# RIP Configuration Guide, 17.2.0

November 2017

Supporting AT&T Vyatta Network Operating System

# Contents

# Copyright Statement



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

# About This Guide

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

# RIP Configuration

## RIP overview

RIP is a dynamic routing protocol suitable for small, homogeneous networks. It is classified as an interior gateway protocol and employs the distance-vector routing algorithm. RIP determines the best path by counting the hops to the destination. The maximum hop count is 15 (16 is considered an infinite distance), making RIP less suitable for large networks. RIP is considered obsoleted by Open Shortest Path First (OSPF).

## Supported standards

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

- RFC 1058: Routing Information Protocol
- RFC 2453: RIP Version 2

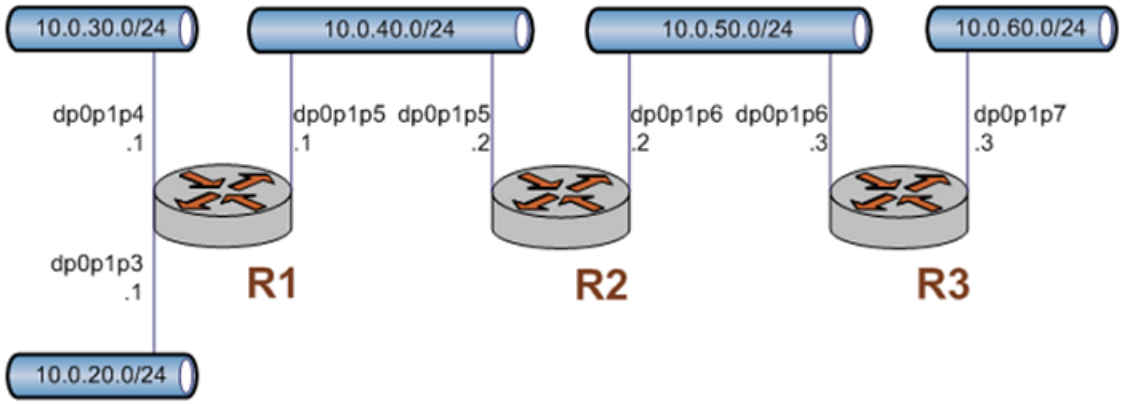
## Configuring RIP

This section presents the following topics:

- Basic RIP configuration
- Verifying the RIP configuration

This section presents a sample configuration of RIP. The RIP configuration in Basic RIP configuration *(page 7)* is based on the diagram in the following figure.

**Figure 1: Sample RIP configuration**




### Basic RIP configuration

In this section, you configure RIP on the routers that are labeled R1, R2, and R3 in the sample configuration in Configuring RIP *(page 7)*. The routers are advertising their routes on the 10.0.40.0/24 and 10.0.50.0/24 networks.

It is assumed for this example that you have already configured the router interfaces; only the steps required to implement RIP are shown.

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

**Table 1: Basic RIP configuration**

| Router | Step | Command |
|---|---|---|
| R1 | Advertise to the 10.0.40.0/24 network. | `vyatta@R1# set protocols rip network 10.0.40.0/24` |
| R1 | Redistribute connected routes to RIP. | `vyatta@R1# set protocols rip redistribute connected` |
| R1 | Commit the configuration. | `vyatta@R1# commit` |
| R1 | Display the configuration. | `vyatta@R1# show protocols rip { network 10.0.40.0/24 redistribute { connected { } } }` |
| R2 | Advertise to the 10.0.40.0/24 network. | `vyatta@R2# set protocols rip network 10.0.40.0/24` |
| R2 | Advertise to the 10.0.50.0/24 network. | `vyatta@R2# set protocols rip network 10.0.50.0/24` |
| R2 | Redistribute connected routes to RIP. | `vyatta@R2# set protocols rip redistribute connected` |
| R2 | Commit the configuration. | `vyatta@R2# commit` |
| R2 | Display the configuration. | `vyatta@R2# show protocols rip { network 10.0.40.0/24 network 10.0.50.0/24 redistribute { connected { } } }` |
| R3 | Advertise to the 10.0.50.0/24 network. | `vyatta@R3# set protocols rip network 10.0.50.0/24` |

| Router | Step | Command |
|---|---|---|
| R3 | Redistribute connected routes to RIP. | `vyatta@R3# set protocols rip redistribute connected` |
| R3 | Commit the configuration. | `vyatta@R3# commit` |
| R3 | Display the configuration. | `vyatta@R3# show protocols rip { network 10.0.50.0/24 redistribute { connected { } } }` |

## Verifying the RIP configuration

The following operational mode commands verify the RIP configuration.

### show ip route

The `show ip route` command shows how to verify RIP on the R3 router.

```
vyatta@R3:~$ show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route
R>* 10.0.20.0/24 [120/3] via 10.0.50.2, dp0p1p6, 00:20:16
R>* 10.0.30.0/24 [120/3] via 10.0.50.2, dp0p1p6, 00:34:04
R>* 10.0.40.0/24 [120/2] via 10.0.50.2, dp0p1p6, 02:15:26
C>* 10.0.50.0/24 is directly connected, dp0p1p6
C>* 10.0.60.0/24 is directly connected, dp0p1p7
C>* 127.0.0.0/8 is directly connected, lo
vyatta@R3:~$
```

The output shows that routes to the 10.0.20.0/24, 10.0.30.0/24, and 10.0.40.0/24 networks have been learned through RIP and that packets to those networks are forwarded out dp0p1p6 to 10.0.50.2. The 10.0.50.0/24 and 10.0.60.0/24 networks are directly connected.

### show ip rip

show ip rip *(page 21)* for R3 displays similar RI verification information in a different format.

```
vyatta@R3:~$ show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface
     Network            Next Hop         Metric From            Tag Time
R(n) 10.0.20.0/24       10.0.50.2             3 10.0.50.2        0 00:23
R(n) 10.0.30.0/24       10.0.50.2             3 10.0.50.2        0 00:23
R(n) 10.0.40.0/24       10.0.50.2             2 10.0.50.2        0 00:23
C(i) 10.0.50.0/24       0.0.0.0               1 self             0
C(r) 10.0.60.0/24       0.0.0.0               1 self (connected:1)  0
vyatta@R3:~$
```

Again, the output shows that routes to 10.0.20.0/24, 10.0.30.0/24, and 10.0.40.0/24 have been learned through RIP and that packets to those networks are forwarded to 10.0.50.2. The 10.0.50.0/24 and 10.0.60.0/24 networks are directly connected.

### ping 10.0.20.1

Using the `ping` command from the R3 router, you can confirm that hosts on remote networks can be reached. In this case we ping an IP address on R1. `ping 10.0.20.1` shows how to ping an IP address on the R1 router.

```
vyatta@R3:~$ ping 10.0.20.1
PING 10.0.20.1 (10.0.20.1) 56(84) bytes of data.
64 bytes from 10.0.20.1: icmp_seq=1 ttl=63 time=7.39 ms
64 bytes from 10.0.20.1: icmp_seq=2 ttl=63 time=1.56 ms
64 bytes from 10.0.20.1: icmp_seq=3 ttl=63 time=1.49 ms
^C
--- 10.0.20.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.497/3.482/7.390/2.763 ms
vyatta@R3:~$
```

This output confirms that the RIP configuration is working and that a remote network can be reached.

# Router-level Configuration Commands

## monitor protocol rip disable events

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

**Syntax:**

```
monitor protocol rip disable events
```

**Operational mode**

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

## monitor protocol rip disable packet

Disables the generation of debug messages that are related to all types of RIP packets.

**Syntax:**

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

**recv**
Optional. Disables debugging on all received packets.

**send**
Optional. Disables debugging on all sent packets.

**Operational mode**

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

## monitor protocol rip disable rib

Disables the generation of debug messages that are related to the RIP Routing Information Base (RIB).

**Syntax:**

```
monitor protocol rip disable rib
```

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

**Operational mode**

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

## monitor protocol rip enable events

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

**Syntax:**

```
monitor protocol rip enable events
```

**Operational mode**

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

## monitor protocol rip enable packet

Enables the generation of debug messages that are related to all types of RIP packets.

**Syntax:**

```
monitor protocol rip enable packet [ recv | send ]
```

**recv**
Optional. Enables debugging on all received packets.

**send**
Optional. Enables debugging on all sent packets.

**Operational mode**

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

# monitor protocol rip enable rib

Enables the generation of debug messages that are related to the RIP Routing Information Base (RIB).

**Syntax:**
```
monitor protocol rip enable rib
```

Debug messages are generated for actions related to the RIP RIB.

**Operational mode**

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

# protocols rip default-distance <distance>

Sets the default administrative distance for RIP.

**Syntax:**
```
set protocols rip default-distance distance
```

**Syntax:**
```
delete protocols rip default-distance
```

**Syntax:**
```
show protocols rip default-distance
```

The default administrative distance is 120.

***distance***
Mandatory. The default administrative distance. The distance ranges from 1 through 255. The default distance is 120.

**Configuration mode**

```
protocols {
 rip {
  default-distance distance
 }
}
```

Use the `set` form of this command to set the default administrative distance for RIP.

Use the `delete` form of this command to restore the default administrative distance for RIP, which is 120.

Use the `show` form of this command to display the default administrative distance for RIP.

# protocols rip default-information originate

Generates a default route to the RIP routing domain.

**Syntax:**

```
set protocols rip default-information originate
```

**Syntax:**

```
delete protocols rip default-information originate
```

**Syntax:**

```
show protocols rip default-information originate
```

By default, the system does not generate a default route.

**Configuration mode**

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

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

Use the `delete` form of this command to restore the default behavior for default route generation to the RIP routing domain, that is, the system does not generate a route.

Use the `show` form of this command to display the default route generation to the RIP routing domain.

# protocols rip default-metric <metric>

Changes the default metric for routes that are redistributed to RIP.

**Syntax:**

`set protocols rip default-metric` *metric*

**Syntax:**

```
delete protocols rip default-metric
```

**Syntax:**

```
show protocols rip default-metric
```

Routes that are redistributed to RIP are assigned a metric of 1.

***metric***

Mandatory. A metric that is assigned to routes. The metric ranges from 1 through 16. The default metric is 1.

**Configuration mode**

```
protocols {
 rip {
  default-metric metric
 }
}
```

Use the `set` form of this command to change the metric for routes that are redistributed to RIP.

Use the `delete` form of this command to restore the default metric to 1 for routes that are redistributed to RIP.

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

# protocols rip interface <interface>

Enables RIP on an interface.

**Syntax:**
`set protocols rip interface` *interface*

**Syntax:**
`delete protocols rip interface` *interface*

**Syntax:**
`show protocols rip interface` *interface*

***interface***
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 34)*.

You can enable RIP on more than one interface by creating multiple **`protocols rip interface`** configuration nodes.

**Configuration mode**

```
protocols {
 rip {
  interface interface
 }
}
```

Use the `set` form of this command to enable RIP on an interface. The interface must be enabled for RIP before you can use it for RIP routing.

Use the `delete` form of this command to disable RIP on an interface.

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

# protocols rip log

Enables logging for RIP.

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

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

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

None

**`all`**
Enables all RIP logs.

**`events`**
Enables only RIP events logs.

**`nsm`**
Enables only RIP NSM logs.

**`packet`**

Enables only RIP packet logs.

**rib**

Enables only RIP RIB logs.

**Configuration mode**

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

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

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

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

# protocols rip log packet

Enables logging for RIP packets.

**Syntax:**

```
set protocols rip log packet { all | detail | rcv | send }
```

**Syntax:**

```
delete protocols rip log packet { all | detail | rcv | send }
```

**Syntax:**

```
show protocols rip log packet { all | detail | rcv | send }
```

None

**all**

Enables all RIP packet logs.

**detail**

Enables only RIP packet detail logs.

**rcv**

Enables only RIP packet receive logs.

**send**

Enables only RIP packet send logs.

**Configuration mode**

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

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

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

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

# protocols rip neighbor <ipv4>

Defines a RIP neighbor router.

**Syntax:**
`set protocols rip neighbor` *ipv4*

**Syntax:**
`delete protocols rip neighbor` *ipv4*

**Syntax:**
`show protocols rip neighbor`

***ipv4***

The IP address of a neighbor router.

You can define more than one RIP neighbor router by creating multiple **`protocols rip neighbor`** configuration nodes.

**Configuration mode**

```
protocols {
 rip {
  neighbor ipv4
 }
}
```

Use the `set` form of this command to define a RIP neighbor router.

Use the `delete` form of this command to remove a RIP neighbor router.

Use the `show` form of this command to display the configuration of RIP neighbor routers.

# protocols rip network <ipv4net>

Specifies a network for RIP.

**Syntax:**
`set protocols rip network` *ipv4net*

**Syntax:**
`delete protocols rip network` *ipv4net*

**Syntax:**
`show protocols rip network`

***ipv4net***

Mandatory. Multi-node. The IP network address of a RIP network.

You can identify more than one RIP network by creating multiple **`protocols rip network`** configuration nodes.

**Configuration mode**

```
protocols {
 rip {
```

```
  network ipv4net
 }
}
```

Use the `set` form of this command to specify a RIP network.

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

Use the `show` form of this command to display RIP network configuration.

# protocols rip network-distance <ipv4net>

Establishes the administrative distance for or applies an access list to a RIP network.

**Syntax:**
`set protocols rip network-distance` *ipv4net* { **`access-list`** *list-name* | **`distance`** *distance* }

**Syntax:**
`delete protocols rip network-distance` *ipv4net* [ **`access-list`** *list-name* | **`distance`** *distance* ]

**Syntax:**
`show protocols rip network-distance` *ipv4net* [ **`access-list`** | **`distance`** ]

***ipv4net***
Mandatory. The IP address of a network.

***access-list***
A defined access list for the specified network.

***distance***
An administrative distance for the network. The distance ranges from 1 through 255. The default distance is 120.

**Configuration mode**

```
protocols {
   rip {
      network-distance ipv4net {
         access-list list-name
         distance distance
      }
   }
}
```

Use the `set` form of this command to establish the administrative distance for or apply an access list to a RIP network.

The administrative distance indicates the trustworthiness of a router or group of routers as a source of routing information. In general, the higher the value, the less trusted the entity. An administrative distance of 1 usually represents a directly connected network, and an administrative distance of 255 means the routing source is unreliable or unknown. The administrative distance conventionally applied to RIP is 120.

Use the `delete` form of this command to restore the default administrative distance, which is 120, to a RIP network or remove an access list.

Use the `show` form of this command to display the administrative distance of a RIP network or the application of an access list.

# protocols rip passive-interface <interface>

Suppresses RIP routing updates on an interface.

**Syntax:**
`set protocols rip passive-interface` *interface*

**Syntax:**
`delete protocols rip passive-interface` *interface*

**Syntax:**
`show protocols rip passive-interface`

RIP routing updates are not suppressed.

***interface***
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 34)*.

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

**Configuration mode**

```
protocols {
 rip {
  passive-interface interface
 }
}
```

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

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

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

# protocols rip route <ipv4net>

Defines a RIP static route.

**Syntax:**
`set protocols rip route` *ipv4net*

**Syntax:**
`delete protocols rip route` *ipv4net*

**Syntax:**
`show protocols rip route`

***ipv4net***
Mandatory. The network address of a RIP static route.

**Configuration mode**

```
protocols {
 rip {
  route ipv4net
 }
}
```

Use the `set` form of this command to define a RIP static route.

Use the `delete` form of this command to remove a RIP static route.

Use the `show` form of this command to display the configuration of RIP static routes.

# protocols rip timers garbage-collection <seconds>

Sets a timer for RIP garbage collection.

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

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

**Syntax:**
```
show protocols rip timers garbage-collection
```

RIP garbage collection occurs at 120 seconds.

***seconds***
Mandatory. An interval in seconds. The number of seconds ranges from 5 through 2147483647.

**Configuration mode**

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

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

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

Use the `show` form of this command to display the RIP garbage collection timer.

# protocols rip timers timeout <seconds>

Sets an interval for RIP time-outs.

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

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

**Syntax:**
```
show protocols rip timers timeout
```

RIP time-outs occur at 180 seconds.

***seconds***
Mandatory. An interval in seconds. The number of seconds ranges from 5 through 2147483647. The default number of seconds is 180.

**Configuration mode**

```
protocols {
 rip {
  timers {
   timeout seconds
```

```
  }
 }
}
```

Use the `set` form of this command to set an interval for RIP time-outs.

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

Use the `show` form of this command to display the RIP time-out interval.

# protocols rip timers update <seconds>

Sets a timer for updates to the RIP routing table.

**Syntax:**

```
set protocols rip timers update seconds
```

**Syntax:**

```
delete protocols rip timers update [ seconds ]
```

**Syntax:**

```
show protocols rip timers update
```

The RIP routing table is updated every 30 seconds.

***seconds***

Mandatory. An interval in seconds. The number of seconds ranges from 5 through 2147483647. The default number of seconds is 30.

**Configuration mode**

```
protocols {
 rip {
  timers {
   update seconds
  }
 }
}
```

Use the `set` form of this command to set a timer for updates to the RIP routing table. A shorter interval means more accurate routing information in the table; however, more protocol network traffic occurs.

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

Use the `show` form of this command to display the interval for updates to the RIP routing table.

# reset ip rip route

Resets data in the RIP routing table.

**Syntax:**

```
reset ip rip [ statistics | route [ all | bgp | connected | kernel | ospf | rip | static | ip-address ] ]
```

**Syntax:**

**`all`**

Removes all entries from the RIP routing table.

**`bgp`**

Removes only BGP routes from the RIP routing table.

**`connected`**

Removes entries for connected routes from the RIP routing table.

**`kernel`**

Removes kernel entries from the RIP routing table.

**ospf**

Removes only OSPF routes from the RIP routing table.

**rip**

Removes only RIP routes from the RIP routing table.

**static**

Removes static entries from the RIP routing table.

***ip-address***

Removes entries that match *ip-address (x.x.x.x/x)*, a destination IP address, from the RIP routing table.

**statistics**

Resets the RIP statistics.

**Operational mode**

Use the `reset ip rip route all` command to clear the RIP routing table.

# show ip rip

Displays information for the Routing Information Protocol (RIP).

**Syntax:**

```
show ip rip [ status ]
```

Displays all RIP protocol information.

**status**

Optional. Displays only protocol status.

**Operational mode**

Use this command to display protocol information for RIP.

The following example shows how to display protocol information for RIP.

```
vyatta@vyatta:~$ show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
      (n) - normal, (s) - static, (d) - default, (r) - redistribute,
      (i) - interface
     Network            Next Hop         Metric From            Tag Time
C(i) 192.168.1.0/24     0.0.0.0               1 self              0
vyatta@vyatta:~$
```

# show ip route rip

Displays all IP RIP routes that are contained in the Routing Information Base (RIB).

**Syntax:**

```
show ip route rip
```

**Operational mode**

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

The following example shows how to display all RIP routes that are contained in the RIB.

```
vyatta@vyatta:~$ show ip route rip
```

```
R       19.1.1.0/24 [120/1] is directly connected, dp0p192p1, 00:01:04
vyatta@vyatta:~$
```

# show monitoring protocols rip

Displays RIP protocol debugging flags.

**Syntax:**

```
show monitoring protocols rip
```

**Operational mode**

Use this command to see how debugging is set for RIP.

# Route Redistribution Commands

## protocols rip redistribute bgp

Redistributes Border Gateway Protocol (BGP) routes into RIP routing tables.

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

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

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

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

***metric***
A routing metric. The metric ranges from 1 through 16. The default metric is 1.

***map-name***
Optional. A route map.

**Configuration mode**

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

Use the `set` form of this command to set the routing metric for BGP routes being redistributed into RIP, or to specify a route map to be applied to redistributed BGP routes.

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

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

## protocols rip redistribute connected

Redistributes directly connected routes into RIP routing tables.

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

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

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

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

***metric***
Optional. A routing metric. The metric ranges from 1 through 16. The default metric is 1.

***map-name***
Optional. A route map.

**Configuration mode**

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

Use the `set` form of this command to set the routing metric for connected routes being redistributed into RIP, or to specify a route map to be applied to redistributed connected routes.

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

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

# protocols rip redistribute kernel

Redistributes kernel routes into RIP routing tables.

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

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

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

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

***metric***
Optional. A routing metric. The metric ranges from 1 through 16. The default metric is 1.

***map-name***
Optional. A route map.

**Configuration mode**

```
protocols {
 rip {
  redistribute {
   kernel {
    metric metric
    route-map map-name
   }
  }
 }
}
```

Use the `set` form of this command to set the routing metric for kernel routes being redistributed into RIP, or to specify a route map to be applied to redistributed kernel routes.

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

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

# protocols rip redistribute ospf

Redistributes (OSPF) routes into RIP routing tables.

**Syntax:**

```
set protocols rip redistribute ospf [ metric metric | route-map map-name ]
```

**Syntax:**

```
delete protocols rip redistribute ospf [ metric | route-map ]
```

**Syntax:**

```
show protocols rip redistribute ospf [ metric | route-map ]
```

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

***metric***

Optional. A routing metric. The metric ranges from 1 through 16. The default metric is 1.

***map-name***

Optional. A route map.

**Configuration mode**

```
protocols {
 rip {
  redistribute {
   ospf {
    metric metric
    route-map map-name
   }
  }
 }
}
```

Use the `set` form of this command to set the routing metric for OSPF routes being redistributed into RIP, or to specify a route map to be applied to redistributed OSPF routes.

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

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

# protocols rip redistribute static

Redistributes static routes into RIP routing tables.

**Syntax:**

```
set protocols rip redistribute static [ metric metric | route-map map-name ]
```

**Syntax:**

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

**Syntax:**

```
show protocols rip redistribute static [ metric | route-map ]
```

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

***metric***

Optional. A routing metric. The metric ranges from 1 through 16. The default metric is 1.

***map-name***

Optional. A route map.

**Configuration mode**

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

Use the `set` form of this command to set the routing metric for static routes being redistributed into RIP, or to specify a route map to be applied to redistributed static routes.

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

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

# Route Filtering Commands

## protocols rip distribute-list access-list

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

**Syntax:**

```
set protocols rip distribute-list access-list { in in-list | out out-list }
```

**Syntax:**

```
delete protocols rip distribute-list access-list { in | out }
```

**Syntax:**

```
show protocols rip distribute-list access-list { in | out }
```

**`in` *in-list* | `out` *out-list***

*in-list* : The identifier of a defined access list. The access list is applied to filter inbound RIP packets.

*out-list* :The identifier of a defined access list. The access list is applied to filter outbound RIP packets.

The number of the access list that is used to filter networks in routing updates. The number ranges are as follows:

*1-99*: IP standard access list.

*100-199*: IP extended access list.

*1300-1999*: IP standard access list (expanded range).

*2000-2699*: IP extended access list (expanded range).

**Configuration mode**

protocols { rip { distribute list { access list { in inlist out out#list } } } }

```
protocols
 rip {
  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 RIP packets.

Use the `delete` form of this command to remove filtering of RIP packets by access list.

Use the `show` form of this command to display the configuration for filtering of RIP packets by access list.

## protocols rip distribute-list interface <interface> access-list

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

**Syntax:**

```
set protocols rip distribute-list interface interface access-list { in in-list | out out-list ]
```

**Syntax:**

```
delete protocols rip distribute-list interface interface access-list { in | out }
```

**Syntax:**

```
show protocols rip distribute-list interface interface access-list { in | out }
```

***interface***

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 34)*.

***in-list***

The identifier of a defined access list. The access list is applied to the interface to filter inbound RIP packets.

***out-list***

The identifier of a defined access list. The access list is applied to the interface to filter outbound RIP packets.

**Configuration mode**

```
protocols {
 rip {
  distribute-list {
   interface interface {
    access-list {
     in in-list
     out out-list
    }
   }
  }
 }
}
```

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

Use the `delete` form of this command to remove filtering of RIP packets by access list from an interface.

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

# protocols rip distribute-list interface <interface> prefix-list

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

**Syntax:**

```
set protocols rip distribute-list interface interface prefix-list { in in-list | out out-list }
```

**Syntax:**

```
delete protocols rip distribute-list interface interface prefix-list { in | out }
```

**Syntax:**

```
show protocols rip distribute-list interface interface prefix-list { in | out }
```

***interface***

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 34)*.

***in-list***

The identifier of a defined prefix list. The prefix list is applied to the interface to filter inbound RIP packets.

***out-list***

The identifier of a defined prefix list. The prefix list is applied to the interface to filter outbound RIP packets.

**Configuration mode**

```
protocols {
 rip {
  distribute-list {
   interface interface {
    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 RIP packets.

Use the `delete` form of this command to remove filtering of RIP packets by prefix list from an interface.

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

# protocols rip distribute-list prefix-list

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

**Syntax:**

```
set protocols rip distribute-list prefix-list { in in-list | out out-list }
```

**Syntax:**

```
delete protocols rip distribute-list prefix-list { in | out }
```

**Syntax:**

```
show protocols rip distribute-list prefix-list { in | out }
```

***in-list***

The identifier of a defined prefix list. The prefix list is applied to filter inbound RIP packets.

***out-list***

The identifier of a defined prefix list. The prefix list is applied to filter outbound RIP packets.

**Configuration mode**

```
protocols {
 rip {
  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 RIP packets.

Use the `delete` form of this command to remove filtering of RIP packets by prefix list.

Use the `show` form of this command to display the configuration for filtering of RIP packets by prefix list.

# RIP Interface Commands

## interfaces <interface> ip rip

Enables RIP on an interface.

**Syntax:**
`set interfaces` *interface* **`ip rip`**

**Syntax:**
`delete interfaces` *interface* **`ip rip`**

**Syntax:**
`show interfaces` *interface* **`ip rip`**

***interface***
Mandatory. A type of interface. For detailed keywords and arguments that can be specified as an interface, refer to Supported Interface Types *(page 34)*.

**Configuration mode**

```
interfaces interface {
 ip {
  rip
 }
}
```

Use this command to enable RIP on an interface.

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

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

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

## interfaces <interface> ip rip authentication

Establishes an authentication method to be used for RIP on an interface.

**Syntax:**
`set interfaces` *interface* **`ip rip authentication`** [ **`md5`** *md5-key* **`password`** *md5-password* | **`plaintext-password`** *password* ]

**Syntax:**
`delete interfaces` *interface* **`ip rip authentication`** [ **`md5`** *md5-key* **`password`** | **`plaintext-password`** ]

**Syntax:**
`show interfaces` *interface* **`ip rip authentication`** [ **`md5`** *md5-key* **`password`** | **`plaintext-password`** ]

***interface***
Mandatory. A type of interface. For detailed keywords and arguments that can be specified as an interface, refer to Supported Interface Types *(page 34)*.

***md5-key***
Optional. An authentication key. This key must be the same on both the sending and receiving systems. The key ranges from 1 through 255.

***md5-password***

Optional. A password to use in MD5 authentication. This password must be the same on both the sending and receiving systems.

***password***

Optional. A password to use in simple (plain text) authentication. This password must be the same on both the sending and receiving systems.

**Configuration mode**

```
interfaces interface {
 ip {
  rip {
   authentication {
    md5 md5-key {
     password md5-password
    }
    plaintext-password password
   }
  }
 }
}
```

Use this command to establish an authentication method to be used for RIP on an interface. This authentication is independent of the authentication configured for the RIP area.

In plain text authentication, passwords are sent through the network in plain text. In MD5 authentication, the system uses the Message Digest 5 (MD5) algorithm to compute a hash value from the contents of the RIP packet and the password. The hash value and the MD5 key are included in the transmitted packet, and the receiving system (configured with the same password) calculates its own hash function, which must match.

The authentication parameters 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 do not consider adjacencies, and disregard communication from each other.

Use the `set` form of this command to specify an authentication method to be used for RIP on an interface.

Use the `delete` form of this command to remove an authentication method to be used for RIP from an interface.

Use the `show` form of this command to display an authentication method to be used for RIP on an interface.

# interfaces <interface> ip rip split-horizon

Enables split-horizon or split-horizon poison-reverse on an interface that is running RIP.

**Syntax:**

`set interfaces` *interface* **`ip rip split-horizon`** [ **`disable`** | **`poison-reverse`** ]

**Syntax:**

`delete interfaces` *interface* **`ip rip split-horizon`** [ **`disable`** | **`poison-reverse`** ]

**Syntax:**

`show interfaces` *interface* **`ip rip split-horizon`**

Split-horizon and split-horizon poison-reverse are disabled.

***interface***

Mandatory. A type of interface. For detailed keywords and arguments that can be specified as an interface, refer to Supported Interface Types *(page 34)*.

**`disable`**

Disables split-horizon on the interface.

**`poison-reverse`**

Enables split-horizon poison-reverse on the interface.

**Configuration mode**

```
interfaces interface {
 ip {
  rip {
   split-horizon {
    disable
    poison-reverse
   }
  }
 }
}
```

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

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 of any routes that it learned from that interface. Split-horizon is effective at preventing loops between routers that are directly connected to each another and speeds convergence when network conditions change; it is the default setting in RIP.

Poison-reverse is a variation of split-horizon. When an interface that has poison-reverse enabled detects that a link 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 are considered reachable on a RIP network, increasing the metric to 16 renders the route unreachable as far as downstream RIP routers are concerned. This is called “poisoning” the route. Poison-reverse can be used to propagate 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 in 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 RIP.

Use the `delete` form of this command to restore the default configuration, that is, split-horizon and split-horizon poison-reverse are disabled.

Use the `show` form of this command to display whether split-horizon and split-horizon poison-reverse are enabled or disabled.

# 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:<br>• `dp`*x*`p`*y*`p`*z*—The name of a data plane interface, where<br>— `dp`*x* specifies the data plane identifier (ID). Currently, only dp0 is supported.<br>— `p`*y* specifies a physical or virtual PCI slot index (for example, p129).<br>— `p`*z* specifies a port index (for example, p1). For example, dp0p1p2, dp0p160p1, and dp0p192p1.<br>• `dp`*x*`em`*y* —The name of a data plane interface on a LAN-on-motherboard (LOM) device that does not have a PCI slot, where `em`*y* specifies an embedded network interface number (typically, a small number). For example, dp0em3.<br>• `dp`*x*`s`*y*—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.<br>• `dp`*x*`P`*n*`p`*y*`p`*z* —The name of a data plane interface on a device that is installed on a secondary PCI bus, where `P`*n* specifies the bus number. You can use this format to name data plane interfaces on large physical devices with multiple PCI buses. For these devices, it is possible to have network interface cards installed on different buses with these cards having the same slot ID. The value of *n* must be an integer greater than 0. For example, dp0P1p162p1 and dp0P2p162p1. |

| Interface Type | Syntax | Parameters |
|---|---|---|
| Data plane vif | `dataplane` *interface-name* `vif` *vif-id* [`vlan` *vlan-id* ] | *interface-name*: Refer to the preceding description.<br>*vif-id*: A virtual interface ID. The ID ranges from 1 through 4094.<br>*vlan-id*: The VLAN ID of a virtual interface. The ID ranges from 1 through 4094. |
| Loopback | `loopback lo`<br>or<br>`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*<br>or<br>`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.<br>**Note:** Before you can configure a vti interface, you must configure a corresponding vpn.<br>**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.<br>*group*: A VRRP group identifier.<br>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 |