RIPng Configuration Guide, 17.2.0

November 2017

Supporting AT&T Vyatta Network Operating System

Contents

Copyright Statement

About This Guide

This guide describes how to configure Routing Information Protocol next generation (RIPng) on the AT&T Vyatta vRouter (referred to as a virtual router, vRouter, or router in the guide).

Router-level Configuration Commands

monitor protocol ripng disable events

Disables the generation of debug messages that are related to RIPng events.

Syntax:

```
monitor protocol ripng disable events
```

Operational mode

Use this command to disable the generation of debug (trace-level) messages that are related to RIPng events.

monitor protocol ripng disable packet

Disables the generation of debug messages that are related to all RIPng packet types.

Syntax:

```
monitor protocol ripng disable packet [ recv | send ]
```

`recv`
: Disables debugging of all received packets.

`send`
: Disables debugging of all sent packets.

Operational mode

Use this command to disable the generation of debug (trace-level) messages that are related to RIPng packet types.

monitor protocol ripng disable rib

Disables the generation of debug messages that are related to the RIPng RIB.

Syntax:

```
monitor protocol ripng disable rib
```

Debug messages are disabled for actions that are related to the RIPng RIB.

Operational mode

Use this command to disable the generation of debug (trace-level) messages that are related to the RIPng RIB.

monitor protocol ripng enable events

Enables the generation of debug messages that are related to RIPng events.

Syntax:

```
monitor protocol ripng enable events
```

Operational mode

Use this command to enable the generation of debug (trace-level) messages that are related to RIPng events.

monitor protocol ripng enable packet

Enables the generation of debug messages that are related to all RIPng packet types.

Syntax:

`monitor protocol ripng enable packet` [**`recv`** | **`send`**]

`recv`
: Enables debugging of all received packets.

`send`
: Enables debugging of all sent packets.

Operational mode

Use this command to enable the generation of debug (trace-level) messages that are related to all RIPng packet types.

monitor protocol ripng enable rib

Enables the generation of debug messages that are related to the RIPng RIB.

Syntax:
`monitor protocol ripng enable rib`

Debug messages are generated for actions that are related to the RIPng RIB.

Operational mode

Use this command to enable the generation of debug (trace-level) messages that are related to the RIPng RIB.

protocols ripng aggregate-address <ipv6net>

Specifies an aggregate RIPng route announcement.

Syntax:
`set protocols ripng aggregate-address` *ipv6net*

Syntax:
`delete protocols ripng aggregate-address` *ipv6net*

Syntax:
`show protocols ripng aggregate-address` [*ipv6net*]

ipv6net
: An IPv6 network from which routes are to aggregate. The format is *ipv6-address/ prefix*.

Configuration mode

```
protocols {
 ripng {
  aggregate-address ipv6net
 }
}
```

Use this command for IPv6 address aggregation.

Use the `set` form of this command to specify a contiguous block of IPv6 addresses to aggregate.

Use the `delete` form of this command to delete an aggregate address.

Use the `show` form of this command to display aggregate address configuration settings.

protocols ripng default-information originate

Generates a default route into the RIPng routing domain.

Syntax:
`set protocols ripng default-information originate`

Syntax:
`delete protocols ripng default-information originate`

Syntax:
`show protocols ripng default-information originate`

A default route into the RIPng routing domain is not generated.

Configuration mode

```
protocols {
 ripng {
  default-information {
   originate
  }
 }
}
```

Use the `set` form of this command to generate a default route into the RIPng routing domain.

Use the `delete` form of this command to restore the default behavior for default route generation into RIPng; that is, a default route is not generated.

Use the `show` form of this command to display the default configuration of route generation into RIPng.

protocols ripng default-metric <metric>

Sets the default metric for external routes that are redistributed into RIPng.

Syntax:
`set protocols ripng default-metric` *metric*

Syntax:
`delete protocols ripng default-metric`

Syntax:
`show protocols ripng default-metric`

Routes that are imported into RIPng are assigned a metric of 1.

metric
: Mandatory. A metric assigned to external routes that are imported into RIPng. The metric ranges from 1 through 16. The default metric is 1.

Configuration mode

```
protocols {
 ripng {
  default-metric metric
 }
}
```

Use the `set` form of this command to set the default metric for external routes that are redistributed into RIPng.

Use the `delete` form of this command to restore the default RIPng metric for external routes that are redistributed into RIPng; that is, routes are assigned a metric of 1.

Use the `show` form of this command to display the default metric for external routes that are redistributed into RIPng.

protocols ripng log

Enables logging for RIPng.

Syntax:
```
set protocols ripng log { all | events | nsm | packet | rib }
```

Syntax:
```
delete protocols ripng log { all | events | nsm | packet | rib }
```

Syntax:
```
show protocols ripng log { all | events | nsm | packet | rib }
```

None

`all`
: Enables all RIPng logs.

`events`
: Enables RIPng events logs.

`nsm`
: Enables RIPng NSM logs.

`packet`
: Enables RIPng packet logs.

`rib`
: Enables RIPng RIB logs.

Configuration mode

```
protocols {
        ripng {
                log {
                        all
                        events
                        nsm
                        packet
                        rib
                }
        }
}
```

Use the `set` form of this command to enable routing information protocol (RIP)ng logs.

Use the `delete` form of this command to remove RIPng logs.

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

protocols ripng log packet

Enables logging for RIPng packets.

Syntax:
```
set protocols ripng log packet { all | detail | rcv | send }
```

Syntax:
```
delete protocols ripng log packet { all | detail | rcv | send }
```

Syntax:
```
show protocols ripng log packet { all | detail | rcv | send }
```

None

all
: Enables all RIPng packet logs.

detail
: Enables only RIPng packet detail logs.

rcv
: Enables only RIPng packet receive logs.

send
: Enables only RIPng packet send logs.

Configuration mode

```
protocols {
       ripng {
               log {
                       packet {
                               all
                               detail
                               rcv
                               send
                       }
               }
       }
}
```

Use the `set` form of this command to enable routing information protocol (RIP)ng packet logs.

Use the `delete` form of this command to remove RIPng packet logs.

Use the `show` form of this command to view RIPng packet logs.

protocols ripng passive-interface <interface-name>

Suppresses updates to RIPng routing on an interface.

Syntax:
`set protocols ripng passive-interface` *interface-name*

Syntax:
`delete protocols ripng passive-interface` *interface-name*

Syntax:
`show protocols ripng passive-interface`

RIPng routing updates are not suppressed.

interface-name
: The identifier of an interface. Supported interface types are:

- Data plane
- Loopback

For more information about these interface types, refer to Supported Interface Types *(page 32)*.

You can suppress routing updates on more than one RIPng interface by creating multiple **`protocols ripng passive-interface`** configuration nodes.

Configuration mode

```
protocols {
 ripng {
  passive-interface interface
 }
}
```

Use the `set` form of this command to suppress updates to RIPng routing on an interface.

Use the `delete` form of this command to disable the suppression of updates to RIPng routing on an interface.

Use the `show` form of this command to display the configuration of RIPng route suppression for an interface.

protocols ripng route <ipv6net>

Sets a static route in RIPng.

Syntax:
`set protocols ripng route` *ipv6net*

Syntax:
`delete protocols ripng route` *ipv6net*

Syntax:
`show protocols ripng route`

ipv6net
Mandatory. The IPv6 network address defining the RIPng static route.

Configuration mode

```
protocols {
 ripng {
  route ipv6net
 }
}
```

Use this command to set a static route in RIPng.

Use the `set` form of this command to set a static route in RIPng.

Use the `delete` form of this command to remove an RIPng static route.

Use the `show` form of this command to display RIPng static route configuration.

protocols ripng timers garbage-collection <seconds>

Sets the timer for RIPng garbage collection.

Syntax:
`set protocols ripng timers garbage-collection` *seconds*

Syntax:
`delete protocols ripng timers garbage-collection` [*seconds*]

Syntax:
`show protocols ripng timers garbage-collection`

RIPng garbage collection occurs at 120 seconds.

seconds
Mandatory. A timer interval in seconds. The interval ranges from 0 through 65535. The default interval is 120.

Configuration mode

```
protocols {
 ripng {
  timers {
   garbage-collection seconds
  }
 }
}
```

Use the `set` form of this command to set the timer for RIPng garbage collection. When the timer expires, the system scans for stale RIPng resources and releases them for use.

Use the `delete` form of this command to restore the default timer interval for RIPng garbage collection, which is 120 seconds.

Use the `show` form of this command to display the current timer interval for RIPng garbage collection.

protocols ripng timers timeout <seconds>

Sets the interval for RIPng timeouts.

Syntax:
`set protocols ripng timers timeout` *seconds*

Syntax:
`delete protocols ripng timers timeout` [*seconds*]

Syntax:
`show protocols ripng timers timeout`

RIPng timeouts occur at 180 seconds.

seconds
Mandatory. A timer interval in seconds. The interval ranges from 0 through 65535. The default interval is 180.

Configuration mode

```
protocols {
 ripng {
  timers {
   timeout seconds
  }
 }
}
```

Use the `set` form of this command to set the interval for RIPng timeouts.

Use the `delete` form of this command to restore the default interval for RIPng time-outs, which is 180 seconds.

Use the `show` form of this command to display the current interval for RIPng time-outs.

protocols ripng timers update <seconds>

Sets the timer interval for updates to the RIPng routing table.

Syntax:
`set protocols ripng timers update` *seconds*

Syntax:
`delete protocols ripng timers update` [*seconds*]

Syntax:
`show protocols ripng timers update`

The RIPng routing table is updated every 30 seconds.

seconds

Mandatory. An interval, in seconds, at which updates to the RIPng routing table occur. The interval ranges from 0 through 65535. The default interval is 30.

Configuration mode

```
protocols {
 ripng {
  timers {
   update seconds
  }
 }
}
```

Use the `set` form of this command to set the timer interval for updates to the RIPng routing table. When the interval is shorter, the routing information in the tables is more accurate; however, more protocol network traffic occurs.

Use the `delete` form of this command to restore the default interval for RIPng updates, which is 30 seconds.

Use the `show` form of this command to display the current interval for RIPng updates.

reset ipv6 ripng route

Resets data in the RIPng routing table.

Syntax:

`reset ipv6 ripng route` [**`all`** | **`bgp`** | **`connected`** | **`kernel`** | **`ospfv6`** | **`ripng`** | **`static`** | *ip-address*]

`all`

Removes all entries from the RIPng routing table.

`bgp`

Removes only BGP routes from the RIPng routing table.

`connected`

Removes entries for connected routes from the RIPng routing table.

`kernel`

Removes kernel entries from the RIPng routing table.

`ospfv6`

Removes only OSPFv6 routes from the RIPng routing table.

`ripng`

Removes only RIPng routes from the RIPng routing table.

`static`

Removes static entries from the RIPng routing table.

ip-address

Removes entries that match *`ip-address`* (*`x:x::x:x/M`*), a destination IPv6 address, from the RIPng routing table.

Operational mode.

Use the `reset ipv6 ripng route all` command to clear the RIPng routing table.

show ipv6 route ripng

Displays all IPv6 RIPng routes.

Syntax:

`show ipv6 route ripng`

Operational mode

Use this command to display all RIPng routes that are contained in the RIB.

The following example shows all RIPng routes from the RIB.

```
vyatta@vyatta:~$show ipv6 route ripng
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv3,
       I - ISIS, B - BGP, * - FIB route.

R>* 2001:db8:2::/64 [120/2] via fe80::20c:29ff:fed6:816c, dp0s1, 00:43:00
R>* 2001:db8:3::/64 [120/3] via fe80::20c:29ff:fed6:816c, dp0s1, 00:00:03
vyatta@vyatta:~$
```

show ipv6 ripng

Displays information about RIPng.

Syntax:
`show ipv6 ripng [ interface | status ]`

Displays all information about RIPng.

`interface`
Optional. Displays information for RIPng interfaces.

`status`
Optional. Displays only RIPng protocol status information.

Operational mode

Use this command to display information about RIPng.

The following example lists RIPng information.

```
vyatta@vyatta:~$ show ipv6 ripng
Codes: R - RIPng, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface, (a/S) - aggregated/Suppressed

   Network      Next Hop                      Via     Metric Tag Time
C(i) 2001:db8:1::/64
                  ::                            self       1    0
R(n) 2001:db8:2::/64
                  fe80::20c:29ff:fed6:816c    dp0s1      2    0  02:56
R(n) 2001:db8:3::/64
                  fe80::20c:29ff:fed6:816c    dp0s1      3    0  02:56
vyatta@vyatta:~$
```

The following example lists RIPng protocol status information.

```
vyatta@vyatta:~$ show ipv6 ripng status
  Routing Protocol is "RIPng"
  Sending updates every 30 seconds with +/-50%, next due in 4 seconds
  Timeout after 180 seconds, garbage collect after 120 seconds
  Outgoing update filter list for all interface is not set
  Incoming update filter list for all interface is not set
  Default redistribution metric is 1
  Redistributing:
      Interface
      dp0s1
```

show monitoring protocols ripng

Displays RIPng protocol debugging flags.

Syntax:
`show monitoring protocols ripng`

Operational mode

Use this command to display how debugging is set for RIPng.

RIPng Configuration

RIPng overview

RIPng is a dynamic routing protocol that is suitable for small, homogenous IPv6 networks. It is classified as an interior gateway protocol (IGP) and employs the distance-vector routing algorithm. RIPng determines the best path by counting the hops to the destination. The maximum hop count is 15 (16 is considered an infinite distance), making RIPng less suitable for large networks. RIPng is an extension of RIP version 2 for IPv6.

Supported standards

The AT&T Vyatta vRouter implementation of RIPng complies with the following standards:

- RFC 2080: RIPng for IPv6
- RFC 2081: RIPng Protocol Applicability Statement

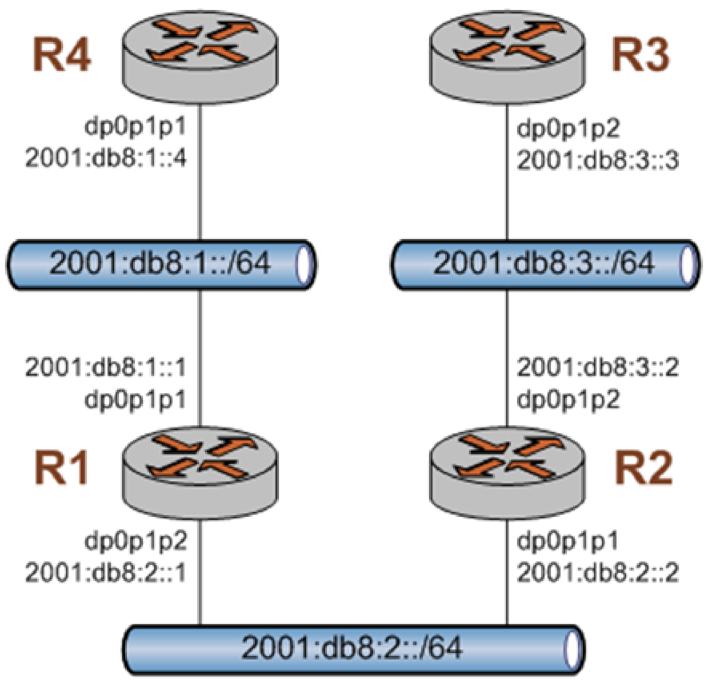
Configuring RIPng

This section presents the following topics:

- Enable forwarding on R1 and R2
- Enable RIPng on an interface
- Advertise connected networks
- Confirm visibility of remote networks

This section presents an example configuration of RIPng. The configuration example is based on the reference diagram in the following figure. This example shows the configuration of the nodes by using dynamic IPv6 routing with RIPng to enable R3 and R4 to communicate through R1 and R2.

Figure 1: Dynamic IPv6 routing example in RIPng



Enabling forwarding on R1 and R2

For R1 to pass data between the dp0p1p1 and dp0p1p3 interfaces and R2 to pass data between the dp0p1p1 and dp0p1p2 interfaces, R1 and R2 must be configured to enable forwarding. To enable forwarding on R1, perform the following step in configuration mode.

Table 1: Enabling forwarding on R1

Step	Command
Enable forwarding on R1.	`vyatta@R1# delete system ipv6 disable-forwarding`
Commit the change.	`vyatta@R1# commit`

To enable forwarding on R2, perform the following steps in configuration mode.

Table 2: Enabling forwarding on R2

Step	Command
Enable forwarding on R2.	`vyatta@R2# delete system ipv6 disable-forwarding`
Commit the change.	`vyatta@R2# commit`

Enabling RIPng on an interface

To allow dynamic routing by using RIPng, RIPng must be enabled on the interfaces that are to use it. To enable RIPng on R1, perform the following steps in configuration mode.

Table 3: Enable RIPng on R1

Step	Command
Enable RIPng on dp0p1p1.	`vyatta@R1# set interfaces dataplane dp0p1p1 ipv6 ripng enable`
Enable RIPng on dp0p1p3.	`vyatta@R1# set interfaces dataplane dp0p1p3 ipv6 ripng enable`
Commit the change.	`vyatta@R1# commit`

Step	Command

Change to operational mode.	`vyatta@R1# exit`
	`vyatta@R1:~$`

Verify the status of RIPng.	`vyatta@R1:~$ show ipv6 ripng status`
	`  Routing Protocol is "RIPng"`
	`    Sending updates every 30 seconds with`
	` +/-50%, next due in 4 seconds`
	`    Timeout after 180 seconds, garbage`
	` collect after 120 seconds`
	`    Outgoing update filter list for all`
	` interface is not set`
	`    Incoming update filter list for all`
	` interface is not set`
	`    Default redistribution metric is 1`
	`    Redistributing:`
	`  Interface`
	`      dp0p1p1`
	`      dp0p1p2`

Display information for RIPng interfaces.	`vyatta@R1:~$ show ipv6 ripng interface`
	`dp0p1p1 is up, line protocol is up`
	`  Routing Protocol: RIPng`
	`    Passive interface: Disabled`
	`    Split horizon: Enabled with Poisoned`
	` Reversed`
	`    IPv6 interface address:`
	`      fe80::5054:ff:fe8b:1/64`
	`dp0p1p2 is up, line protocol is up`
	`  Routing Protocol: RIPng`
	`    Passive interface: Disabled`
	`    Split horizon: Enabled with Poisoned`
	` Reversed`
	`    IPv6 interface address:`
	`      fe80::5054:ff:fe98:2/64`

Advertising connected networks

The `redistribute` command is then used to advertise the connected networks. To advertise connected networks on R1, perform the following steps in configuration mode.

Table 4: Advertising connected networks on R1

Step	Command

Advertise connected networks through RIPng.	`vyatta@R1# set protocols ripng redistribute`
	` connected`

Commit the change.	`vyatta@R1# commit`

Step	Command

Verify the redistribution.	
	```
	vyatta@R1:~$ show ipv6 ripng status
	Routing Protocol is "RIPng"
	Sending updates every 30 seconds with +/-50%,
	 next due in
	4 seconds
	Timeout after 180 seconds, garbage collect
	 after 120 seconds
	Outgoing update filter list for all interface
	 is not set
	Incoming update filter list for all interface
	 is not set
	Default redistribution metric is 1
	Redistributing: connected
	Interface
	dp0p1p1
	dp0p1p2
	```

Confirming visibility of remote networks

After enabling RIPng on the other interfaces of R2, R3, and R4 and advertising connected networks on R2, check the routing table of R4 to verify that it has learned the network. To confirm visibility of remote networks on R4, perform the following step in operational mode.

Table 5: Confirming visibility of remote networks on R4

Step	Command

Trace the route from R2 to R4.	
	```
	vyatta@R4:~$ show ipv6 route
	IPv6 Routing Table
	Codes: K - kernel route, C - connected, S -
	 static, R - RIP, O - OSPF,
	       IA - OSPF inter area, E1 - OSPF
	 external type 1,
	       E2 - OSPF external type 2, I - IS-IS,
	 B - BGP
	       > - selected route, * - FIB route, p -
	 stale info
	Timers: Uptime
	S>* ::/0 [1/0] via 2001:db8:1::1, dp0s1
	C>* ::1/128 is directly connected, lo
	C>* 2001:db8:1::/64 is directly connected,
	 dp0s1
	R>* 2001:db8:2::/64 [120/2] via
	 fe80::20c:29ff:fed6:816c,
	dp0s1, 00:43:00
	R>* 2001:db8:3::/64 [120/3] via
	 fe80::20c:29ff:fed6:816c,
	dp0s1, 00:00:03
	C>* fe80::/64 is directly connected, dp0s1
	```

The R in the first column indicates that two routes have been learned from RIPng. Because a route now exists for 2001:db8:3::/64, R3 can be pinged. To confirm connectivity, perform the following steps in operational mode.

Table 6: Confirming connectivity between R4 and R3

Step	Command

Ping R3 from R4.	
	```
	vyatta@R4:~$ ping 2001:db8:3::3
	PING 2001:db8:3::3(2001:db8:3::3) 56 data
	 bytes
	64 bytes from 2001:db8:3::3: icmp_seq=1
	 ttl=62 time=5.98 ms
	64 bytes from 2001:db8:3::3: icmp_seq=2
	 ttl=62 time=0.603 ms
	^C
	--- 2001:db8:3::3 ping statistics ---
	2 packets transmitted, 2 received, 0% packet
	 loss, time 1011ms
	rtt min/avg/max/mdev =
	 0.603/3.294/5.986/2.692 ms
	```

Display the RIPng status.	
	```
	vyatta@R4:~$ show ipv6 ripng
	Codes: R - RIP, Rc - RIP connected, Rs - RIP
	 static, Ra - RIP aggregated,
	       Rcx - RIP connect suppressed, Rsx -
	 RIP static suppressed,
	       K - Kernel, C - Connected, S - Static,
	 O - OSPF, I - IS-IS, B - BGP
	   Network                     Next Hop
	            If      Met Tag  Time
	C  2001:db8:1::/64            ::
	           dp0p1p1   1   0
	R  2001:db8:2::/64            2001:db8:1::1
	                      dp0p1p1   1   0
	R  2001:db8:3::/64            2001:db8:1::1
	                      dp0p1p1   1   0
	```

Route Redistribution Commands

protocols ripng redistribute bgp

Redistributes BGP routes into RIPng routing tables.

Syntax:
`set protocols ripng redistribute bgp [ ` **`metric`** *metric* `|` **`route-map`** *map-name* `]`

Syntax:
`delete protocols ripng redistribute bgp [ ` **`metric`** `|` **`route-map`** `]`

Syntax:
`show protocols ripng redistribute bgp [ ` **`metric`** `|` **`route-map`** `]`

BGP routes that are redistributed into RIPng are assigned a routing metric of 1. By default, no route map is applied to redistributed BGP routes.

metric
Applies a metric to BGP routes that are imported into RIPng routing tables. The metric ranges from 1 through 16. The default metric is 1.

`route-map` ***map-name***
Applies a route map to BGP routes that are imported into RIPng routing tables.

Configuration mode

```
protocols {
 ripng {
  redistribute {
   bgp {
    metric metric
    route-map map-name
   }
  }
 }
}
```

Use the `set` form of this command to redistribute BGP routes into RIPng routing tables. You can set the routing metric for or specify a route map to apply to redistributed BGP routes.

Use the `delete` form of this command to remove the current configuration of BGP route redistribution.

Use the `show` form of this command to display the current configuration of BGP route redistribution.

protocols ripng redistribute connected

Redistributes directly connected routes into RIPng routing tables.

Syntax:
`set protocols ripng redistribute connected [ ` **`metric`** *metric* `|` **`route-map`** *map-name* `]`

Syntax:
`delete protocols ripng redistribute connected [ ` **`metric`** `|` **`route-map`** `]`

Syntax:
`show protocols ripng redistribute connected [ ` **`metric`** `|` **`route-map`** `]`

Connected routes that are redistributed into RIPng are assigned a routing metric of 1. By default, no route map is applied to redistributed connected routes.

metric
: Optional. The routing metric to be applied to connected routes being imported into RIPng routing tables. The range is 1 to 16. The default is 1.

map-name
: Optional. Applies the specified route map to connected routes being imported into RIPng routing tables.

Configuration mode

```
protocols {
 ripng {
  redistribute {
   connected {
    metric metric
    route-map map-name
   }
  }
 }
}
```

Use the `set` form of this command to redistribute directly connected routes into RIPng routing tables. You can set the routing metric for or specify a route map to apply to directly connected BGP routes.

Use the `delete` form of this command to remove the current configuration of directly connected route redistribution.

Use the `show` form of this command to display the current configuration of directly connected route redistribution.

protocols ripng redistribute kernel

Redistributes kernel routes into RIPng routing tables.

Syntax:
`set protocols ripng redistribute kernel` [**`metric`** *metric* | **`route-map`** *map-name*]

Syntax:
`delete protocols ripng redistribute kernel` [**`metric`** | **`route-map`**]

Syntax:
`show protocols ripng redistribute kernel` [**`metric`** | **`route-map`**]

Kernel routes that are redistributed into RIPng are assigned a routing metric of 1. By default, no route map is applied to redistributed kernel routes.

metric
: Optional. The routing metric to be applied to kernel routes being imported into RIPng routing tables. The range is 1 to 16. The default is 1.

map-name
: Optional. Applies the specified route map to kernel routes being imported into RIPng routing tables.

Configuration mode

```
protocols {
 ripng {
  redistribute {
   kernel {
    metric metric
    route-map map-name
```

```
      }
    }
  }
}
```

Use the `set` form of this command to redistribute kernel routes into RIPng routing tables. You can set the routing metric for or specify a route map to apply to redistributed kernel routes.

Use the `delete` form of this command to remove the current configuration of kernel route redistribution.

Use the `show` form of this command to display the current configuration of kernel route redistribution.

protocols ripng redistribute ospfv3

Redistributes OSPFv3 routes into RIPng routing tables.

Syntax:
`set protocols ripng redistribute ospfv3` [**`metric`** *metric* | **`route-map`** *map-name*]

Syntax:
`delete protocols ripng redistribute ospfv3` [**`metric`** | **`route-map`**]

Syntax:
`show protocols ripng redistribute ospfv3` [**`metric`** | **`route-map`**]

OSPFv3 routes that are redistributed into RIPng are assigned a routing metric of 1. By default, no route map is applied to redistributed OSPFv3 routes.

metric
Optional. The routing metric to be applied to OSPFv3 routes being imported into RIPng routing tables. The range is 1 to 16. The default is 1.

map-name
Optional. Applies the specified route map to OSPFv3 routes being imported into RIPng routing tables.

Configuration mode

```
protocols {
  ripng {
    redistribute {
      ospfv3 {
        metric metric
        route-map map-name
      }
    }
  }
}
```

Use the `set` form of this command to redistribute OSPFv3 routes into RIPng routing tables. You can set the routing metric for or specify a route map to apply to redistributed OSPFv3 routes.

Use the `delete` form of this command to remove the current configuration of OSPFv3 route redistribution.

Use the `show` form of this command to display the current configuration of OSPFv3 route redistribution.

protocols ripng redistribute static

Redistributes static routes into RIPng routing tables.

Syntax:
`set protocols ripng redistribute static` [**`metric`** *metric* | **`route-map`** *map-name*]

Syntax:

`delete protocols ripng redistribute static [ metric | route-map ]`

Syntax:
`show protocols ripng redistribute static [ metric | route-map ]`

Static routes that are redistributed into RIPng are assigned a routing metric of 1. By default, no route map is applied to redistributed static routes.

metric
Optional. The routing metric to be applied to static routes being imported into RIPng routing tables. The range is 1 to 16. The default is 1.

map-name
Optional. Applies the specified route map to static routes being imported into RIPng routing tables.

Configuration mode

```
protocols {
 ripng {
  redistribute {
   static {
    metric metric
    route-map map-name
   }
  }
 }
}
```

Use the `set` form of this command to redistribute static routes into RIPng routing tables. You can set the routing metric for or specify a route map to apply to redistributed static routes.

Use the `delete` form of this command to remove the current configuration of static route redistribution.

Use the `show` form of this command to display the current configuration of static route redistribution.

Route Filtering Commands

protocols ripng distribute-list access-list

Applies an access list to filter inbound or outbound RIPng packets.

Syntax:
`set protocols ripng distribute-list access-list` { **in** *in-list* | **out** *out-list* }

Syntax:
`delete protocols ripng distribute-list access-list` { **in** | **out** }

Syntax:
`show protocols ripng distribute-list access-list` { **in** | **out** }

in-list
Specifies the identifier of a defined access list. The access list filters inbound RIPng packets.

out-list
Specifies the identifier of a defined access list. The access list filters outbound RIPng packets.

Configuration mode

```
protocols {
 ripng {
  distribute-list {
   access-list {
    in in-list
    out out-list
   }
  }
 }
}
```

Use the `set` form of this command to apply an access list to filter inbound or outbound RIPng packets.

Use the `delete` form of this command to remove the filtering of RIPng inbound or outbound packets by an access list.

Use the `show` form of this command to display RIPng access list filtering configuration.

protocols ripng distribute-list interface <interface-name> access-list

Applies an access list to an interface to filter inbound or outbound RIPng packets.

Syntax:
`set protocols ripng distribute-list interface` *interface-name* **access-list** { **in** *in-list* | **out** *out-list* }

Syntax:
`delete protocols ripng distribute-list interface` *interface-name* **access-list** { **in** | **out** }

Syntax:
`show protocols ripng distribute-list interface` *interface-name* **access-list** { **in** | **out** }

interface-name
The identifier of an interface. Supported interface types are:

- Data plane
- Loopback

For more information about these interface types, refer to Supported Interface Types *(page 32)*.

`in` ***in-list***
Specifies the identifier of a defined access list. The access list applies to the specified interface to filter inbound RIPng packets.

`out` ***out-list***
Specifies the identifier of a defined access list. The access list applies to the specified interface to filter outbound RIPng packets.

Configuration mode

```
protocols {
 ripng {
  distribute-list {
   interface interface-name {
    access-list {
     in in-list
     out out-list
    }
   }
  }
 }
}
```

Use the `set` form of this command to apply an access list to a specific interface to filter inbound or outbound RIPng packets.

Use the `delete` form of this command to remove the filtering of RIPng inbound or outbound packets on an interface by an access list.

Use the `show` form of this command to display RIPng access list filtering configuration for an interface.

protocols ripng distribute-list interface <interface-name> prefix-list

Applies a prefix list to an interface to filter inbound or outbound RIPng packets.

Syntax:
`set protocols ripng distribute-list interface` *interface-name* **`prefix-list`** { **`in`** *in-list* | **`out`** *out-list* }

Syntax:
`delete protocols ripng distribute-list interface` *interface-name* **`prefix-list`** { **`in`** | **`out`** }

Syntax:
`show protocols ripng distribute-list interface` *interface-name* **`prefix-list`** { **`in`** | **`out`** }

interface-name
The identifier of an interface. Supported interface types are:

- Data plane
- Loopback

For more information about these interface types, refer to Supported Interface Types *(page 32)*.

`in` ***in-list***
Specifies the identifier of a defined prefix list. The prefix list applies to the specified interface to filter inbound RIPng packets.

`out` ***out-list***

Specifies the identifier of a defined prefix list. The prefix list applies to the specified interface to filter outbound RIPng packets.

Configuration mode

```
protocols {
 ripng {
  distribute-list {
   interface interface-name {
    prefix-list {
     in in-list
     out out-list
    }
   }
  }
 }
}
```

Use the `set` form of this command to apply a prefix list to an interface to filter inbound or outbound RIPng packets.

Use the `delete` form of this command to remove the filtering of RIPng inbound or outbound packets on an interface by a prefix list.

Use the `show` form of this command to display RIPng prefix list filtering configuration for an interface.

protocols ripng distribute-list prefix-list

Applies a prefix list to filter inbound or outbound RIPng packets.

Syntax:
`set protocols ripng distribute-list prefix-list` { **`in`** *in-list* | **`out`** *out-list* }

Syntax:
`delete protocols ripng distribute-list prefix-list` { **`in`** | **`out`** }

Syntax:
`show protocols ripng distribute-list prefix-list` { **`in`** | **`out`** }

`in` ***in-list***
Specifies the identifier of a defined prefix list. The prefix list filters inbound RIPng packets.

`out` ***out-list***
Specifies the identifier of a defined prefix list. The prefix list filters outbound RIPng packets.

Configuration mode

```
protocols {
 ripng {
  distribute-list {
   prefix-list {
    in in-list
    out out-list
   }
  }
 }
}
```

Use the `set` form of this command to apply a prefix list to filter inbound or outbound RIPng packets.

Use the `delete` form of this command to remove the filtering of RIPng inbound or outbound packets by a prefix list.

Use the `show` form of this command to display RIPng prefix list filtering configuration.

RIPng Interface Commands

interfaces <interface> ipv6 ripng enable

Enables RIPng on an interface.

Syntax:
`set interfaces` *interface* **`ipv6 ripng enable`**

Syntax:
`delete interfaces` *interface* **`ipv6 ripng enable`**

Syntax:
`show interfaces` *interface* **`ipv6 ripng`**

interface
Mandatory. A type of interface. For detailed keywords and arguments that can be specified as interface types, refer to Supported Interface Types *(page 32)*.

Configuration mode

```
interfaces interface {
 ipv6 {
  ripng
 }
}
```

Use this command to enable RIPng.

Use the `set` form of this command to enable RIPng on an interface.

Use the `delete` form of this command to remove all RIPng configuration and disable RIPng on the interface.

Use the `show` form of this command to display the current RIPng configuration on an interface.

interfaces <interface> ipv6 ripng metric-offset

Sets a metric to add to routes that are received from RIPng on an interface.

Syntax:
`set interfaces` *interface* **`ipv6 ripng metric-offset`** *metric*

Syntax:
`show interfaces` *interface* **`ipv6 ripng metric-offset`**

interface
Mandatory. A type of interface. For detailed keywords and arguments that can be specified as interface types, refer to Supported Interface Types *(page 32)*.

metric
Mandatory. A metric to be added to the routes over the interface. The metric ranges from 1 through 16.

Configuration mode

```
interfaces interface {
 ipv6 {
  ripng {
   metric-offset metric
  }
 }
}
```

Use this command to set the metric for inbound and outbound routes on an interface that are beyond the normal operation of RIPng.

Use the `set` form of this command to set a metric to add to routes that are received from RIPng on an interface.

Use the `show` form of this command to display the current metric that is added to routes that are received from RIPng on an interface.

interfaces <interface> ipv6 ripng split-horizon

Configures split-horizon and split-horizon poison-reverse on an interface that is running RIPng.

Syntax:
`set interfaces` *interface* **`ipv6 ripng split-horizon`** [**`disable`** | **`poison-reverse`**]

Syntax:
`show interfaces` *interface* **`ipv6 ripng split-horizon`**

Split-horizon is enabled.

interface
Mandatory. A type of interface. For detailed keywords and arguments that can be specified as interface types, refer to Supported Interface Types *(page 32)*.

`disable`
Disables split-horizon on the specified interface.

`poison-reverse`
Enables poison-reverse on the specified interface.

Configuration mode

```
interfaces interface {
 ipv6 {
  ripng {
   split-horizon {
    disable
    poison-reverse
   }
  }
 }
}
```

Use this command to disable split-horizon or enable split-horizon poison-reverse on an interface that is running RIPng.

Split-horizon is a stability feature that reduces the possibility of network loops, particularly when links become disconnected. It stops an interface from including in its network updates to any routes that it learned from that interface. Split-horizon is effective at preventing loops between routers that are directly connected to each other, and it speeds convergence when network conditions change. Split-horizon is the default setting in RIPng.

Poison-reverse is a variation of split-horizon. When an interface that has poison-reverse enabled detects a link that is down, it increases the metric for that route to 16 and propagates that information in its next update. Because 15 is the largest number of hops that can be reached on a RIPng network, increasing the metric to 16 renders the route unreachable as far as downstream RIPng routers are concerned. This is called “poisoning” the route. Poison-reverse is useful for propagating information about bad routes to routers that are downstream but not immediate neighbors, where split-horizon is ineffective.

When this option is enabled, the router includes the route in announcements to the neighbor from which it was learned. When this option is disabled, the router omits the route from announcements to the neighbor from which it was learned.

Use the `set` form of this command to configure split-horizon and split-horizon poison-reverse on an interface that is running RIPng.

Use the `show` form of this command to display the current configuration of split-horizon.

interfaces <interface> ipv6 ripng neighbor <ip-address>

Configures the IPv6 link-local address of a neighbor for RIPng.

Syntax:
`set interfaces` *interface* **`ipv6 ripng neighbor`** *ip-address*

Syntax:
`show interface` *interface* **`ipv6 ripng neighbor`**

interface
A type of interface. For detailed keywords and arguments that can be specified as interface types, refer to Supported Interface Types *(page 32)*.

ip-address
The IPv6 link-local address of a neighbor.

Configuration mode.

```
interfaces interface {
         address address {
                ipv6 {
                       ripng {
                             neighbor ip-address
                       }
                }
         }
}
```

Use the `set` form of this command to configure the IPv6 link-local address of a neighbor for RIPng.

Use the `show` form of this command to display the IPv6 link-local address of the neighbor.

Supported Interface Types

The following table shows the syntax and parameters of supported interface types. Depending on the command, some of these types may not apply.

Interface Type	Syntax	Parameters
Bridge	`bridge` *brx*	*brx*: The name of a bridge group. The name ranges from br0 through br999.

Interface Type	Syntax	Parameters
Data plane	`dataplane` *interface-name*	*interface-name*: The name of a data plane interface. Following are the supported formats of the interface name: • `dpxpypz`—The name of a data plane interface, where — `dpx` specifies the data plane identifier (ID). Currently, only dp0 is supported. — `py` specifies a physical or virtual PCI slot index (for example, p129). — `pz` specifies a port index (for example, p1). For example, dp0p1p2, dp0p160p1, and dp0p192p1. • `dpxemy` —The name of a data plane interface on a LAN-on-motherboard (LOM) device that does not have a PCI slot, where `emy` specifies an embedded network interface number (typically, a small number). For example, dp0em3. • `dpxsy`—The name of a data plane interface in a system in which the BIOS identifies the network interface card to reside in a particular physical or virtual slot *y*, where *y* is typically a small number. For example, for the dp0s2 interface, the BIOS identifies slot 2 in the system to contain this interface. • `dpxPnpypz` —The name of a data plane interface on a device that is installed on a secondary PCI bus, where `Pn` specifies the bus number. You can use this format to name data plane interfaces on large physical devices with multiple PCI buses. For these devices, it is possible to have network interface cards installed on different buses with these cards having the same slot ID. The value of *n* must be an integer greater than 0. For example, dp0P1p162p1 and dp0P2p162p1.

Interface Type	Syntax	Parameters
Data plane vif	`dataplane` *interface-name* `vif` *vif-id* [`vlan` *vlan-id*]	*interface-name*: Refer to the preceding description. *vif-id*: A virtual interface ID. The ID ranges from 1 through 4094. *vlan-id*: The VLAN ID of a virtual interface. The ID ranges from 1 through 4094.
Loopback	`loopback lo` or `loopback lo`*n*	*n*: The name of a loopback interface, where *n* ranges from 1 through 99999.
OpenVPN	`openvpn` *vtunx*	*vtunx*: The identifier of an OpenVPN interface. The identifier ranges from vtun0 through vtun*x*, where *x* is a nonnegative integer.
Tunnel	`tunnel` *tunx* or `tunnel` *tunx* `parameters`	*tunx*: The identifier of a tunnel interface you are defining. The identifier ranges from tun0 through tun*x*, where *x* is a nonnegative integer.
Virtual tunnel	`vti` *vtix*	*vtix*: The identifier of a virtual tunnel interface you are defining. The identifier ranges from vti0 through vti*x*, where *x* is a nonnegative integer. **Note:** Before you can configure a vti interface, you must configure a corresponding vpn. **Note:** This interface does not support IPv6.
VRRP	*parent-interface* `vrrp vrrp-group` *group*	*parent-interface*: The type and identifier of a parent interface; for example, data plane dp0p1p2 or bridge br999. *group*: A VRRP group identifier. The name of a VRRP interface is not specified. The system internally constructs the interface name from the parent interface identifier plus the VRRP group number; for example, dp0p1p2v99. Note that VRRP interfaces support the same feature set as does the parent interface.

VRF Support

VRF support for RIP and RIPng

This section describes VRF support for RIP and RIPng configuration- and operational-mode commands. This section also describes VRF support for monitoring and logging commands.

VRF support for router-mode commands

You can run RIP and RIPng router-mode configuration commands in the context of a routing instance by using the optional **`routing routing-instance`** *instance-name* keywords and variable. The following examples show how to configure RIP and RIPng in the context of the RED routing instance.

```
routing routing-instance RED protocols rip …
routing routing-instance RED protocols ripng …
```

If you do not specify a routing instance, the vRouter applies the configuration to the default routing instance.

Note: An interface belongs to only one routing instance.

VRF support for interface-mode commands

The RIP and RIPng interface-mode configuration commands do not support the **`routing routing-instance`** *instance-name* keywords and variable because these commands run in the context of the routing instance to which the interfaces belong.

```
interfaces <intf_type> <intf_name> ip rip …
interfaces <intf_type> <intf_name> ipv6 ripng …
```

VRF support for operational commands

You can use the optional **`routing-instance`** *instance-name* keyword and variable with the RIP and RIPng operational commands. If you do not use this optional keyword and variable, the commands run in the context of the default routing instance.

```
show ip rip [routing-instance <instance_name>] …
reset ip rip [routing-instance <instance_name>] route …
show ipv6 ripng [routing-instance <instance_name>] …
reset ipv6 ripng [routing-instance <interface_name>] route …
```

VRF support for monitoring and logging commands

You can run the RIP and RIPng monitoring and logging commands in the context of a routing instance with the exception of the commands that enable RIB and NSM logging. If you do not use the **`routing-instance`** *instance-name* keyword and variable, the commands run in the context of the default routing instance.

```
monitor protocol rip [routing-instance <instance_name>]…
[routing routing-instance <instance_name>] protocols rip log …

monitor protocol ripng [routing-instance <instance_name>] …
[routing routing-instance <instance_name>] protocols ripng log …
```

The **`rib`** and **`nsm`** logging options are global options and apply to all routing instances. The **`rib`** and **`nsm`** logging options cannot be enabled or disabled on a routing instance basis. The following commands apply to all routing instances.

```
monitor protocol rip … nsm
monitor protocol rip … rib
protocols rip log nsm
protocols rip log rib
monitor protocol ripng … nsm
monitor protocol ripng … rib
protocols ripng log nsm
protocols ripng log rib
```

The output of the following commands displays routing instance information, if relevant.

```
show monitoring protocols rip
show monitoring protocols ripng
```

Command support for VRF routing instances

VRF allows an AT&T Vyatta vRouter to support multiple routing tables, one for each VRF routing instance. Some commands in this guide support VRF and can be applied to particular routing instances.

Use the guidelines in this section to determine correct syntax when adding VRF routing instances to commands. For more information about VRF, refer to AT&T Vyatta Network Operating System Basic Routing Configuration Guide. This guide includes an overview of VRF, VRF configuration examples, information about VRF-specific features, and a list of commands that support VRF routing instances.

Adding a VRF routing instance to a Configuration mode command

For most Configuration mode commands, specify the VRF routing instance at the beginning of a command. Add the appropriate VRF keywords and variable to follow the initial action (**`set`**, **`show`**, or **`delete`**) and before the other keywords and variables in the command.

Example: Configuration mode example: syslog

The following command configures the syslog logging level for the specified syslog host. The command does not include a VRF routing instance, so the command applies to the default routing instance.

```
vyatta@R1# set system syslog host 10.10.10.1 facility all level debug
vyatta@R1# show system syslog
syslog {
    host 10.10.10.1 {
            facility all {
                    level debug
            }
    }
}
```

The following example shows the same command with the VRF routing instance (GREEN) added. Notice that **`routing routing-instance GREEN`** has been inserted between the basic action (**set** in the example) and the rest of the command. Most Configuration mode commands follow this convention.

```
vyatta@R1# set routing routing-instance GREEN system syslog host 10.10.10.1 facility all
 level debug
vyatta@R1# show routing
routing {
     routing-instance GREEN {
             system {
                     syslog {
                            host 11.12.13.2:514 {
                                     facility all {
                                             level debug
                                     }
                             }
                     }
             }
     }
}
```

Example: Configuration mode example: SNMP

Some features, such as SNMP, are not available on a per-routing instance basis but can be bound to a specific routing instance. For these features, the command syntax is an exception to the convention of specifying the routing instance at the beginning of Configuration mode commands.

The following example shows how to configure the SNMPv1 or SNMPv2c community and context for the RED and BLUE routing instances. The first two commands specify the RED routing instance as the context for community A and BLUE routing instance as the context for community B. The subsequent commands complete the configuration.

For more information about configuring SNMP, refer to AT&T Vyatta Network Operating System Remote Management Configuration Guide.

```
vyatta@R1# set service snmp community commA context RED
vyatta@R1# set service snmp community commB context BLUE
vyatta@R1# set service snmp view all oid 1
vyatta@R1# set service snmp community commA view all
vyatta@R1# set service snmp community commB view all
vyatta@R1# show service snmp community
 community commA {
        context RED
        view all
 }
 community commB {
        context BLUE
        view all
 }
[edit]
vyatta@vyatta#
```

Adding a VRF routing instance to an Operational mode command

The syntax for adding a VRF routing instance to an Operational mode command varies according to the type of command parameters:

- If the command does not have optional parameters, specify the routing instance at the end of the command.
- If the command has optional parameters, specify the routing instance after the required parameters and before the optional parameters.

Example: Operational mode examples without optional parameters

The following command displays dynamic DNS information for the default routing instance.

```
vyatta@vyatta:~$ show dns dynamic status
```

The following command displays the same information for the specified routing instance (GREEN). The command does not have any optional parameters, so the routing instance is specified at the end of the command.

```
vyatta@vyatta:~$ show dns dynamic status routing-instance GREEN
```

Example: Operational mode example with optional parameters

The following command obtains multicast path information for the specified host (10.33.2.5). A routing instance is not specified, so the command applies to the default routing instance.

```
vyatta@vyatta:~$ mtrace 10.33.2.5 detail
```

The following command obtains multicast path information for the specified host (10.33.2.5) and routing instance (GREEN). Notice that the routing instance is specified before the optional **detail** keyword.

```
vyatta@vyatta:~$ mtrace 10.33.2.5 routing-instance GREEN detail
```

Example: Operational mode example output: SNMP

The following SNMP **show** commands display output for routing instances.

```
vyatta@vyatta:~$ show snmp routing-instance
Routing Instance SNMP Agent is Listening on for Incoming Requests:
Routing-Instance            RDID
-----------------           ----
RED                         5

vyatta@vyatta:~$ show snmp community-mapping
SNMPv1/v2c Community/Context Mapping:
Community                   Context
---------                   -------
commA                       'RED'
commB                       'BLUE'
deva                        'default'


vyatta@vyatta:~$ show snmp trap-target
SNMPv1/v2c Trap-targets:
Trap-target                   Port   Routing-Instance Community
-----------                   ----   ---------------- ---------
1.1.1.1                              'RED'            'test'


vyatta@vyatta:~$ show snmp v3 trap-target
SNMPv3 Trap-targets:
Trap-target                   Port   Protocol Auth Priv Type   EngineID              Routing-
Instance User
-----------                   ----   -------- ---- ---- ----   --------
 ---------------- ----
2.2.2.2                       '162'  'udp'    'md5      'infor                       'BLUE'
         'test'
```

List of Acronyms

Acronym	Description
ACL	access control list
ADSL	Asymmetric Digital Subscriber Line
AH	Authentication Header
AMI	Amazon Machine Image
API	Application Programming Interface
AS	autonomous system
ARP	Address Resolution Protocol
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BIOS	Basic Input Output System
BPDU	Bridge Protocol Data Unit
CA	certificate authority
CCMP	AES in counter mode with CBC-MAC
CHAP	Challenge Handshake Authentication Protocol
CLI	command-line interface
DDNS	dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
DLCI	data-link connection identifier
DMI	desktop management interface
DMVPN	dynamic multipoint VPN
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System
DSCP	Differentiated Services Code Point
DSL	Digital Subscriber Line
eBGP	external BGP
EBS	Amazon Elastic Block Storage
EC2	Amazon Elastic Compute Cloud
EGP	Exterior Gateway Protocol
ECMP	equal-cost multipath
ESP	Encapsulating Security Payload
FIB	Forwarding Information Base
FTP	File Transfer Protocol
GRE	Generic Routing Encapsulation
HDLC	High-Level Data Link Control
I/O	Input/Output
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers

Acronym	Description
IGMP	Internet Group Management Protocol
IGP	Interior Gateway Protocol
IPS	Intrusion Protection System
IKE	Internet Key Exchange
IP	Internet Protocol
IPOA	IP over ATM
IPsec	IP Security
IPv4	IP Version 4
IPv6	IP Version 6
ISAKMP	Internet Security Association and Key Management Protocol
ISM	Internet Standard Multicast
ISP	Internet Service Provider
KVM	Kernel-Based Virtual Machine
L2TP	Layer 2 Tunneling Protocol
LACP	Link Aggregation Control Protocol
LAN	local area network
LDAP	Lightweight Directory Access Protocol
LLDP	Link Layer Discovery Protocol
MAC	medium access control
mGRE	multipoint GRE
MIB	Management Information Base
MLD	Multicast Listener Discovery
MLPPP	multilink PPP
MRRU	maximum received reconstructed unit
MTU	maximum transmission unit
NAT	Network Address Translation
NBMA	Non-Broadcast Multi-Access
ND	Neighbor Discovery
NHRP	Next Hop Resolution Protocol
NIC	network interface card
NTP	Network Time Protocol
OSPF	Open Shortest Path First
OSPFv2	OSPF Version 2
OSPFv3	OSPF Version 3
PAM	Pluggable Authentication Module
PAP	Password Authentication Protocol
PAT	Port Address Translation
PCI	peripheral component interconnect
PIM	Protocol Independent Multicast
PIM-DM	PIM Dense Mode
PIM-SM	PIM Sparse Mode
PKI	Public Key Infrastructure
PPP	Point-to-Point Protocol
PPPoA	PPP over ATM

Acronym	Description
PPPoE	PPP over Ethernet
PPTP	Point-to-Point Tunneling Protocol
PTMU	Path Maximum Transfer Unit
PVC	permanent virtual circuit
QoS	quality of service
RADIUS	Remote Authentication Dial-In User Service
RHEL	Red Hat Enterprise Linux
RIB	Routing Information Base
RIP	Routing Information Protocol
RIPng	RIP next generation
RP	Rendezvous Point
RPF	Reverse Path Forwarding
RSA	Rivest, Shamir, and Adleman
Rx	receive
S3	Amazon Simple Storage Service
SLAAC	Stateless Address Auto-Configuration
SNMP	Simple Network Management Protocol
SMTP	Simple Mail Transfer Protocol
SONET	Synchronous Optical Network
SPT	Shortest Path Tree
SSH	Secure Shell
SSID	Service Set Identifier
SSM	Source-Specific Multicast
STP	Spanning Tree Protocol
TACACS+	Terminal Access Controller Access Control System Plus
TBF	Token Bucket Filter
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
ToS	Type of Service
TSS	TCP Maximum Segment Size
Tx	transmit
UDP	User Datagram Protocol
VHD	virtual hard disk
vif	virtual interface
VLAN	virtual LAN
VPC	Amazon virtual private cloud
VPN	virtual private network
VRRP	Virtual Router Redundancy Protocol
WAN	wide area network
WAP	wireless access point
WPA	Wired Protected Access