its loopback interface is not, so a route must be provided to tell R1 how to reach R2's loopback interface. Alternatively, you could create a RIP or OSPF route between R1 and the loopback interface on R2. (Note that if you were configuring R2, you would likewise have to supply a route from R2 to R1's loopback interface.)
- Router R1 and router R2 reside on the same LAN, in AS 100. Therefore, router R2 is an iBGP peer (rather than an eBGP peer) of router R1.
- IP address 172.16.0.65 is designated as the egress address to use to reach router R2, and also as the next-hop to be advertised to the peer.

To create an iBGP peer, perform the following steps in configuration mode:

Example 10-2 Configuring an iBGP peer

Step	Command
Create the iBGP peer, peering on its loopback interface at 10.0.026. The peer is an iBGP peer because it resides within the same AS as the router.	`root@R1# set protocols bgp peer 10.0.0.26 as 100` `[edit]`
Specify the egress IP address for traffic destined for this peer.	`root@R1# set protocols bgp peer 10.0.0.26 local-ip 10.0.0.65` `[edit]`
Specify the next hop for this peer.	`root@R1# set protocols bgp peer 10.0.0.26 next-hop 10.0.0.65` `[edit]`
Create a route to the loopback interface of R2 through the R2 interface on the 172.16.0.0/24 network.	`root@R1# set protocols static route 10.0.0.26/32 next-hop 172.16.0.26` `[edit]`

Example 10-2 Configuring an iBGP peer

Commit and view the configuration.	root@R1# **commit**
	OK
	[edit]
	root@R1# **show protocols bgp**
	bgp-id: 10.0.0.65
	local-as: 100
	peer "10.0.0.26" {
	local-ip: 10.0.0.65
	as: 100
	next-hop: 10.0.0.65
	}
	
	[edit]
	root@R1# **show protocols static**
	route 10.0.0.26/32 {
	next-hop: 172.16.0.26
	}
	
	[edit]

Configuring an eBGP Peer

eBGP usually takes place over WAN links, where there is a single physical path between eBGP peers. Redundancy is accomplished by using multiple peers over different WAN links with distinct BGP sessions so that, if one session fails, another can take over. For eBGP peerings, therefore, the peer identifier is normally the IP address of the peer router on the physical interface over which BGP traffic is to be exchanged.

Example 10-3 creates an eBGP peering to a remote router across the WAN. In this example:

- An IP address on the actual physical interface of the eBGP peer is used—in this case 99.99.99.70. Then, if the eBGP link fails, the session will correctly be dropped.
- The peer is an eBGP peer because it is located within a different AS (AS 300) from router R1.
- IP address 99.99.99.65 is designated as the egress address to use to reach the eBGP peer, and also as the next-hop to be advertised to the peer.

To configure an eBGP peer, perform the following steps in configuration mode:

Example 10-3 Configuring an eBGP peer

Step	Command
Create the eBGP peer, peering on the physical interface at 99.99.99.70. The peer is an eBGP peer because it resides within a different AS from this router.	`root@R1# set protocols bgp peer 99.99.99.70 as 300` `[edit]`
Specify the egress address for traffic destined for this peer.	`root@R1# set protocols bgp peer 99.99.99.70 local-ip 99.99.99.65` `[edit]`
Specify the next hop for this peer.	`root@R1# set protocols bgp peer 99.99.99.70 next-hop 99.99.99.65` `[edit]`

Commit and view the configuration.	`root@R1# commit`
	`OK`
	`[edit]`
	`root@R1# show protocols bgp`
	`  bgp-id: 10.0.0.65`
	`  local-as: 100`
	`peer "10.0.0.26" {`
	`  local-ip: 10.0.0.65`
	`  as: 100`
	`  next-hop: 10.0.0.65`
	`  }`
	`  peer "99.99.99.70" {`
	`  local-ip: 99.99.99.65`
	`  as: 300`
	`  next-hop: 99.99.99.65`
	`  }`
	
	`[edit]`
	`root@R1#`

Monitoring BGP

This section presents the following topics:

- BGP Operational Commands
- Showing BGP Routes
- Showing BGP Peers
- Sending BGP Messages to Syslog

BGP Operational Commands

You can use the following operational commands to monitor BGP. Please see the *Vyatta OFR Command Reference* for details on these commands and their options.

Command	Description
`show bgp peers`	Displays information about BGP peerings.
`show bgp routes`	Displays BGP route information.
`show route bgp`	Displays information about BGP routes stored in the routing table.
`show route ebgp`	Displays information about eBGP routes stored in the routing table.
`show route ibgp`	Displays information about iBGP routes stored in the routing table.

This section includes the following examples:

- Example 10-4 Showing iBGP routes
- Example 10-5 Showing eBGP routes
- Example 10-6 Showing all BGP routes
- Example 10-7 Showing BGP peer information

NOTE *The output in these examples shows information unrelated to the sample configurations.*

Showing BGP Routes

To display route information, use the **show route** command. To show just iBGP routes, use the **show route protocol ibgp** option, as shown in Example 10-4.

Example 10-4 Showing iBGP routes

```
root@R1> show route protocol ibgp
Total routes: 43534, Total paths: 43534
3.0.0.0/8    [ibgp(0)]       > to 192.168.1.26      via eth0
4.0.0.0/8    [ibgp(0)]       > to 192.168.1.26      via eth0
4.0.0.0/9    [ibgp(0)]       > to 192.168.1.26      via eth0
4.17.250.0/24[ibgp(0)]       > to 192.168.1.26      via eth0
```

To show just eBGP routes, use the **show route protocol ebgp** option, as shown in Example 10-5.

Example 10-5 Showing eBGP routes

```
root@R1> show route protocol ebgp
Total routes: 43534, Total paths: 43534

4.21.206.0/24   [ebgp(0)]       > to 192.168.1.26      via eth0
4.23.84.0/22    [ebgp(0)]       > to 192.168.1.26      via eth0
4.23.112.0/24   [ebgp(0)]       > to 192.168.1.26      via eth0
4.23.113.0/24   [ebgp(0)]       > to 192.168.1.26      via eth0
4.23.114.0/24   [ebgp(0)]       > to 192.168.1.26      via eth0
4.36.100.0/23   [ebgp(0)]       > to 192.168.1.26      via eth0
:
:
```

To show all BGP routes, use the **show route protocol bgp** option, as shown in Example 10-6.

Example 10-6 Showing all BGP routes

```
root@R1> show route protocol bgp
Total routes: 150704, Total paths: 150704

6.3.0.0/18   [ebgp(0)]       > to 192.168.1.26      via eth0
6.4.0.0/16   [ebgp(0)]       > to 192.168.1.26      via eth0
6.5.0.0/19   [ebgp(0)]       > to 192.168.1.26      via eth0
8.0.0.0/9    [ebgp(0)]       > to 192.168.1.26      via eth0
8.2.64.0/23  [ebgp(0)]       > to 192.168.1.26      via eth0
8.2.144.0/22 [ebgp(0)]       > to 192.168.1.26      via eth0
:
:
```

Showing BGP Peers

To view information about BGP peers, use the **show bgp peers** command in operational mode. Example 10-7 uses the **show bgp peers detail** option of this command.

Example 10-7 Showing BGP peer information

```
root@R1> show bgp peers detail
Peer 1: local 172.16.0.65/179 remote 172.16.0.26/179
  Peer ID: none
  Peer State: ACTIVE
  Admin State: START
  Negotiated BGP Version: n/a
  Peer AS Number: 1
  Updates Received: 0,  Updates Sent: 0
  Messages Received: 0,  Messages Sent: 0
  Time since last received update: n/a
  Number of transitions to ESTABLISHED: 3
  Time since last in ESTABLISHED state: 112 seconds
  Retry Interval: 120 seconds
  Hold Time: n/a,  Keep Alive Time: n/a
  Configured Hold Time: 90 seconds,  Configured Keep Alive
   Time: 30 seconds
  Minimum AS Origination Interval: 0 seconds
  Minimum Route Advertisement Interval: 0 seconds

Peer 2: local 10.10.30.65/179 remote 10.10.30.46/179
  Peer ID: none
  Peer State: ACTIVE
  Admin State: START
  Negotiated BGP Version: n/a
  Peer AS Number: 5
  Updates Received: 0,  Updates Sent: 0
  Messages Received: 0,  Messages Sent: 0
  Time since last received update: n/a
  Number of transitions to ESTABLISHED: 0
  Time since last in ESTABLISHED state: n/a
  Retry Interval: 120 seconds
  Hold Time: n/a,  Keep Alive Time: n/a
  Configured Hold Time: 90 seconds,  Configured Keep Alive
   Time: 30 seconds
  Minimum AS Origination Interval: 0 seconds
  Minimum Route Advertisement Interval: 0 seconds
```

Sending BGP Messages to Syslog

The BGP process generates log messages during operation. You can configure the system to send BGP-specific log messages to syslog, by creating and enabling the **traceoptions** configuration node. The result will depend on how the system syslog is configured.

Keep in mind that in the current implementation, the main syslog file **/var/log/messages** reports only messages of severity **warning** and above, regardless of the severity level configured. If you want to configure a different level of severity for log messages (for example, if you want to see debug messages during troubleshooting), you must configure syslog to send messages into a different file, which you define within syslog.

Example 10-8 enables logging for BGP.

To enable logging for BGP, perform the following steps in configuration mode:

Example 10-8 Enabling logging for BGP

Step	Command

Enable logging within the **bgp** configuration node.	`root@R1# set protocols bgp traceoptions flag all disable false`
	`OK`
	`[edit]`

Commit and view the change.	`root@R1# commit`
	`OK`
	`[edit]`
	`root@R1# show -all protocols bgp traceoptions`
	`  flag {`
	`  all {`
	`  disable: false`
	`  }`
	`  }`
	
	`[edit]`

Example 10-8 disables logging for BGP.

To disable logging for BGP without discarding logging configuration, perform the following steps in configuration mode:

Example 10-9 Disabling logging for BGP

Step	Command

Disable logging within the **bgp** configuration node.	`root@R1# set protocols bgp traceoptions flag all disable true`
	`OK`
	`[edit]`

Commit and view the change.	`root@R1# commit`
	`OK`
	`[edit]`
	`root@R1# show -all protocols bgp traceoptions`
	`flag {`
	`all {`
	`disable: true`
	`}`
	`}`
	
	`[edit]`

Chapter 11: VRRP

This chapter describes how to configure the Virtual Router Redundancy Protocol on the Vyatta OFR.

The following topics are covered:

- VRRP Overview
- Configuring VRRP
- Viewing VRRP Information

VRRP Overview

Virtual Router Redundancy Protocol (VRRP) is a protocol for allowing a cluster of routers to act as one virtual router. VRRP, as specified by RFC 2338 and RFC 3678, was designed to provide router failover services in the event of an interface failure.

Routers in a VRRP cluster share a virtual IP address (the VIP) and a virtual MAC address. This provides alternate paths through the network for hosts without explicitly configuring them, and creates redundancy that eliminates any individual router as a single point of failure in the network. This is particularly important for statically configured default routers, the failure of which can otherwise be a catastrophic event on a network.

In VRRP, the IP addresses of interfaces on different real routers are mapped onto a “virtual router”. The virtual router is an abstract object, managed by the VRRP process, that is defined by its virtual router ID (the group identifier of the set of routers forming the virtual router) plus the virtual IP address (the “virtual address”, or VIP) presented to the network. Hosts on the network are configured to direct packets to the VIP, rather than to the IP addresses of the real interfaces.

The virtual router uses the group identifier to construct a virtual MAC address from a standard MAC prefix (specified in the VRRP standard) plus the group identifier. ARP requests for the VIP are resolved to the virtual MAC address, which “floats” from real router to real router, depending on which is acting as the master router of the virtual router. If the master router fails, the backup router is brought into service using the virtual MAC address and VIP of the virtual router. In this way, service can continue around a failed gateway transparently to hosts on the LAN.

VRRP dynamically elects the router that is to be the master. In most cases, this is simply the router with the interface having highest configured priority. If two interfaces have identical priorities, the router with the one having the highest IP address is elected master.

The master router forwards packets for local hosts and responds to ARP requests, ICMP pings, and IP datagrams directed to the VIP. Backup routers remain idle, even if healthy. ARP requests, pings, and datagrams made to the real IP addresses of interfaces are responded to by the interface in the normal way.

The master router also assumes responsibility for the virtual router, continuously forwarding advertisement packets (MAC-level multicast packets) as a “heartbeat” through TCP port 112 (the IANA well-known port for VRRP) to the backup routers. If the heartbeat stops for a configured period (the “dead interval”), VRRP assumes that the master router has become unavailable, and it elects one of the backup routers to become the new master router.

On the Vyatta OFR, VRRP can be run on either a standard Ethernet interface, or it can be run on the vif of an Ethernet interface (that is, a VLAN interface).

Configuring VRRP

This sequence sets up a basic VRRP configuration between two Vyatta routers.

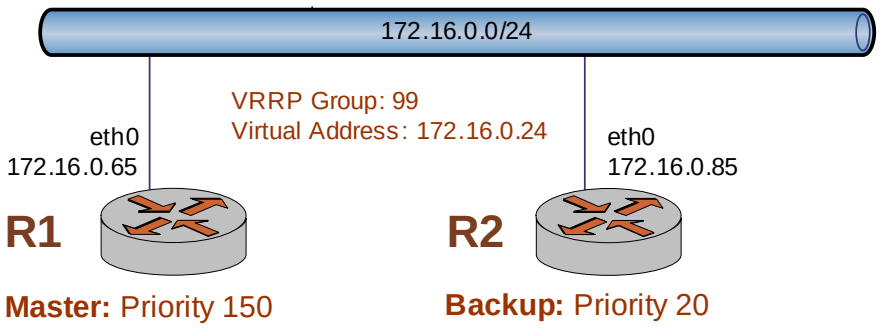
Remember that in VRRP:

- The router configured with the highest priority will initially be elected the master router. If more than one router has the highest priority, then the first active router will be elected the master router.
- Enabling preemption will allow a higher-priority neighbor to preempt the current master and become master itself.

The implementation is currently restricted to one VRRP group per interface, regardless of whether the group is defined at the physical interface level or the vif level.

In this section, sample configurations are presented for VRRP. When you have finished, the router will be configured as shown in Figure 11-1.

Figure 11-1 VRRP



This section includes the following examples:

- Example 11-1 Configuring a first router for VRRP
- Example 11-2 Configuring a backup router for VRRP

Configuring the First Router

Example 11-1 enables VRRP on eth0 of the first router (R1) and assigns it to VRRP group 99. The virtual address is 172.16.0.24. Preemption is enabled, and R1 is assigned a priority of 150.

To configure the first router for VRRP, perform the following steps in configuration mode:

Example 11-1 Configuring a first router for VRRP

Step	Command
Create the VRRP configuration node for eth0 on R1. This enables VRRP on that interface. Assign the VRRP group.	`root@R1# set interfaces ethernet eth0 vrrp vrrp-group 99` `[edit]`
Specify the virtual address of the VRRP group.	`root@R1# set interfaces ethernet eth0 vrrp virtual-address 172.16.0.24` `[edit]`
Enable preemption.	`root@R1# set interfaces ethernet eth0 vrrp preempt true` `[edit]`
Set the priority of this router to 150.	`root@R1# set interfaces ethernet eth0 vrrp priority 150` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Configuring the Second Router

Example 11-2 enables VRRP on eth0 of the second router (R1Backup), and assigns it to VRRP group 99. The virtual address is the same as that for R1: 172.16.0.24. Preemption is enabled, and R1Backup is assigned a priority of 20. This is lower than the priority of R1, so R1 will be the master and R2 will be the backup under ordinary circumstances.

To configure the second router for VRRP, perform the following steps in configuration mode:

Example 11-2 Configuring a backup router for VRRP

Step	Command
Create the VRRP configuration node for eth0 of R2. This enables VRRP on that interface. Assign the VRRP group.	`root@R2# set interfaces ethernet eth0 vrrp vrrp-group 99` `[edit]`
Specify the virtual address of the VRRP group.	`root@R2# set interfaces ethernet eth0 vrrp virtual-address 172.160.0.24` `[edit]`
Enable preemption.	`root@R2# set interfaces ethernet eth0 vrrp preempt true` `[edit]`
Set the priority of this router to 20. This is a lower priority than that set for R1, so R1 will become the master.	`root@R2# set interfaces ethernet eth0 vrrp priority 20` `[edit]`

Commit the configuration.	`root@R2# commit`
	`OK`
	`[edit]`

Viewing VRRP Information

This section includes the following examples:

- Example 11-3 Showing VRRP group information
- Example 11-4 Viewing the “vrrp” configuration node for an interface

Showing VRRP Configuration

To view information about how VRRP groups are configured on the router, use the **show vrrp** command in operational mode. Example 11-3 shows VRRP configuration for R1.

Example 11-3 Showing VRRP group information

```
root@R1> show vrrp
Physical interface: eth0, Address: 172.16.0.24
Interface state: up, Group: 99, State: master
Priority: 150, Advertisement interval: 1s, Authentication
   type: none
  Preempt: yes, VIP count: 1, VIP: 172.16.0.24
  Advertisement timer: 429s, Master router: 172.16.0.65
  Virtual MAC: 00:00:5E:00:01:63
```

Showing the VRRP Configuration Node

You can always view the information in configuration nodes by using the **show** command in configuration mode. VRRP is configured as a property of an interface or a vif, so in this case, you can view VRRP configuration by using the **show interfaces ethernet** *int-name* **vrrp** command, or the **show interfaces ethernet** *int-name***.***vif-name* **vrrp** command, as required. Example 11-4 shows the command you would enter to view the VRRP configuration node for interface eth0 of router R1.

To show VRRP information for an interface, perform the following step in configuration mode:

Example 11-4 Viewing the “vrrp” configuration node for an interface

Step	Command

Show the contents of the **vrrp** configuration node for eth0.	`root@R1# show interfaces ethernet eth0 vrrp`
	`  vrrp-group: 99`
	`  virtual-address: 172.16.0.24`
	`  priority: 150`
	`  preempt: true`

Chapter 12: NAT

This chapter describes how to configure NAT on the Vyatta OFR.

The following topics are covered:

- NAT Overview
- Configuring NAT
- Viewing NAT Information

NAT Overview

Network Address Translation (NAT) is a process whereby internal addresses on a private network are mapped, or “translated” to a globally unique public IP address, or to another private address, by an intermediate device or application, such as a router, firewall, or VPN. NAT was originally designed to help conserve the number of IP addresses used by the growing number of devices accessing the Internet, but it also has important applications in network security.

The computers on the internal network can use any of the addresses set aside by the Internet Assigned Numbers Authority (IANA) for private addressing (see also RFC 1918). These are the following:

- 10.0.0.0 to 10.255.255.255 (CIDR: 10.0.0.0/8)
- 172.16.0.0 to 172.31.255.255 (CIDR: 172.16.0.0/12)
- 192.168.0.0 to 192.168.255.255 (CIDR: 192.268.0.0/16)

These reserved IP addresses are not in use on the Internet, so an external machine cannot directly route to them.

A NAT-enabled router hides the IP addresses of your internal network from the external network, by replacing the internal, private IP addresses with the public IP addresses that have been provided to it. These public IP addresses are the only addresses that are ever exposed to the external network.

The router can manage multiple public IP addresses, from which it can dynamically choose when performing address replacement (“dynamic NAT”). You should be aware that, although NAT can minimize the possibility that your internal computers make unsafe connections to the external network, it provides no protection to a computer that, for one reason or another, connects to an untrusted machine.

Therefore, you should always combine NAT with packet filtering and other features of a complete security policy to fully protect your network.

Configuring NAT

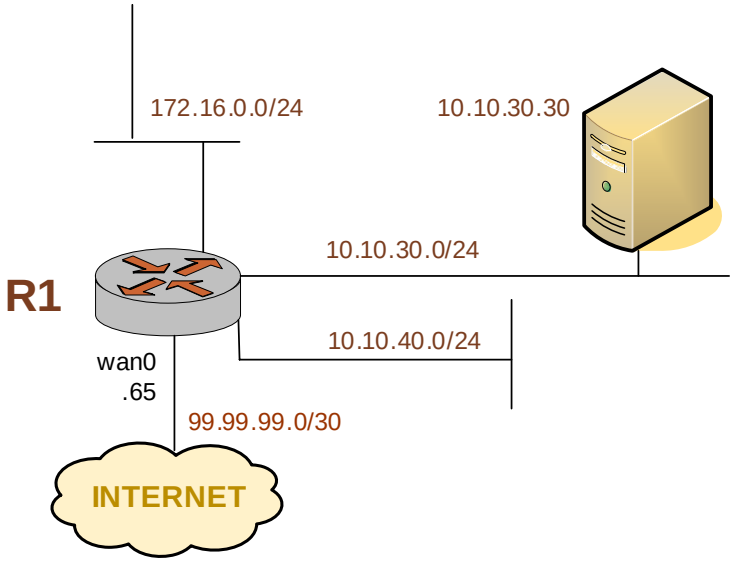
This sequence sets up a basic NAT configuration on router R1.

The first three rules of this configuration allow outbound traffic from three private subnets: 172.16.0.0/24, 10.10.30.0/24, and 10.10.40.0/24. Traffic from these subnets exits to the Internet through interface wan0 on R1.

The last rule of the configuration allows inbound SSH traffic to pass through wan0 on R1 to the target internal host for SSH, which is 10.10.30.30.

The subnets involved are shown in Figure 12-1.

Figure 12-1 Basic NAT configuration



First, create the three rules that allow traffic from selected internal networks to exit to the Internet through interface wan0 on R1. To create these rules, perform the following steps in configuration mode:

Example 12-1 Allowing outbound traffic from internal subnets

Step	Command
Create Rule 1, which allows outbound traffic from network 172.16.0.0/24.	root@R1# **create service nat rule 1** [edit] root@R1# **edit service nat rule 1** [edit service nat rule 1] root@R1# **set type source** [edit service nat rule 1] root@R1# **set translation-type masquerade** [edit service nat rule 1] root@R1# **set outbound-interface wan0** [edit service nat rule 1] root@R1# **set protocols all** [edit service nat rule 1] root@R1# **set source network 172.16.0.0/24** [edit service nat rule 1] root@R1# **set destination network 0.0.0.0/0** [edit service nat rule 1] root@R1# **top** [edit] root@R1#
Create Rule 2, which allows outbound traffic from network 10.10.30.0/24.	root@R1# **create service nat rule 2** [edit] root@R1# **edit service nat rule 2** [edit service nat rule 2] root@R1# **set type source** [edit service nat rule 2] root@R1# **set translation-type masquerade** [edit service nat rule 2] root@R1# **set outbound-interface wan0** [edit service nat rule 2] root@R1# **set protocols all** [edit service nat rule 2] root@R1# **set source network 10.10.30.0/24** [edit service nat rule 2] root@R1# **set destination network 0.0.0.0/0** [edit service nat rule 2] root@R1# **top** [edit] root@R1#

Example 12-1 Allowing outbound traffic from internal subnets

Create Rule 3, which allows outbound traffic from network 10.10.40.0/24.	root@R1# **create service nat rule 3** [edit] root@R1# **edit service nat rule 3** [edit service nat rule 3] root@R1# **set type source** [edit service nat rule 3] root@R1# **set translation-type masquerade** [edit service nat rule 3] root@R1# **set outbound-interface wan0** [edit service nat rule 3] root@R1# **set protocols all** [edit service nat rule 3] root@R1# **set source network 10.10.40.0/24** [edit service nat rule 3] root@R1# **set destination network 0.0.0.0/0** [edit service nat rule 3] root@R1# **top** [edit] root@R1#
Commit the configuration.	root@R1# **commit** OK [edit]

Now, create the rule that accepts SSH traffic directed at the public IP address of wan0 on R1 (99.99.99.65) to pass through to a single internal host at the private IP address 10.10.30.30.

Note that, in effect, this configuration “exports” the private server “outside” the protected network. This means that you will not be able to access the router from outside using SSH. That is, trying to access address 99.99.99.65 will now access the SSH server rather than the Vyatta OFR.

To create this rule, perform the following steps in configuration mode:

Example 12-2 Allowing inbound SSH traffic

Step	Command
Create Rule 4, which allows inbound SSH traffic destined for the public IP address.	root@R1# **create service nat rule 4** [edit] root@R1# **edit service nat rule 4** [edit service nat rule 4] root@R1# **set type destination** [edit service nat rule 4] root@R1# **set translation-type static** [edit service nat rule 4] root@R1# **set inbound-interface wan0** [edit service nat rule 4] root@R1# **set protocols tcp** [edit service nat rule 4] root@R1# **set source network 0.0.0.0/0** [edit service nat rule 4] root@R1# **set destination address 99.99.99.65** [edit service nat rule 4] root@R1# **set destination port-name ssh** [edit service nat rule 4] root@R1# **set inside-address address 10.10.30.30** [edit service nat rule 4] root@R1# **top** [edit service nat rule 4] root@R1#
Commit the configuration.	root@R1# **commit** OK [edit]

Viewing NAT Information

This section includes the following examples:

- Example 12-3 Showing NAT rules
- Example 12-4 Viewing the “service nat” configuration node

Showing NAT Rules

To display NAT rules, use the **show nat rules** command in operational mode, as shown in Example 12-3.

Example 12-3 Showing NAT rules

```
root@R1> show nat rules
```

Showing NAT Configuration

You can always view the information in configuration nodes by using the **show** command in configuration mode. NAT is a protocol service, so in this case you can view NAT configuration by using the **show service nat** command, as shown in Example 12-4.

To show the information in the **service nat** configuration node, perform the following step in configuration mode:

Example 12-4 Viewing the “service nat” configuration node

Step	Command

Show the contents of the **service nat** configuration node.	`root@R1# show service nat`
	`  rule 1 {`
	`  type: "source"`
	`  translation-type: "masquerade"`
	`  inbound-interface: "wan0"`
	`  protocols: "all"`

Example 12-4 Viewing the “service nat” configuration node

```
            source {
               network: 172.16.0.0/24
            }
            destination {
               network: 0.0.0.0/0
            }
         }
         rule 2 {
            type: "source"
            translation-type: "masquerade"
            inbound-interface: "wan0"
            protocols: "all"
            source {
               network: 10.10.30.0/24
            }
            destination {
               network: 0.0.0.0/0
            }
         }
         rule 3 {
            type: "source"
            translation-type: "masquerade"
            inbound-interface: "wan0"
            protocols: "all"
            source {
               network: 10.10.30.0/24
            }
            destination {
               network: 0.0.0.0/0
            }
         }
```

Example 12-4 Viewing the “service nat” configuration node

```
                rule 4 {
                    type: "destination"
                    translation-type: "static"
                    inbound-interface: "wan0"
                    protocols: "tcp"
                    source {
                        network: 0.0.0.0/0
                    }
                    destination {
                        address: 99.99.99.65
                        port-name: “ssh”
                    }
                    inside-address {
                        address: 10.10.30.30
                    }
                }
            [edit]
            root@R1#
```

Chapter 13: Firewall

This chapter describes how to configure the Firewall on the Vyatta OFR.

The following topics are covered:

- Firewall Overview
- Configuring the Firewall
- Viewing Firewall Information

Firewall Overview

The Vyatta OFR's firewall functionality analyzes and filters IP packets between network interfaces. The most common application of this is to protect traffic between an internal network and the Internet. It allows you to filter packets based on their characteristics and perform actions on packets that match the rule. It provides:

- Packet filtering can be performed for traffic traversing the router, using "in" and "out" on an interface. Packets destined to the router itself can be filtered using the "local" keyword.
- Criteria that can be defined for packet-matching rules include source IP address, destination IP address, source port, destination port, IP protocol, and ICMP type.
- General detection on IP options such as source routing and broadcast packets

The Vyatta OFR is not a full-featured firewall, but it can provide significant additional protection in a layered security strategy. The router can intercept network activity, categorize it against its configured database of permitted traffic, and allow or deny the attempt. This adds add an extra layer of security when used in conjunction with stateful packet-filtering devices.

To use the firewall feature, you define a firewall rule set as a named firewall instance. You then apply the firewall instance to interfaces, where the instance acts as a packet filter. The firewall instance will filter packets in one of the following ways, depending on what you specify when you apply the firewall instance:

- **in.** If you apply the rule set as **in**, the firewall will filter packets entering the interface.
- **out.** If you apply the rule set as **out**, the firewall will filter packets leaving the interface.
- **local.** If you apply the rule set as **local**, the firewall will filter packets destined for the router directly connected to this interface.

For each interface, you can apply up to three firewall instances: one firewall **in** instance, one firewall **out** instance, and one firewall **local** instance.

Note that after the final user-defined rule in a rule set is executed, an implicit rule of **deny all** takes effect.

Make sure the firewall instance you apply to an interface is already defined, or you may experience unintended results. If you apply a firewall instance that does not exist to an interface, the implicit firewall rule of **allow all** will be applied.

Configuring the Firewall

This section sets up a basic firewall configuration. To configure the firewall:

1 You define a number of named firewall rule sets. This includes:

- Specifying match conditions for traffic.
- Specifying the action to be taken if traffic matches the specified criteria. Traffic can be **accepted**, silently **dropped**, or **rejected** with a TCP reset.

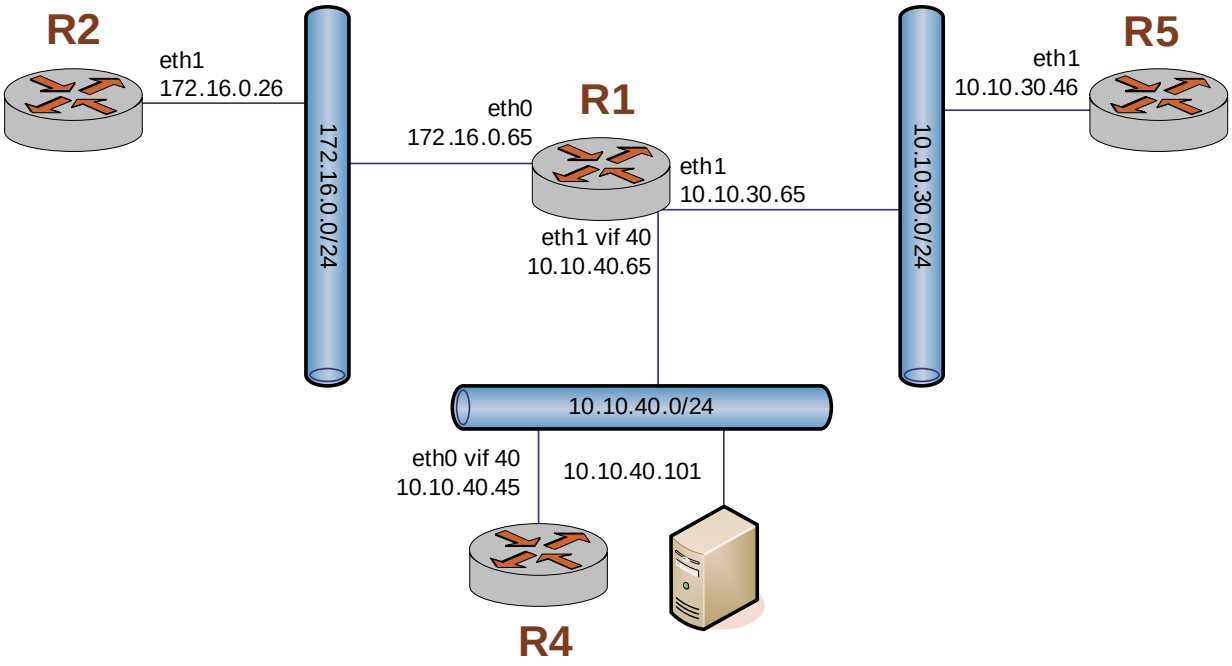
2 You apply the named rule sets to an interface as packet filters. You can apply one named rule set to each of the following:

- **in.** If you apply the rule set as **in**, the firewall will filter packets entering the interface.
- **out.** If you apply the rule set as **out**, the firewall will filter packets leaving the interface.
- **local.** If you apply the rule set as **local**, the firewall will filter packets destined for the router directly connected to this interface.

Note that after the final user-defined rule in a rule set is executed, an implicit rule of **reject all** takes effect.

This section presents a sample configuration for firewall. When you have finished, the router will be configured on router R1 as shown in Figure 13-1.

Figure 13-1 Firewall



This section includes the following examples:

- Example 13-1 Filtering on source IP
- Example 13-2 Filtering on source and destination IP
- Example 13-3 Filtering on source IP and destination protocol
- Example 13-4 Defining a network-to-network filter

Filter on Source IP

Example 13-1 defines a firewall rule set containing one rule, which filters on source IP address only. This rule will deny packets coming from router R2. It then applies the firewall rule set to packets inbound on interface eth0.

To create a rule set that filters on source IP, perform the following steps in configuration mode:

Example 13-1 Filtering on source IP

Step	Command
Create the configuration node for FWTEST-1 and its rule Rule 1. This rule rejects traffic matching the specified criteria.	`root@R1# set firewall name FWTEST-1 rule 1 action reject` `[edit]`
This rule applies to traffic that has 176.16.10.26 as the source.	`root@R1# set firewall name FWTEST-1 rule 1 source address 172.16.10.26` `[edit]`
Apply FWTEST-1 to inbound packets on eth0.	`root@R1# set interfaces ethernet eth0 firewall in name FWTEST-1` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Filter on Source and Destination IP

Example 13-2 defines another firewall rule set. It contains one rule, which filters on both source and destination IP address. This rule accepts packets leaving R5 through eth1 using 10.10.30.46, and destined for 10.10.40.101. It then applies the firewall rule set to packets outbound from vif 1 on interface eth1.

To create a rule set that filters on source and destination IP, perform the following steps in configuration mode:

Example 13-2 Filtering on source and destination IP

Step	Command
Create the configuration node for FWTEST-2 and its rule Rule 1. This rule accepts traffic matching the specified criteria.	`root@R1# set firewall name FWTEST-2 rule 1 action accept` `[edit]`

Example 13-2 Filtering on source and destination IP

This rule applies to traffic that has 10.10.30.46 as the source.	`root@R1# set firewall name FWTEST-2 rule 1 source address 10.10.30.46` `[edit]`
This rule applies to traffic that has 10.10.40.101 as the destination.	`root@R1# set firewall name FWTEST-2 rule 1 destination address 10.10.40.101` `[edit]`
Apply FWTEST-2 to outbound packets on eth1 vif 40.	`root@R1# set interfaces ethernet eth1 vif 40 firewall out name FWTEST-2` `[edit]`
Commit the configuration.	`root@R1# commit` `[edit]`

Filter on Source IP and Destination Protocol

Example 13-3 defines a firewall rule that filters on source IP address and destination protocol. This rule allows TCP packets originating from address 10.10.30.46 (that is, R5), and destined for the Telnet port of R1. The rule set is applied to local packets (that is, packets destined for this router, R1) through eth1.

To create a rule set that filters on source IP and destination protocol, perform the following steps in configuration mode:

Example 13-3 Filtering on source IP and destination protocol

Step	Command
Create the configuration node for FWTEST-3 and its rule Rule 1. This rule accepts traffic matching the specified criteria.	`root@R1# set firewall name FWTEST-3 rule 1 action accept` `[edit]`
This rule applies to traffic that has 10.10.30.46 as the source.	`root@R1# set firewall name FWTEST-3 rule 1 source address 10.10.30.46` `[edit]`
This rule applies to TCP traffic.	`root@R1# set firewall name FWTEST-3 rule 1 protocol tcp` `[edit]`
This rule applies to traffic that is destined for the Telnet service.	`root@R1# set firewall name FWTEST-3 rule 1 destination port-name telnet` `[edit]`
Apply FWTEST-3 to packets bound for this router arriving on eth1.	`root@R1# set interfaces ethernet eth1 firewall local name FWTEST-3` `[edit]`

Example 13-3 Filtering on source IP and destination protocol

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Defining a Network-to-Network Filter

Example 13-4 creates a network-to-network packet filter, allowing packets originating from 10.10.40.0/24 and destined for 172.16.0.0/24. It then applies the firewall rule set to packets inbound through vif 40 on interface eth1.

To create a network-to-network filter, perform the following steps in configuration mode:

Example 13-4 Defining a network-to-network filter

Step	Command
Create the configuration node for FWTEST-4 and its rule Rule 1. This rule accepts traffic matching the specified criteria.	`root@R1# set firewall name FWTEST-4 rule 1 action accept` `[edit]`
This rule applies to traffic coming from the network 10.10.40.0/24.	`root@R1# set firewall name FWTEST-4 rule 1 source network 10.10.40.0/24` `[edit]`
This rule applies to traffic destined for the network 172.16.0.0/24.	`root@R1# set firewall name FWTEST-4 rule 1 destination network 172.16.0.0/24` `[edit]`
Apply FWTEST-4 to packets bound for this router arriving through vif 40 on eth1.	`root@R1# set interfaces ethernet eth1 vif 40 firewall in name FWTEST-4` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Viewing Firewall Information

This section includes the following examples:

- Example 13-5 Showing a firewall rule set
- Example 13-6 Showing firewall configuration on an interface
- Example 13-7 Displaying the “firewall” configuration node
- Example 13-8 Showing the firewall configuration of an interface

Showing Firewall Rule Set Information

You can see how firewall rule sets are set up by using the **show firewall** command in operational mode and specifying the name of the rule set. Example 13-5 shows what you configured for firewall rule set FWTEST-1.

Example 13-5 Showing a firewall rule set

```
root@R1> show firewall FWTEST-1

State Codes: E - Established, I - Invalid, N - New, R - Related

rule  action  source            destination         proto  state
----  ------  ------            -----------         -----  -----
1     REJECT  172.16.10.26       0.0.0.0/0            all    any
1025  DROP    0.0.0.0/0          0.0.0.0/0                   any


root@R1>
```

Showing Firewall Configuration on Interfaces

Example 13-6 shows how firewall rule set FWTEST-1 is applied to interface eth0.

Example 13-6 Showing firewall configuration on an interface

```
root@R1# show interfaces ethernet eth0 firewall
   in {
      name: "FWTEST-1"
   }

[edit]
root@R1#
```

Showing Firewall Configuration

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case you can view firewall configuration by using the **show firewall** command in configuration mode, as shown in Example 13-7.

Example 13-7 Displaying the “firewall” configuration node

```
root@R1# show firewall
    name "FWTEST-1" {
        rule 1 {
            action: "reject"
            source {
                address: 172.16.10.26
            }
        }
    }
    name "FWTEST-2" {
        rule 1 {
            action: "accept"
            source {
                address: 10.10.30.46
            }
            destination {
                address: 10.10.40.101
            }
        }
    }
    name "FWTEST-3" {
        rule 1 {
```

```
            protocol: "tcp"
            action: "accept"
    source {
                address: 10.10.30.46
            }
            destination {
                port-name: "telnet"
            }
        }
    }
    name "FWTEST-4" {
        rule 1 {
            action: "accept"
            source {
                network: 10.10.40.0/24
            }
            destination {
                network: 172.16.0.0/24
            }
        }
    }

[edit]
root@R1#
```

To see what firewall rule sets have been applied to a given interface, use the **show interfaces ethernet** *int-name* **firewall** command, as shown in Example 13-8.

Example 13-8 Showing the firewall configuration of an interface

```
root@R1> configure
Entering configuration mode.
There are no other users in configuration mode.
root@R1# show interfaces ethernet eth0 firewall
    in {
        name: "FWTEST-1"
    }

[edit]
root@R1#
```

Chapter 14: User Authentication

This chapter describes how to set up user accounts and user authentication.

The following topics are covered:

- Authentication Overview
- Creating “Login” User Accounts
- Configuring for a RADIUS Server
- Viewing Authentication Information

Authentication Overview

The Vyatta OFR supports either of the following options for user account management:

- A local user database (“login” authentication).
- RADIUS authentication server

The system creates two login user accounts by default: user **vyatta** and user **root**. The user account **vyatta** can be deleted, but the user account **root** is protected and cannot be deleted. The default password for each is **vyatta**.

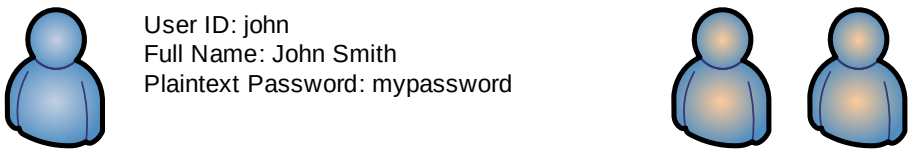
By default, users are authenticated first using the local user database (“login” authentication). If this fails, the system looks for a configured RADIUS server. If found, the router queries the RADIUS server using the supplied RADIUS secret. After the query is validated, the server authenticates the user from information in its database.

You supply login user passwords and RADIUS secrets in plain text. After configuration is committed, the system encrypts them and stores the encrypted version internally. When you display user configuration, only the encrypted version of the password or secret is displayed.

Creating “Login” User Accounts

In this section, a sample configuration is presented for a user account that will be validated using the local user database. The sample configuration used is shown in Figure 14-1.

Figure 14-1 “Login” User Account



This section includes the following example:

- Example 14-1 Creating a “login” user account

Example 14-1 creates a user account for **John Smith**. John has a user ID of **john** and will use a plain text password of **mypassword**. Note that once configuration has been committed, only the encrypted version of the password displays when configuration is shown.

NOTE *User information can be changed through the UNIX shell (providing you have sufficient permissions). However, any changes to OFR user accounts or authentication through the UNIX shell will be overwritten the next time you commit OFR CLI configuration.*

To create a login user account, perform the following steps in configuration mode:

Example 14-1 Creating a "login" user account

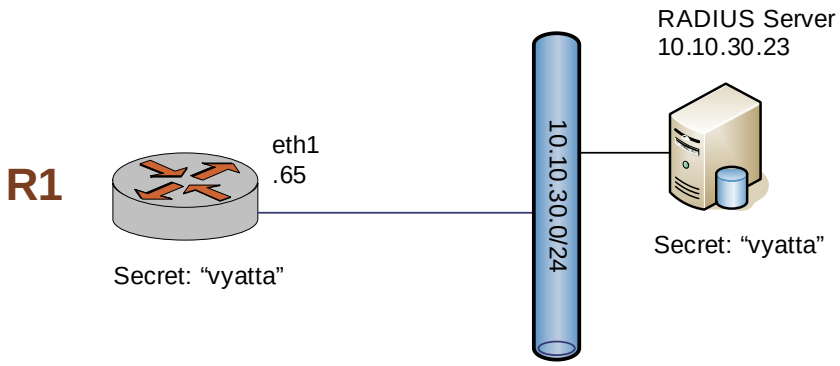
Step	Command
Create the user configuration node, define the user ID, and give the user's full name.	`root@R1# set system login user john full-name "John Smith"` `[edit]`
Specify the user's password in plain text.	`root@R1# set system login user john authentication plaintext-password mypassword` `[edit]`

Commit the change. After a password has been committed, it can be displayed only in encrypted form, as the value of the **encrypted-password** attribute.	`root@R1# commit`
	`OK`
	`[edit]`

Configuring for a RADIUS Server

In this section, a sample configuration is presented for a user account that will be validated using the RADIUS authentication server. The sample configuration used is shown in Figure 14-2.

Figure 14-2 RADIUS User Account



This section includes the following example:

- Example 14-2 Configuring for a RADIUS server

Example 14-2 defines a RADIUS authentication server at 10.10.30.23. The router will access the RADIUS server using a secret of **vyatta**. In this example, the port used for RADIUS traffic is left at the default 1812, which is the well-known port for RADIUS. The timeout after which the next RADIUS server should be queried is left at 2 seconds.

To define a RADIUS server, perform the following steps in configuration mode:

Example 14-2 Configuring for a RADIUS server

Step	Command
Provide the location of the server, and the secret to be used to access it.	`root@R1# set system radius-server 10.10.30.23 secret vyatta` `[edit]`

Commit the change. After the secret has been committed, it can be displayed only in encrypted form.	`root@R1# commit`
	`OK`
	`[edit]`

Viewing Authentication Information

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view authentication configuration by using the **show system login** command or the show system radius-server command, as shown in Example 14-3.

To show user authentication configuration, perform the following step in configuration mode:

Example 14-3 Viewing the “system login” and “system radius-server” configuration nodes

Step	Command

Show the contents of the **system login** configuration node.	`root@R1# show system login`
	`user root {`
	`  authentication {`
	`  encrypted-password: “$1$A.EPAdNj$/2bTvc433VZ.VV5YWAbd1"`
	`  }`
	`}`
	`user vyatta {`
	`  authentication {`
	`  encrypted-password: "$1$$ZbzUPUD24iyfRwCKIT16q0"`
	`  }`
	`}`
	`user john {`
	`  full-name: "John Smith"`
	`  authentication {`
	`  encrypted-password: "!!"`
	`  }`
	`  }`

Show the contents of the **system radius-server** configuration node.	`root@R1# show system radius-server`
	`radius-server 10.10.30.23 {`
	`  port: 1812`
	`  secret: “EPAdNj$/2b”`
	`  timeout: 2`
	`}`

Chapter 15: Logging

This chapter explains how the Vyatta OFR generates and manages system logging messages, and describes how to configure logging.

The following topics are covered:

- Logging Overview
- Enabling and Disabling Logging for Specific Features

Logging Overview

Significant system events are captured in log messages (also called syslog messages), which you can view on the console, save to a file, or forward to an external server such as a syslog server, or direct to the terminal session of one or more specific users.

Depending on the level of message severity you choose to log, system log messages (also called syslog messages) can include notices of ordinary and routine operations, as well as warnings, failure, and error messages.

The Vyatta OFR's logging function makes use of the UNIX **syslogd** process. Logging configuration performed within the router's CLI is stored in the **/etc/sylogd.conf** file.

By default, local logging is enabled, and sends messages to **/var/log/messages**.

Logging Facilities

The Vyatta OFR supports standard syslog facilities. These are as follows:

Table 15-1 Syslog facilities

Facility	Description
auth	Authentication and authorization
authpriv	Non-system authorization
cron	Cron daemon
daemon	System daemons
kernel	Kernel
lpr	Line printer spooler
mail	Mail subsystem
mark	Timestamp
news	USENET subsystem
security	Security subsystem
syslog	System logging
user	Application processes
uucp	UUCP subsystem
local0	Local facility 0
local1	Local facility 1

Table 15-1 Syslog facilities

local2	Local facility 2
local3	Local facility 3
local4	Local facility 4
local5	Local facility 5
local6	Local facility 6
local7	Local facility 7
*	All facilities excluding “mark”

In addition, logging can be selectively enabled for some specific routing components. For this informaiton, please see the section ““Enabling and Disabling Logging for Specific Features” on page 145.

Log Destinations

When logging is enabled, system log messages are always written to the **messages** file in the **/var/log** directory of the local file system. In addition, system logs can be sent to the console, to a named file in the local file system, to a server running the **syslogd** utility (that is, a syslog server), or to the terminal session of one or more specific users.

- To direct syslog messages to the console, use the **system syslog console** command.
- To direct syslog messages to a named file in the local file system, use the **system syslog file** command.
- To direct syslog messages to a remote machine running the **syslogd** utility, use the **system syslog host** command.
- To direct syslog messages to the terminal of a specific user, to multiple users, or to all users logged into the routing platform, use the **system syslog user** command.

Log File Locations and Archiving

Messages are written either to the main log file (the default) or to a file that you specify. The main log file is created in the **/var/log** directory, to the **messages** file. User-defined log files are written to the **/var/log/user** directory.

The router uses standard UNIX log rotation to prevent the file system from filling up with log files. When log messages are written to a file, the system will write up to 500 KB of log messages into the file *logfile*, where *logfile* is either the system-defined **messages** file, or a name you have assigned to the file. When *logfile* reaches its maximum size, the system closes it and compresses it into an archive file. The archive file is named *logfile***.0.gz**.

At this point, the logging utility opens a new *logfile* file and begins to write system messages to it. When the new log file is full, the first archive file is renamed *logfile***.1.gz** and the new archive file is named *logfile***.0.gz**. The system archives log files in this way until a maximum number of log files exists. By default, this is five (that is, up to *logfile***.4.gz**), where *logfile***.0.gz** always represents the most recent file. After this, the oldest log archive file is deleted as it is overwritten by the next oldest file.

To change the properties of log file archiving, configure the **system syslog archive** node:

- Use the **size** parameter to specify the maximum size of the log file.
- Use the **files** parameter to specify the maximum number of archive files to be maintained.

Log Severities

System events generate log messages at different severities, which represent their level of importance for the system.

When you configure severity level for syslog, the system captures log messages at that severity and above. The lower the level of severity specified, the more detail is captured in the logs. For example, if you configure a log severity level of **crit**, the system captures log messages that have severity **crit**, **alert**, and **emerg**.

Currently, log severity is configurable *for user-defined log files only.* The main log file in **/var/log/messages** captures log messages of severity **warning** and above.

Log messages generated by the Vyatta OFR router will be associated with one of the following levels of severity.

NOTE *Currently, log severities can only be changed for user-defined files. The main log file at /var/log/messages always uses log severity* ***warning****..*

Table 15-2 Syslog message severities

Severity	Meaning
emerg	Emergency. A general system failure or other serious failure has occurred, such that the router is unusable.
alert	Alert. Immediate action is required to prevent the system from becoming unusable—for example, because a network link has failed, or the database has become compromised.
crit	Critical. A critical condition exists, such as resource exhaustion—for example, the system is out of memory, CPU processing thresholds are being exceeded, or a hardware failure has occurred.
err	Error. An error condition has occurred, such as a failed system call. However, the system is still functioning.
warning	Warning. An event has occurred that has the potential to cause an error, such as invalid parameters being passed to a function. This situation should be monitored.
notice	Notice. A normal but significant event has occurred, such as an unexpected event. It is not an error, but could potentially require attention.
info	Informational. Normal events of interest are being reported as they occur.
debug	Debug level. Trace-level information is being provided.

When you specify a severity to report, the system understands that as the *minimum* severity to report, and it reports all log messages of that severity and higher. For example, if you specify a severity of Alert, the system reports all Alert, Critical, and Emergency log messages.

CAUTION *Risk of service degradation. Debug severity is resource-intensive. Setting logging levels to Debug can affect performance.*

Enabling and Disabling Logging for Specific Features

Some features of the OFR—currently BGP and OSPF—produce feature-specific log messages that can be enabled and disabled. When you enable logging for a system feature, the log messages are sent to whatever destinations are configured for syslog.

By default, log messages are sent to the main log file at **/var/log/messages**. You can configure syslog to send log messages to a file you specify in **/var/user**.

- For information about configuring BGP logging, please see “Chapter 10: BGP.”
- For information about configuring OSPF logging, please see “Chapter 9: OSPF.”

Chapter 16: SNMP

This chapter describes how to configure Simple Network Management Protocol on the Vyatta OFR.

The following topics are covered:

- SNMP Overview
- Configuring SNMP Information
- Viewing SNMP Information

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.

MIB Objects

The MIB contains the set of variables/objects that are managed (e.g., MTU on a network interface). Those objects are organized in a tree structure where each object is a leaf node. Each object has its unique Object Identifier (OID).

There are two types of objects: *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 router 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 will asynchronously send the traps to the SNMP managers that are configured as trap destinations or *targets*. This keeps the network manager informed of the status and health of the device.

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 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 manager to change the value of a specific object. Setting a configuration object changes the device’s configuration. 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 there are three versions of SNMP:

- SNMP v1. This is the first version of the protocol. It is described in RFC 1157.
- SNMP v2. This is an evolution of the first version, and it adds a number of improvements to SNMPv1.
- SNMP v3. This version improves the security model in SNMPv2, and adds support for proxies.

The Vyatta OFR supports SNMP v2 with community string (SNMP v2c)

SNMP MIB Locations

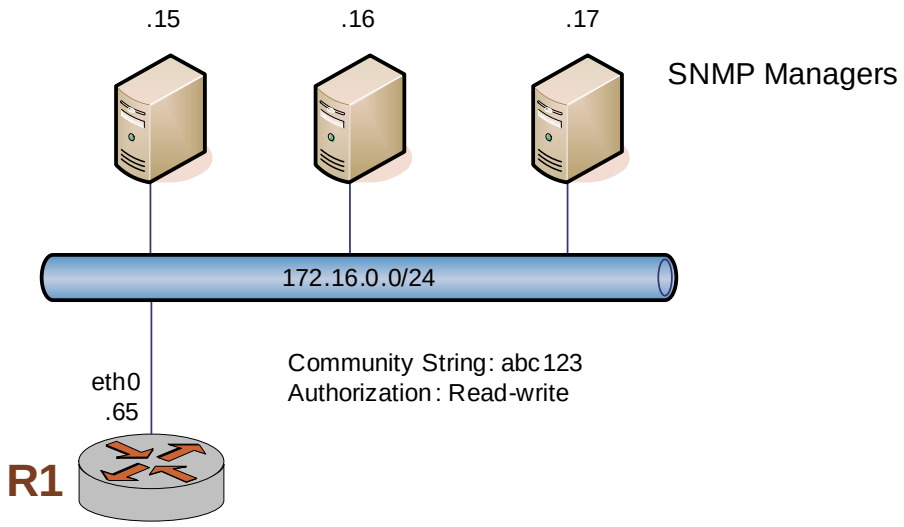
MIBs are typically located in the **/usr/share/snmp/mibs** directory. The Vyatta OFR does not currently have its own enterprise MIB.

Configuring SNMP Information

To configure SNMP, there must be at least one user created, and the Vyatta MIB model must be loaded.

This sequence sets up an SNMP community that includes three hosts, which will serve as SNMP managers, and configures the router to send traps to all three managers. When you have finished, the router will be configured as shown in Figure 16-1.

Figure 16-1 Configuring SNMP communities and traps



This section includes the following examples:

- Example 16-1 Defining an SNMP community
- Example 16-2 Specifying SNMP trap destinations

Defining the SNMP Community

The SNMP community of a router is the list of SNMP clients authorized to make requests of the router. 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 router against spurious SNMP requests.

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

Example 16-1 sets the SNMP community string to abc123 and specifies three clients for the community: 176.16.0.15, 176.16.0.16, and 176.16.0.17. Read-write access is provided for this community.

To define an SNMP community, perform the following steps in configuration mode:

Example 16-1 Defining an SNMP community

Step	Command
Create the **snmp** configuration node and the **community** configuration node. Set the community string. Navigate to the configuration node of the community.	root@R1# **set protocols snmp community abc123** [edit] root@R1# **edit protocols snmp community abc123** [edit protocols snmp community abc123]
List the SNMP clients making up this community.	root@R1# **set client 176.16.0.15** [edit protocols snmp community abc123] root@R1# **set client 176.16.0.16** [edit protocols snmp community abc123] root@R1# **set client 176.16.0.17** [edit protocols snmp community abc123]
Set the privilege level for this community to read-write.	root@R1# **set authorization rw** [edit protocols snmp community abc123]
Commit the change, and return to the top of the configuration tree.	root@R1# **commit** OK [edit protocols snmp community abc123] root@R1# **top** [edit]

Specifying Trap Destinations

Example 16-1 directs the router to send SNMP traps to the configured network managers at 176.16.0.15, 176.16.0.16, and 176.16.0.17.

To specify trap destinations, perform the following steps in configuration mode:

Example 16-2 Specifying SNMP trap destinations

Step	Command

Define the trap destinations, one at a time.	`root@R1# set protocols snmp trap-target 176.16.0.15`
	`[edit]`
	`root@R1# set protocols snmp trap-target 176.16.0.16`
	`[edit]`
	`root@R1# set protocols snmp trap-target 176.16.0.17`
	`[edit]`

Commit the change.	`root@R1# commit`
	`OK`
	`[edit]`

Viewing SNMP Information

You can always view the information in configuration nodes by using the **show** command in configuration mode. In this case, you can view SNMP configuration by using the **show protocols snmp** command, as shown in Example 16-3.

To show the information in the **protocols snmp** configuration node, perform the following step in configuration mode:

Example 16-3 Viewing the “protocols snmp” configuration node

Step	Command

Show the contents of the **protocols snmp** configuration node.	`root@R1# show protocols snmp`
	`community abc123 {`
	`client 176.16.0.15`
	`client 176.16.0.16`
	`client 176.16.0.17`
	`authorization: "rw"`
	`}`

Chapter 17: Software Upgrades

This chapter describes how to use the Vyatta OFR's package mechanism to update your software.

The following topics are covered:

- Upgrade Overview
- Configuring for Automatic Upgrade
- Working with Packages

Upgrade Overview

Software components updates are stored in *packages*. A package is a precompiled software component that can be installed or upgraded independently of other software components on your machine. Each package contains software and any scripts required to install, configure, or remove it.

To update software packages, you must be running the router shell as user **root**.

Vyatta OFR packages are stored in the Vyatta software repository.

- The stock repository name for the current release is **vyatta/VC2**.
- The host on which the repository resides is **archive.vyatta.com**.

The Vyatta OFR keeps a record of which of all available packages have been installed on the router and which have not. The system keeps track of differences between packages, including differences between versions of packages, as necessary. The system also keeps track of dependencies between packages. When you direct the system to install a package, the system will also install any packages on which the specified package depends.

Installation of packages is atomic: all packages, and the packages on which they depend must install successfully, or else the installation will be rolled back, leaving the system as it was before.

Some updates (for example, a kernel upgrade) may require a system reboot, but in general, updates will not require a reboot.

Configuring for Automatic Upgrade

Software update is configured in the **system package** configuration node. To use the automatic upgrade feature, the system must be configured:

- The name of the software repository, and
- The URL of the host on which the repository is to be found.

You can view package configuration by using the **show system package** command in configuration mode, as shown in Example 17-1.

Example 17-1 Viewing the “system package” configuration node

Step	Command

Show the contents of the **system package configuration** node.	`root@R1# show system package`
	`repository 1.1 {`
	`component: "main"`
	`url: "http://archive.vyatta.com"`
	`}`
	
	`[edit]`
	`root@R1#`

If you need to restore package configuration, you can perform the following steps, in configuration mode:

Example 17-2 Configuring for automatic software update

Step	Command
Create the **package** configuration node, specifying the repository and host.	`root@R1# set system package repository VC2 url http://archive.vyatta.com` `[edit]`
Specify “main” as a component.	`root@R1# set system package repository VC2 component main` `[edit]`

Commit the configuration.	`root@R1# commit`
	`OK`
	`[edit]`

Working with Packages

To update software, you can use the following workflow:

1. View the List of Available Packages
2. Upgrade Packages

If you want to install a specific package, or remove a package, you can:

3. Install Packages
4. Delete a package

View the List of Available Packages

The router keeps a list of available packages, which it synchronizes with the software repository. However, you can force the system to poll the repository by issuing the **update package-list** command.

We recommend monitoring the list of available packages regularly, so that you can be aware of what updates are available.

Updating the list of packages may take some time to complete, and the system displays a progress indicator. You can cancel the update at any time, by pressing <Ctrl>-c.

To update the list of packages, issue the **update package-list** command in operational mode:

Example 17-3 Updating the package list

```
root@R1> update package-list
```

To view the information in the current package list, issue the **show package info** command in operational mode:

Example 17-4 Viewing package information

```
root@R1> show package info
```

Upgrade Packages

Running this command upgrades software packages on your system, including their dependencies. All packages are upgraded, unless you filter upgrades by specifying package names.

Packages are downloaded from the repository and upgraded in the correct order. Packages are upgraded to the most recent version available in the repository, provided all dependencies can be satisfied. Packages for which dependencies cannot be satisfied, or that have conflicts with installed software, are not “kept back” and not installed.

Before upgrading, you should use the **show package info** command to confirm the complete list of packages that will be upgraded.

To upgrade software packages, issue the **update package** command in operational mode:

Example 17-5 Upgrading software packages

```
root@R1> update package
```

Install Packages

Instead of upgrading your entire system, you can select specific packages to install.If the package has a dependency, the necessary packages will also be installed.

To upgrade software packages, issue the **install package** command in operational mode. Example 17-6 installs any package matching the string **grub-man**.

Example 17-6 Installing specific packages

```
root@R1> install package grub-man
```

Removing a Package

You can remove any unwanted packages from the system. When you remove a package, any packages that depend on it are also removed. You cannot remove a package without removing packages that depend on it. Although the packages are removed, the configuration files (if any) will remain in the system.

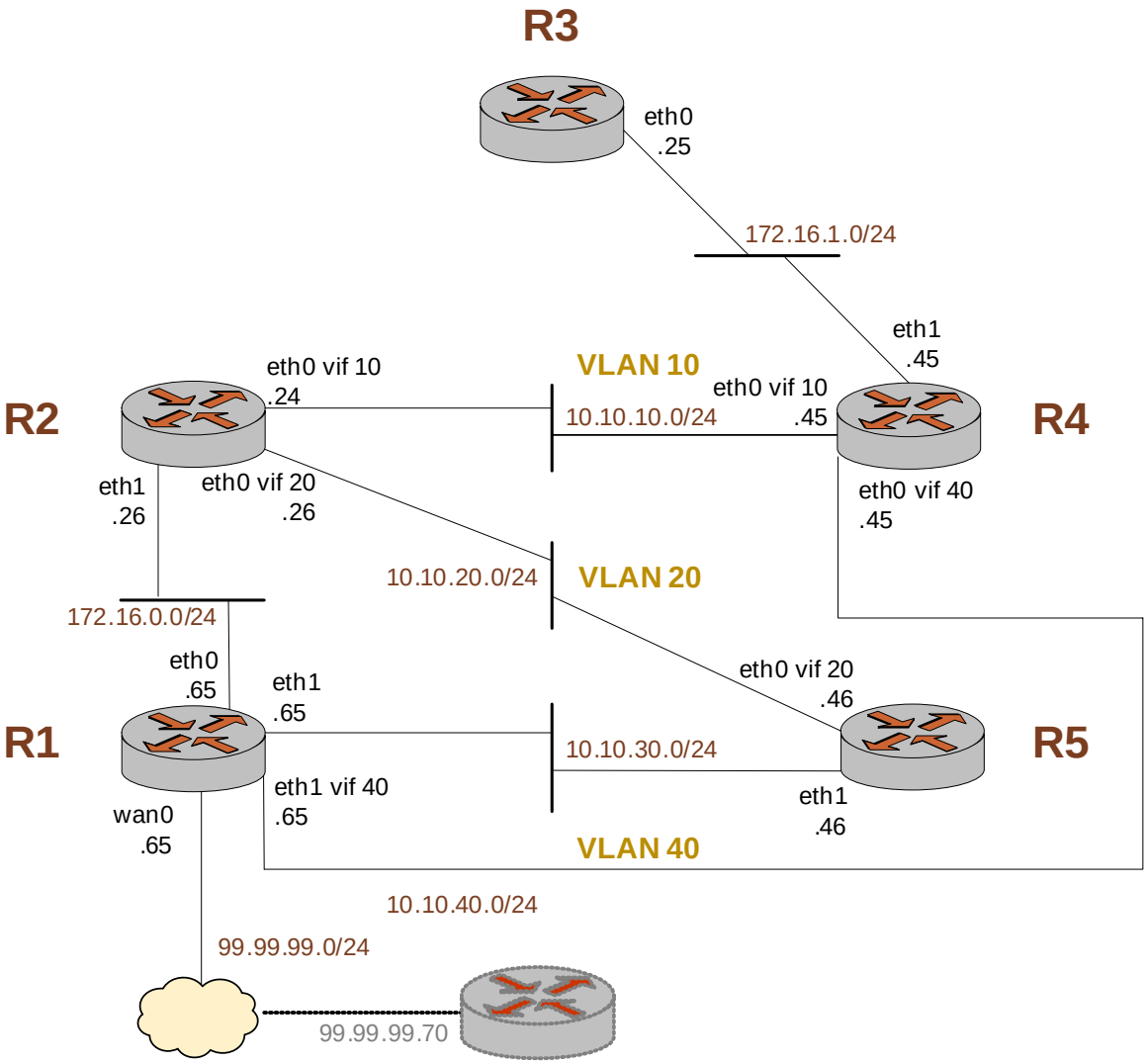
To remove a software package, issue the **delete package** command in operational mode. Example 17-7 removes any package matching the string **grub-man**.

Example 17-7 Removing a package

```
root@R1> delete package grub-man
```

Topology Diagram

This section shows the topology used in examples in Vyatta documentation.



Quick Guide to Configuration Statements

Use this section to quickly see the complete syntax of configuration statements.

The Vyatta OFR supports the following configuration statements:

- firewall
- interfaces
- multicast
- policy
- protocols
- rtrmgr
- service
- system
- vpn

firewall

```
firewall {
   log-martians: [enable|disable]
   send-redirects: [enable|disable]
   receive-redirects: [enable|disable]
   ip-src-route: [enable|disable]
   broadcast-ping: [enable|disable]
   syn-cookies: [enable|disable]
   name: text {
      description: text
      rule: 1–1024 {
         protocol: [all|tcp|udp|icmp|igmp|ipencap|gre|esp|ah|
            ospf|pim|vrrp]
         icmp {
            type: text {
            code: text
         }
         state {
            established: [enable|disable]
            new: [enable|disable]
            related: [enable|disable]
            invalid: [enable|disable]
            }
         action: [accept|drop|reject]
         log: [enable|disable]
         source {
            address: ipv4
            network: ipv4net
            range {
               start: ipv4
               stop: ipv4
            }
            port-number: 1–65535
            port-name: [http|ftp|smtp|telnet|ssh|dns|snmp]
            port-range {
               start: 1–65535
               stop: 1–65535
            }
         }
         destination {
            address: ipv4
            network: ipv4net
            range {
               start: ipv4
               stop: ipv4
            }
```

```
                port-number: 1–65535
                port-name: [http|ftp|smtp|telnet|ssh|dns|snmp]
                port-range {
                   start: 1–65535
                   stop: 1–65535
                }
             }
          }
       }
    }
```

interfaces

```
interfaces {
   restore: [true|false]
   loopback: lo {
      description: text
      address: [ipv4|ipv6]{
         prefix-length: [0–32|0-128]
         broadcast: ipv4
         multicast-capable: [true|false]
         disable: [true|false]
      }
   }
   bridge br0..br9 {
      description: text
      disable: [true|false]
      aging: 1–4294967296
      stp: [true|false]
      priority: 1–4294967296
      forwarding-delay: 1–4294967296
      hello-time: 1–4294967296
      max-age: 1–4294967296
   }
   ethernet: eth0..eth23 {
      disable:[true|false]
      discard:[true|false]
      description:text
      mac: mac-addr
      mtu: 68–65535
      duplex: [full|half|auto]
      speed: [10|100|1000|auto]
      address: [ipv4 | ipv6]{
         prefix-length: [0–32|0-128]
         broadcast: ipv4
         multicast-capable: [true|false]
         disable: [true|false]
      }
      bridge-group {
         bridge: br0..br9
         cost: 1–4294967296
         priority: 1–4294967296
      }
      vrrp {
         vrrp-group: 1–255
         virtual-address: ipv4
         authentication:text
         advertise-interval: 1–255
```

```
            preempt:[true|false]
            priority: 1–255
        }
        firewall {
            in {
                name: text
            }
            out {
                name: text
            }
            local {
                name: text
            }
        }
        vif 1–4096 {
            disable:[true|false]
            address: [ipv4 | ipv6]{
                prefix-length: [0–32|0-128]
                broadcast: ipv4
                multicast-capable: [true|false]
                disable: [true|false]
            }
            bridge-group {
                bridge: br0..br9
                cost: 1–4294967296
                priority: 1–4294967296
            }
            vrrp {
                vrrp-group: 1–255
                virtual-address: ipv4
                authentication:text
                advertise-interval: 1–255
                preempt:[true|false]
                priority: 1–255
            }
            firewall {
                in {
                    name: text
                }
                out {
                    name: text
                }
                local {
                    name: text
                }
            }
    serial [wan0..wan9] {
```

```
encapsulation: [ppp|cisco-hdlc|frame-relay]
description: text
t1-options {
   lbo: [0-110ft|110-220fr|220-330ft|330-440ft|440-550ft]
   timeslots {
      start: [1–24]
      stop: [1–24]
   }
   mtu: 8–8188
   clock: [internal|external]
}
e1-options {
   framing: [g704|g704-no-crc4|unframed]
   timeslots {
      start: [1–32]
      stop: [1–32]
   }
   mtu: 8–8188
   clock: [internal|external]
}
t3-options {
   framing: [c-bit|m13]
   line-coding: [ami|b8zs]
}
ppp {
   authentication {
      type: [none|chap|pap]
      user-id: text
      password: text
   }
   vif 1 {
      address {
         local-address: ipv4
         prefix-length: 0–32
         remote-address: ipv4
      }
      description: text
      firewall {
         in {
            name: text
         }
         out {
            name: text
         }
         local {
            name: text
         }
```

```
        }
    }
}
cisco-hdlc {
    keepalives {
        require-rx: [enable|disable]
        timer: 10-60000
    }
    vif 1 {
        address {
            local-address: ipv4
            prefix-length: 0–32
            remote-address: ipv4
        }
        description: text
        firewall {
            in {
                name: text
            }
            out {
                name: text
            }
            local {
                name: text
            }
        }
    }
}
frame-relay {
    signaling: [auto|ansi|q933|lmi]
    signaling-options {
        n39ldte: 1–255
        n392dte: 1–100
        n393dte: 1–10
        t391dte: 5–30
    }
    vif [16..991] {
        address {
            local-address: ipv4
            prefix-length: 0–32
            remote-address: ipv4
        }
        description: text
        firewall {
            in {
                name: text
            }
```

```
                out {
                   name: text
                }
                local {
                   name: text
                }
             }
          }
       }
    }
```

multicast

```
multicast {
   mfea4 {
      disable:bool
      interface: eth0..eth23
      traceoptions {
         flag {
            all {
               disable:bool
            }
         }
      }
   }
   mfea6 {
      disable:bool
      interface: eth0..eth23
      traceoptions {
         flag {
            all {
               disable:bool
            }
         }
      }
   }
}
```

policy

```
policy {
   policy-statement: text {
      term: text {
         from {
            protocol: text
            network4: ipv4net
            network6: ipv6net
            network4-list: text
            network6-list: text
            prefix-length4: 0–32-range
            prefix-length6: 0–128-range
            nexthop4: ipv4-range
            nexthop6: ipv6-range
            as-path: text
            as-path-list: text
            community: text
            community-list: text
            neighbor: ipv4-range
            origin: [0|1|2]
            med: int-range
            localpref: int-range
            metric: 1–65535-range
            external: [type-1|type-2]
            tag: int-range
         }
         to {
            network4: ipv4net
            network6: ipv6net
            network4-list: text
            network6-list: text
            prefix-length4: 0–32-range
            prefix-length6: 0–128-range
            nexthop4: ipv4-range
            nexthop6: ipv6-range
            as-path: text
            as-path-list: text
            community: text
            neighbor: ipv4-range
            origin: int
            med: int-range
            localpref: int-range
            was-aggregated: bool
            metric: 1–65535-range
            external: [type-1|type-2]
            tag: int-range
```

```
            }
            then {
               action: [accept|reject]
               trace: int
               nexthop4: next-hop
               nexthop6: ipv6
               as-path-prepend: int
               as-path-expand: int
               community: text
               community-add: text
               community-del: text
               origin: int
               med: int
               med-remove: [true|false]
               localpref: int
               aggregate-prefix-len: int
               aggregate-brief-mode: int
               metric: 1–65535
               external: [type-1|type-2]
               tag: int
            }
         }
      }
      community-list: text {
         elements: text
      }
      community-list: text {
         elements: text
      }
      network6-list: text {
         elements: text
      }
   }
```

protocols

```
protocols
   bgp {
      bgp-id: ipv4
      local-as: 1–65535
      route-reflector {
          cluster-id: ipv4
          disable: [true|false]
      }
      confederation {
          identifier: 1–4294967296
          disable: [true|false]
      }
      damping {
          half-life: 1–4294967296
          max-suppress: 1–4294967296
          reuse: 1–4294967296
          suppress: 1–4294967296
          disable: [true|false]
      }
      peer: text {
         peer-port: 1–4294967296
         local-port: 1–4294967296
         local-ip: text
         as: 1–65535
         next-hop: ipv4
         holdtime: 0,3–65535
         delay-open-time: 1–4294967296
         client: [true|false]
         confederation-member: [true|false]
         prefix-limit {
            maximum: 1–4294967296
            disable: [true|false]
         disable: [true|false]
         ipv4-unicast: [true|false]
         ipv4-multicast: [true|false]
         }
      traceoptions {
         flag {
            verbose {
               disable: [true|false]
            }
            all {
               disable: [true|false]
            }
            message-in {
```

```
            disable: [true|false]
         }
         message-out {
            disable: [true|false]
         }
         state-change {
            disable: [true|false]
         }
         policy-configuration {
            disable: [true|false]
         }
      }
      import: text
      export: text
   }
}
ospf4 {
   router-id: ipv4
   RFC1538Compatibility: [true|false]
   ip-router-alert: [true|false]
   traceoptions {
   flag {
      all {
         disable:[true|false]
      }
   }
   area: ipv4 {
      area-type:[normal|stub|nssa]
      default-lsa {
         disable:[true|false]
         metric: 1–4294967296
      }
      summaries {
         disable:[true|false]
      }
      area-range: ipv4net {
         advertise:[true|false]
      }
      virtual-link: ipv4 {
         transit-area: ipv4
         hello-interval:1–65535
         router-dead-interval: 1–4294967295
         retransmit-interval: 1–65535
         transit-delay:0–3600
         authentication {
            simple-password:text
            md5: 0–255 {
               password: text
```

```
                    start-time: YYYY-MM-DD.HH:MM
                    end-time: YYYY-MM-DD.HH:MM
                    max-time-drift: 0–65534,65535
                }
            }
        }
        interface: text {
            link-type:[broadcast|p2p|p2m]
            address: ipv4 {
                priority:0–255
                hello-interval:1–65535
                router-dead-interval: 1–4294967296
                interface-cost:1–65535
                retransmit-interval: 1–65535
                transit-delay:0–3600
                authentication {
                    simple-password:text
                    md5: 0–255 {
                        password: text
                        start-time: YYYY-MM-DD.HH:MM
                        end-time: YYYY-MM-DD.HH:MM
                        max-time-drift: 0–65534,65535
                    }
                }
                passive: [true|false]
                neighbor: ipv4 {
                    router-id: ipv4
                }
                disable: [true|false]
            }
        }
    }
    import: text
    export: text
}
rip {
    interface: text {
        address: ipv4 {
            metric: 1–16
            horizon:
                [none|split-horizon|split-horizon-poison-reverse]
            disable: [true|false]
            passive: [true|false]
            accept-non-rip-requests: [true|false]
            accept-default-route: [true|false]
            advertise-default-route: [true|false]
            route-expiry-secs: 1–4294967296
            route-deletion-secs: 1–4294967296
```

```
                triggered-update-min-secs: 1–4294967296
                triggered-update-max-secs: 1–4294967296
                table-announce-min-secs: 1–4294967296
                table-announce-max-secs: 1–4294967296
                table-request-secs: 1–4294967296
                interpacket-delay-msecs: 1–4294967296
                authentication {
                   simple-password: text
                   md5: 0–255 {
                      password: text
                      start-time: YYYY-MM-DD.HH:MM
                      end-time: YYYY-MM-DD.HH:MM
                   }
                }
             }
          }
          import: text
          export: text
       }
       snmp {
          mib-module: text {
             abs-path:text
             mib-index:int
          }
          community: text {
             authorization: [ro|rw]
             client: ipv4 {}
          }
          contact: text
          description: text
          location: text
          trap-target: ipv4 {}
       }
       static {
          disable: [true|false]
          route: ipv4net {
             next-hop: ipv4
             metric: 1–65535
          }
          interface-route: ipv4net {
             next-hop-interface: text
             next-hop-router: ipv4
             metric: 1–65535
          }
          import: text
       }
    }
```

rtrmgr

```
rtrmgr {
   config-directory: text
}
```

service

```
service {
   dhcp-server {
      name text {
         interface: eth0..eth23
         network-mask: 0–32
         start ipv4 {
            stop: ipv4
         }
         exclude: ipv4 {}
         static-mapping: text {
            ip-address: ipv4
            mac-address: macaddr
         }
         dns-server ipv4 {}
         default-router: ipv4
         wins-server ipv4 {}
         lease: 120–4294967296
         domain-name: text
         authoritative: [enable|disable]
      }
   }
   http {
      port: 1–65534
   }
   ssh {
      port: 1–65534
      protocol-version: [v1|v2|all]
   }
   telnet {
      port: 1–65534
   }
   nat {
      rule: 1–1024 {
         type: [source|destination]
         translation-type: [static|dynamic|masquerade]
         inbound-interface: text
         outbound-interface: text
         protocols: [tcp|udp|icmp|all]
         source {
            address: ipv4
            network: ipv4net
            port-number: 1–4294967296 {}
            port-name: [http|ftp|smtp|telnet|ssh|dns|snmp] {}
            port-range {
               start: 1–4294967296
```

```
                stop: 1–4294967296
            }
        }
        destination {
            address: ipv4
            network: ipv4net
            port-number: 1–4294967296 {}
            port-name: [http|ftp|smtp|telnet|ssh|dns|snmp] {}
            port-range {
                start: 1–4294967296
                stop: 1–4294967296
            }
        inside-address {
            address: ipv4
            network: ipv4net
        }
        outside-address {
            address: ipv4
            network: ipv4net
            range {
                start: ipv4
                stop: ipv4
            }
        }
    }
  }
}
```

system

```
system {
   disable: [true | false]
   host-name: text
   domain-name: text
   domain-search {
      domain: text [text ...]
   }
   name-server: ipv4 {}
   time-zone: text
   ntp-server: [ipv4|text] {}
   static-host-mapping {
      host-name: text {
         inet: ipv4
         alias: text {}
      }
   }
   login {
      user text {
         full-name: text
         authentication {
            plaintext-password: text
            encrypted-password: text
         }
      }
      radius-server ipv4 {
         port: 1–65534
         secret: text
         timeout: 1–4294967296
      }
   }
   syslog {
      console {
         facility: text {
            level: text
         }
      }
      file: text {
         facility: text {
            level: text
         }
         archive {
            files: 1–4294967296
            size: 1–4294967296
         }
      }
```

```
            host: text {
                facility: text {
                    level: text
                }
            }
            user: text {
                facility: text {
                    level: text
                }
            }
        }
        package {
            repository: text {
                description: text
                url: text
                component: text
            }
        }
    }
```

vpn

```
vpn {
   ipsec {
      ipsec-interfaces {
         interface int-name {}
      }
      nat-traversal: [enable|disable]
      nat-networks {
         allowed-network ipv4net {
            exclude ipv4net {}
         }
      }
      copy-tos: [enable|disable]
      ike-group text{
         proposal: 1-65535 {
            encryption: [aes128|aes256|3des]
            hash: [sha1|md5]
            dh-group: [2|5]
         }
         lifetime: 30-86400
         aggressive-mode: [enable|disable]
         dead-peer-detection {
            interval: 15-86400
            timeout: 30-86400
            action: [hold|clear|restart]
         }
      }
      esp-group text {
         proposal 1-65535 {
            encryption: [aes128|aes256|3des]
            hash: [sha1|md5]
         }
         mode: [tunnel|transport]
         lifetime: 30-86400
         pfs: [enable|disable]
         compression: [enable|disable]
      }
      site-to-site {
         peer ipv4 {
            authentication {
               mode: [pre-shared-secret|rsa]
               pre-shared-secret: text
               rsa-key-sig: text
            }
            ike-group: text
            local-ip: ipv4
```

```
            tunnel:1-65535 {
               local-subnet: ipv4net
               remote-subnet: ipv4net
               esp-group: text
            }
         }
      }
      rekey-timers {
         rekey-time: 10-86400
         rekey-random: 0-100
      }
      rsa-keys {
         local-key{
            generate
            rsa-key: key-data
         remote-key text {
            rsa-key: key-data
      }
      logging {
         facility: [daemon|local0..local7]
         level: [emerg|crit|err|warning|alert|notice|info|debug]
         esp-log-modes [all|raw|crypt|parsing|emitting|control|
            private] {}
      }
   }
}
```

Glossary

AS	*See* Autonomous System.
Autonomous System	A routing domain that is under one administrative authority, and which implements its own routing policies. A key concept in BGP.
BGP	Border Gateway Protocol.
Bootstrap Router	A PIM-SM router that chooses the RPs for a domain from amongst a set of candidate RPs.
BSR	*See* Bootstrap Router.
Candidate RP	A PIM-SM router that is configured to be a candidate to be an RP. The Bootstrap Router will then choose the RPs from the set of candidates.
Dynamic Route	A route learned from another router via a routing protocol such as RIP or BGP.
EGP	*See* Exterior Gateway Protocol.
Exterior Gateway Protocol	A routing protocol used to route between Autonomous Systems. The main example is BGP.
IGMP	Internet Group Management Protocol. *TBD*
IGP	*See* Interior Gateway Protocol.
Interior Gateway Protocol	A routing protocol used to route within an Autonomous System. Examples include RIP, OSPF and IS-IS.
MLD	Multicast Listener Discovery protocol. TBD
MRIB	*See* Multicast RIB.

Multicast RIB	The part of the RIB that holds multicast routes. These are not directly used for forwarding, but instead are used by multicast routing protocols such as PIM-SM to perform RPF checks when building the multicast distribution tree.
OSPF	Open Shortest Path First. An IGP routing protocol based on a link-state algorithm. Used to route within medium to large networks.
PIM-SM	Protocol Independent Multicast, Sparse-Mode TBD
Rendezvous Point	A router used in PIM-SM as part of the rendezvous process by which new senders are grafted on to the multicast tree.
Reverse Path Forwarding	Many multicast routing protocols such as PIM-SM build a multicast distribution tree based on the best route back from each receiver to the source, hence multicast packets will be forwarded along the reverse of the path to the source.
RIB	*See* Routing Information Base.
RIP	Routing Information Protocol. *TBD*
Routing Information Base	The collection of routes learned from all the dynamic routing protocols running on the router. Subdivided into a Unicast RIB for unicast routes and a Multicast RIB.
RP	*See* Rendezvous Point.
RPF	*See* Reverse Path Forwarding.
Static Route	A route that has been manually configured on the router.
xorpsh	XORP command shell.
xorp rtrmgr	XORP router manager process.