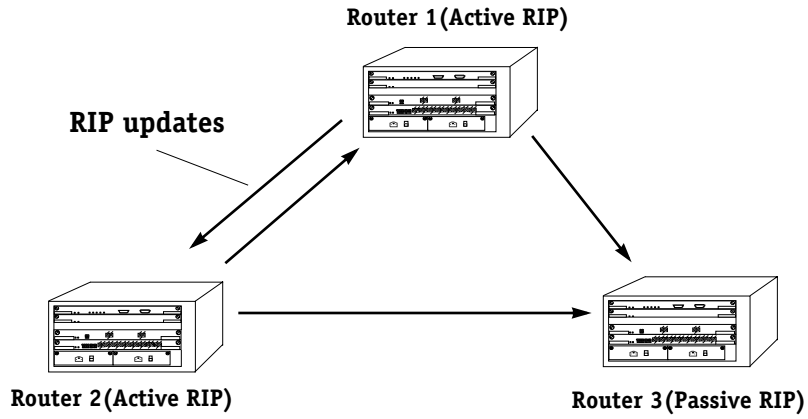
# 6 The Routing Information Protocol (RIP)

One of the most widely used interior gateway protocols is the Routing Information Protocol (RIP). RIP is an implementation of a distance-vector protocol for local networks. It classifies routers as active and passive (silent). Active routers advertise their routes, or reachability information, to other routers. Passive routers listen and update their routes based on advertisements, but do not advertise. Typically, routers run RIP in active mode, while hosts use passive mode.

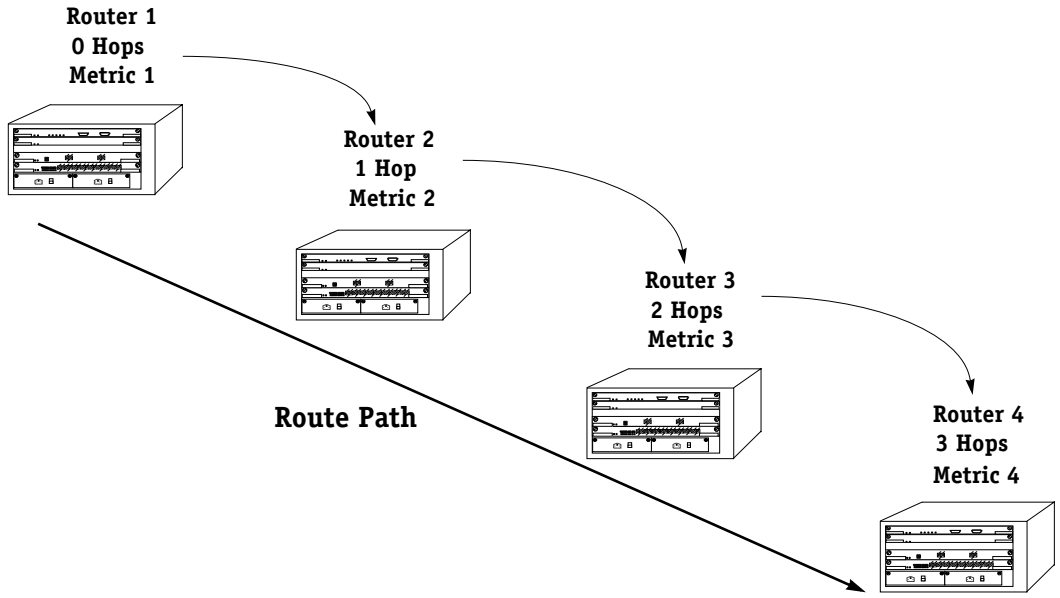
A router running RIP in active mode broadcasts updates at set intervals. Each update contains a pair of values consisting of an IP network address and an integer distance to that network. The diagram below shows the behavior of active and passive modes:




**RIP Network Updates**

In this example, Router 1 and 2 are actively advertising their RIP status, while Router 3 is passive and only receives updates.

RIP uses a hop count metric to measure the distance to a destination. In the RIP metric, a router advertises directly connected networks at a metric of **1**. Networks that are reachable through one other gateway are two hops, networks that are reachable through 2 gateways are three hops, etc. Thus, the number of hops (or hop count) along a path from a given source to a given destination refers to the number of gateways that a datagram would encounter along that path, as shown below:




Using hop counts to calculate shortest paths does not always produce optimal results. For example, a path with a hop count 3 that crosses three Ethernets may be substantially faster that a path with a hop count 2 that crosses two slow-speed serial lines. To compensate for differences in technology, many routers advertise artificially high hop counts for slow links.

## The RIP Routing Table

RIP dynamically builds on information received through RIP updates. When started up, it issues a REQUEST for routing information, and then listens for responses to the request. If a system configured to supply RIP hears the request, it responds with a RESPONSE packet based on information in its routing database. The RESPONSE packet contains destination network addresses and the routing metric for each destination.

When a RIP RESPONSE packet is received, RIP takes the information and rebuilds the routing database, adding new routes and "better" (lower metric) routes to destinations already listed in the database. RIP also deletes routes from the database if the next router to that destination says the route contains more than 15 hops, or if the route is deleted. All routes through a gateway are deleted if no updates are received from that gateway for a specified time period. In general, routing updates are issued every 30 seconds. In many implementations, if a gateway is not heard from for 180 seconds, all routes from that gateway are deleted from the routing database. This 180-second interval also applies to deletion of specific routes.

## RIP Version 2

RIP version 2 (more commonly known as RIP II) adds additional capabilities to RIP. Not all RIP II enhancements are compatible with RIP I. To avoid supplying information to RIP I routes that could be misinterpreted, RIP II can only use non-compatible features when its packets are multicast (as RIP I doesn't support multicast). On interfaces that are not compatible with IP multicast, the RIP I-compatible packets used do not contain potentially confusing information.

RIP II enhancements are listed below.

- **Next hop**. RIP II can advertise a next hop other than the router supplying the routing update. This capability is useful when advertising a static route to a silent router not using RIP, since packets passing through the silent router do not have to cross the network twice.

- **Network Mask**. RIP I assumes that all subnetworks of a given network have the same network mask. It uses this assumption to calculate the network masks for all routes received. This assumption prevents subnets with different netmasks from being included in RIP packets. RIP II adds the ability to specify the network mask with each network in a packet.

  Due to the fact that RIP I routers ignore the network mask in RIP II packets, their calculation of the network mask could possibly be wrong. For this reason, RIP I-compatible RIP II packets cannot contain networks that would be misinterpreted by RIP I. These networks must only be provided in native RIP II packets that are multicast.

- **Authentication**. RIP II packets can contain one of two types of authentication that may be used to verify the validity of the supplied routing data. Authentication may be used in RIP I-compatible RIP II packets, but RIP I routers will ignore authentication information.

  The first method is a simple password in which an authentication key of up to 16 characters is included in the packet. If this key does not match what is expected, the packet will be discarded. This method provides very little security as it is possible to learn the authentication key by watching RIP packets.

  The second method uses the MD5 algorithm to create a crypto-checksum of a RIP packet and an authentication key of up to 16 characters. The transmitted packet does not contain the authentication key itself, instead it contains a crypto-checksum, called the *digest*. The receiving router will perform a calculation using the correct authentication key and discard the packet if the digest does not match. In addition, a sequence number is maintained to prevent the replay of older packets. This method provides a much stronger assurance that routing data originated from a router with a valid authentication key.

### RIP I and Network Masks

RIP I derives the network mask of received networks and hosts from the network mask of the interface though which the packet was received. If a received network or host is on the same network as the interface over which it was received, and that network is subnetted (i.e., the specified mask is more specific than the natural netmask), the subnet mask is applied to the destination. If bits outside the mask are set, it is assumed to be a host; otherwise it is assumed to be a subnet.

On point-to-point interfaces, the netmask is applied to the remote address. The netmask on these interfaces is ignored if it matches the natural network of the remote address, or is all ones.

Unlike previous releases, the zero subnet mask (a network that matches the natural network of the interface, but has a more specific, or longer, network mask) is ignored. If this is not desirable, a route filter may be used to reject it.

## Specifying Gateways When Exporting RIP Routes

When exporting RIP routes you can specify either a list of interfaces or a list of gateways. If you specify export routes with a gateway list, then you must also include the **sourcegateways** statement in your main rip statement.

For example, the following **gated.conf** file shows RIP routes being exported to OSPF:

```
routerid 202.206.184.108;
rip yes {
        interface 202.206.184.108 version 2;
};
ospf yes {
        area 1 {
                interface 80.0.0.1 {priority 1; };
        };
};
export proto rip gateway 202.206.184.106 metric 3 {
        proto ospfase { all;};
};
```

RIP and OSPF are enabled by these file statements and the statements are syntactically correct. The line near the end of the file

```
export proto rip gateway 202.206.184.106 metric 3 {
```

indicates that OSPFASE routes should be exported via RIP to the gateway 202.206.184.106. However, there is an error in the file due to the omission of a **sourcegateways** statement in the **rip yes** clause. Because the 202.208.184.106 gateway is not mentioned in the RIP clause routes will not be exported to this gateway. The **rip yes** clause needs to be modified to the following:

```
rip yes {
        interface 202.206.184.108 version 2;
        sourcegateways 202.206.184.106;
};
```

before the routes will be exported to 202.208.184.106 by GateD. The full **gated.conf** file should read as follows:

```
router-id 202.206.184.108;
rip yes {
        interface 202.206.184.108 version 2;
        sourcegateways 202.206.184.106;
};
ospf yes {
        area 1 {
                interface 80.0.0.1 {priority 1; };
        };
};
export proto rip gateway 202.206.184.106 metric 3 {
        proto ospfase { all;};
};
```

## Restricting RIP Traffic to Gateways

It is not possible to restrict RIP traffic to a specific gateway by simply adding a **restrict** statement to an **export rip** clause. For example, the following statement

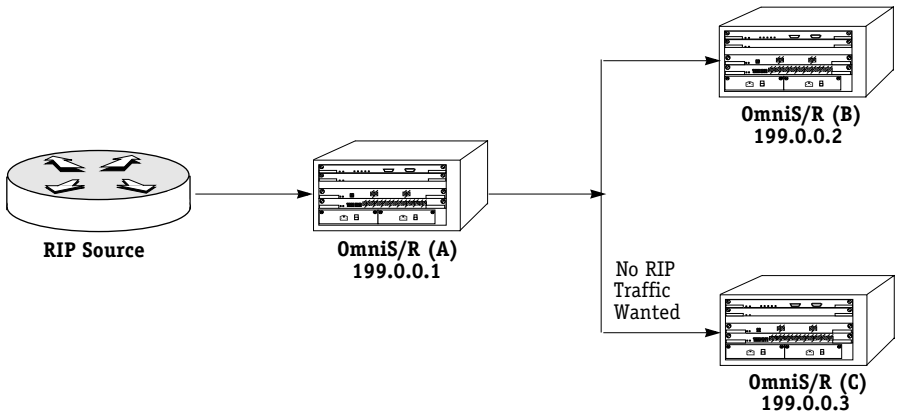
```
export proto rip gateway 199.0.0.100 restrict;
```

may not keep RIP traffic from the gateway at address 199.0.0.100. Because most RIP packets are either broadcast or multicast, you cannot prevent a connected gateway from receiving such traffic simply by issuing this statement. Instead, you must consider an export strategy for each individual gateway on the connected network.

The general steps for limiting RIP traffic to a gateway are as follows:

1. Prevent all broadcast/multicast RIP traffic on the entire connected network.
2. Selectively enable unicast RIP traffic to each gateway that *should* receive this traffic.

As an example, consider a network (199.0.0.0 and mask of 255.255.255.0) with three switch/routers as follows:




**Restricting RIP Traffic from a Selected Gateway**

OmniS/R A receives RIP traffic from a router outside the 199.0.0.0 network. You want to distribute this routing information to OmniS/R B, but *not* to OmniS/R C. Since OmniS/R A is already sending RIP broadcasts and multicasts on the 199.0.0.0 network, there is no way to prevent a single source (i.e., OmniS/R C) from seeing this information without careful configuration changes.

First you must turn off broadcast and multicast traffic on the entire 199.0.0.0 network, then enable the traffic to OmniS/R B. The following sections describe how to make these changes through GateD configuration file commands.

### 1. Restricting RIP Broadcast/Multicast Traffic on 199.0.0.0

There are several ways to restrict RIP traffic on the 199.0.0.0 network. The first is to use a **nobroadcast** statement in the **rip** clause as follows:

```
rip yes {
        nobroadcast;
        interface 199.0.0.1 version 2;
};
```

This specification actually turns off RIP broadcast traffic on all interfaces, not just the 199.0.0.0 network. A less restrictive option would be to use the **noripout** statement for the specific router interface you want to restrict, as follows:

```
rip yes {
        interface 199.0.0.1 version 2 noripout;
};
```

This clause would only disable RIP traffic to the 199.0.0.1 switch. It is not as restrictive as the **nobroadcast** statement, but still halts RIP traffic on the 199.0.0.0 network. You could also restrict RIP traffic to the 199.0.0.1 switch by using an **export** filter statement as follows:

```
rip yes {
        interface 199.0.0.1 version 2;
};
export proto rip interface 199.0.0.1 restrict;
```

### 2. Enabling RIP Unicast Traffic to 199.0.0.2

In this example we will use the **export** statement option to restrict traffic on the 199.0.0.0 network (the third option described earlier). The goal of this configuration is to stop RIP traffic to the 199.0.0.3 switch. However, RIP traffic is now not reaching the 199.0.0.3 switch *and* the 199.0.0.2 switch. RIP traffic still needs to be forwarded to the 199.0.0.2 switch.

Using another **export** statement, you can allow traffic to the 199.0.0.2 switch as follows:

```
rip yes {
        interface 199.0.0.1 version 2;
        sourcegateways 199.0.0.2
};
export proto rip interface 199.0.0.1 restrict;
export proto rip gateway 199.0.0.2 { proto rip { all; }; };
```

Two statements have been added. The **sourcegateways** statement in the main rip clause identifies the 199.0.0.2 switch to GateD so that GateD will recognize it when it is specified in the export statement. (See *Specifying Gateways When Exporting RIP Routes* on page 6-4 for more information on use of the sourcegateways statement.)

The **export proto rip gateway** statement tells GateD which routes to send to 199.0.0.2 (i.e., **all** routes). This statement is necessary since we had already restricted RIP traffic on the 199.0.0.1 switch with the first export statement.

This final file will achieve the purpose of the configuration—to restrict RIP traffic to the 199.0.0.3 switch. OmniS/R A (199.0.0.1) will still send RIP traffic to OmniS/R B (199.0.0.2) via unicast packets. However, it will not forward any RIP traffic to OmniS/R C (199.0.0.3).

# RIP UI Commands

The switch's User Interface (UI) contains several commands for viewing RIP configuration parameters. The GateD UI commands operate slightly differently than normal switch commands, as you must modify the root command (i.e., **rip**) with a parameter. For example, to see statistics for RIP interfaces, you would enter the following:

```
rip stat
```

To view the possible RIP command parameters, enter **rip** at the system prompt. The following menu displays:

```
parameters for the rip root command are:
        stat            RIP interface statistics
        conf            RIP interface configuration
        peers           List of RIP peers
```

The parameters for the RIP root command are described in the following sections. RIP is configured with the **gated.conf** file command statements. These command statements are described beginning in *The RIP Statement* on page 6-11.

## Viewing RIP Interface Statistics

You can obtain general RIP interface statistics by entering the following command:

```
rip stat
```

A screen similar to the following displays:

```
RIP is running on 2 interface(s)
IP Address              Bad Packets     Bad Routes      Updates Sent
================================================================
172.23.9.106                  0               0               0
172.23.9.107                  0               0               0

Total Route Changes     :       10
Total Queries           :       0
```

The first line indicates how many interfaces on the switch are set up to run RIP. In the sample screen above, RIP is active on 2 interfaces shown in the table (172.23.9.106 and 172.23.9.107).

**IP Address**. The IP address of the RIP interface. Interfaces for RIP are specified in the configuration file. For specifics on setting up an interface to use RIP, see *The RIP Statement* on page 6-11.

**Bad Packets**. A counter showing how many packets were rejected by the RIP protocol on the indicated interface. Packets are rejected if they are not recognized due to errors such as bad routing information or packet corruption.

**Bad Routes**. A counter showing the number of routes that are known to the switch but are not responding correctly to RIP requests. Bad routes can be caused by broken links or other network errors.

**Updates Sent**. A counter showing the number of times this switch has sent routing information updates. Route updates are learned from RIP queries and responses sent between routers in a network.

**Total Route Changes**. A counter showing the number of times this switch has changed information in its routing table due to updates sent from other routers in the network.

**Total Queries**. A counter showing the total number of queries from other routers received by this switch.

## Viewing RIP Interface Configuration

You can view general RIP interface configuration information by entering the following command:

```
rip conf
```

A screen similar to the following displays:

```
2 RIP interface(s) currently active

IP Address      Authentication Type   Sending Version   Receiving Version   Default Metric
=============================================================================================
172.23.9.106    None                  Version 1         Version 1 and 2     0
172.23.9.107    None                  Version 1         Version 1 and 2     0
```

The first line indicates how many interfaces on the switch are currently running RIP. In the sample screen above, RIP is active on the interfaces shown in the table (172.23.9.106 and 172.23.9.107).

**IP Address**. The IP address of the RIP interface. Interfaces for RIP are specified in the configuration file. For specifics on setting up an interface to use RIP, see *The RIP Statement* on page 6-11.

**Authentication Type**. The type of authentication used when RIP packets are sent and received by this interface. The options for this field are **none**, **simple**, and **MD5**. For information on configuring the type of authentication, see page 6-12. Authentication is only available for RIP version 2. For information on RIP II and authentication, see *RIP Version 2* on page 6-3.

**Sending Version**. The version of RIP used by this interface when sending RIP requests. The options are **1**, **2,** or **both**. For information on specifying the RIP version number, see page 6-12. RIP version 2 can communicate with version 1, but not the other way around.

**Receiving Version**. The version of RIP used by this interface when receiving RIP requests. The options are **1**, **2,** or **both**. For information on specifying the RIP version number, see page 6-12. RIP version 2 can communicate with version 1, but not the other way around.

**Default Metric**. This is a metric number assigned to routes exported from this interface if the route was originally learned through another protocol. If no default metric is assigned, the metric is automatically set to 16 (unreachable). For information on metrics, see Chapter 3 of this manual. For information on setting the default metric, see page 6-12.

## Listing RIP Peers

A *peer* is any router in the network using RIP that communicates with the local switch. You can view a list of RIP peers by entering the following command:

```
rip peers
```

A screen similar to the following appears:

```
4 RIP peer(s)

                  Seconds since                  Bad Packets   Bad Routes
IP Address        Last Update      Version       Received      Received
================================================================================
172.23.0.250      7                1             0             0
172.23.0.253      12               1             0             0
172.23.0.254      25               1             0             0
172.23.1.1        19               1             0             0
```

The first line indicates how many peers the switch has discovered in the network via RIP queries and responses. In the sample screen above, four (4) RIP peers are shown in the table.

**IP Address**. The IP address of the peer's interface. Interfaces for RIP are specified in the configuration file. For specifics on setting an interface to use RIP, see *The RIP Statement* on page 6-11.

**Seconds since Last Update**. Routers using RIP periodically send out updates to all of their known peers. This column shows the last time in seconds a RIP update was received by the switch from the indicated peer.

**Version**. The version of RIP used by the indicated peer. The options are **1**, **2,** or **both**. For information on specifying the RIP version number, see page 6-12. RIP version 2 can communicate with version 1, but RIP version 1 cannot communicate with RIP version 2.

**Bad Packets Received**. A counter showing how many packets received from the indicated peer were rejected by the switch's RIP protocol. Packets are rejected if they are not recognized due to errors such as bad routing information or packet corruption.

**Bad Routes Received**. A counter showing the number of routes received from the indicated peer that are known to the switch but are not responding correctly to RIP requests. Bad routes can be caused by broken links and other network errors.

# The RIP Statement

The following is the complete RIP statement, which is used in the **gated.conf** file to manage RIP and RIP version 2. Each of the commands is described below.

```
rip yes | no | on | off [ {
    broadcast ;
    nobroadcast ;
    nocheckzero ;
    preference preference ;
    defaultmetric metric ;
    query authentication [none | ([simple|md5] password)] ;
    interface interface_list
            [noripin] | [ripin]
            [noripout] | [ripout]
            [metricin metric]
            [metricout metric]
            [version 1]|[version 2 [multicast|broadcast]]
            [[secondary] authentication [none| ([simple|md5] password)]]
    ;
    trustedgateways gateway_list ;
    sourcegateways gateway_list ;
    traceoptions trace_file_option trace_options ;
} ] ;
```

The syntax conventions used in the above statement are fully described in Chapter 3 of this manual.

**rip yes | no | on | off [ {**

The **rip** statement enables or disables RIP. If enabled, RIP will assume **nobroadcast** when there is only one interface, and **broadcast** when there is more than one. The details of these options are given below.

**broadcast**

This command specifies that RIP packets are broadcast regardless of the number of interfaces present. This is useful when propagating static routes or routes learned from another protocol into RIP. In some cases, the use of broadcast when only one network interface is present can cause data packets to traverse a single network twice.

**nobroadcast**

This command specifies that RIP packets will not be broadcast on attached interfaces, even if there are multiple interfaces. If the **sourcegateways** command is present (as described on 6-14), and **nobroadcast** is specified, routes will still be unicast directly to the specified gateway.

**nocheckzero**

This command specifies that RIP will not check that reserved fields in incoming version 1 RIP packets are zero. Normally RIP will reject packets with reserved fields that are non-zero.

**preference** *preference*

Sets the preference for routes learned from RIP. The default preference is 100. This preference may be overridden by a preference specified in an import policy. For general information on preference, see Chapter 4 of this manual. For information on setting the import preference, see Chapter 10.

**defaultmetric** *metric*

This command defines the metric used when advertising routes learned from other protocols . This value is a number between 1 and 16, with a lower number marking a faster route than a higher number. If not specified, the default value is 16 (unreachable).

This metric may be overridden by a metric specified in export policy. For information on setting the export preference, see Chapter 10.

**query authentication [none| ([simple|md5]** *password***)]**

This command specifies the authentication required of query packets that do not originate from routers. After the **query authentication** token, enter **none**, **simple**, or **md5**. For **simple** and **md5** authentication, a password of up to 16 characters must also be specified. For example, to use simple authentication with a password of **test**, enter the following:

```
query authentication simple test
```

The default is no authentication. This command is only valid for RIP II. For more information on authentication, see *RIP Version 2* on page 6-3.

**interface** *interface_list*

This command specifies the interface or interfaces that will pass RIP information. It also allows the configuration of various attributes for sending RIP on specific interfaces.

**♦ Note ♦**

If there are multiple interfaces configured on the same subnet, RIP updates will only be sent from the first one for which RIP output is configured.

To specify an interface, use the interface token followed by an interface address, as shown:

```
interface 1.1.1.1;
```

RIP would now operate on interface 1.1.1.1. It is possible to specify more than one interface at a time by adding more interface statements, as shown:

```
interface 1.1.1.1;
interface 1.1.1.2;
interface 1.1.1.3;
```

RIP would now operate on all interfaces specified in the statement. To enable RIP on all interfaces in the switch, use the **all** modifier, as shown:

```
interface all;
```

In this instance, all interfaces on the switch would use RIP.

The possible parameters to modify the **interface** statement are:

| | |
|---|---|
| **noripin** | Specifies that RIP packets received from the specified interface are ignored. The default is to listen to RIP packets on all non-loopback interfaces. |
| **ripin** | Specifies that RIP packets received from the specified interface are followed, and is the default mode of operation. |
| **noripout** | Specifies that no RIP packets will be sent on the specified interfaces when in broadcast mode. The default is to send RIP on all broadcast and non-broadcast interfaces when in broadcast mode. |
| **ripout** | Specifies that RIP packets are sent on the specified interface when in broadcast mode, and is the default. This argument is necessary to send RIP on point-to-point interfaces. |
| **metricin** *metric* | Specifies the RIP metric to add to incoming routes before the routes are installed in the routing table. The default metric is the kernel interface metric plus 1 (which is the default RIP hop count). If this value is specified, it will be used as the absolute value, and the kernel metric will not be added. This option is used to make RIP routes learned from the specified interface(s) less preferred than RIP routes learned from other interfaces. |
| **metricout** *metric* | Specifies the RIP metric added to routes that are sent from the specified interface. The default is zero. This option is used to make network routers prefer other sources of RIP routes over this router. |
| **version 1** | Specifies that the RIP packets sent on the specified interface(s) will be version 1 packets. This is the default. |
| **version 2** | Specifies that RIP version 2 packets will be sent on the specified interfaces. If IP multicast support is available on the specified interface, the default is to send full version 2 packets. If multicast support is not available, version 1 compatible version 2 packets will be sent. |
| **multicast** | Specifies that RIP version 2 packets should be multicast on this interface. This is the default if version 2 is specified. |
| **broadcast** | Specifies that RIP 1-compatible RIP version 2 packets should be broadcast on this interface, even if IP multicast is available. |
| **authentication** | This defines the authentication type to use. It applies only to RIP version 2 and is ignored for RIP 1 packets. There are three options: **none**, **simple**, and **md5**. The default authentication type is none. The password should be a quoted string with between 0 and 16 characters.<br><br>If **secondary** is specified, this defines the secondary authentication. Secondary authentication is second authentication that is run after the primary authentication. If omitted, the primary authentication is specified. The default is primary authentication of none and no secondary authentication. |

**trustedgateways** *gateway_list*

This command defines the list of gateways from which RIP will accept updates. The *gateway_list* is simply a list of host names or IP addresses. By default, all routers on the shared network are trusted to supply routing information. If the **trustedgateways** command is specified, only updates from the gateways in the list are accepted.

**sourcegateways** *gateway_list*

This command defines a list of routers to which RIP sends packets directly, not through multicast or broadcast. This list can be used to send different routing information to specific gateways. Updates to gateways in this list are not affected by the **noripout** statement on this interface.

**traceoptions** *trace_file_option trace_options*

This command specifies the tracing options for RIP. Tracing options log information whenever a new route is announced, or the metric being announced changes. Specific RIP tracing options are:

| | |
|---|---|
| **packets** | All RIP packets. |
| **request** | RIP information request packets, such as REQUEST, POLL and POLLENTRY |
| **response** | RIP RESPONSE packets, which are the type of packet that actually contains routing information. |
| **other** | Any other type of packet. The only valid ones are TRACE_ON and TRACE_OFF, both of which are ignored. |

For more information on setting tracing options, see Chapter 12 of this manual.

# RIP Statement Example

The following is an example of an RIP statement, with an explanation of the characteristics of the individual statements:

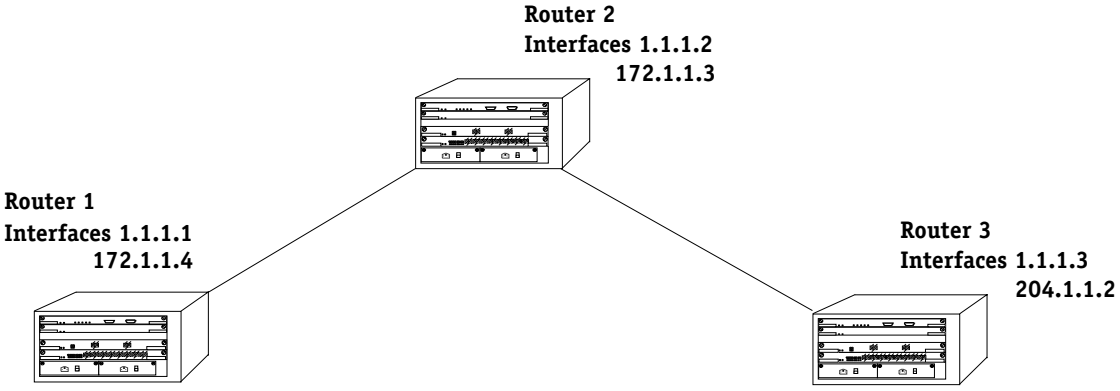
```
rip yes {
    broadcast ;
    defaultmetric 5 ;
    preference 80 ;
    interface 1.1.1.2 noripin version 2 authentication simple “test”;
    interface 1.1.1.3 noripin version 2 authentication simple “test”;
    interface 1.1.1.4 noripin version 2 authentication simple “test”;
} ;
```

In the above RIP statement:

- The **rip yes** statement activates the RIP protocol.
- There are several modifiers for this RIP protocol. The **broadcast** token specifies that RIP packets are to be broadcast even if only one interface is present.
- The **defaultmetric** statement specifies that routes learned from other protocols in the network are advertised with a metric of **5**.
- The **interface** statement defines interfaces 1.1.1.2, 1.1.1.3, and 1.1.1.4 are using RIP, and the **noripin** token specifies that RIP packets received on these interfaces are ignored. The specified interfaces are using **version 2** of RIP, and use **simple** authentication with a password of **test**.

# Application Example

The following diagram represents a simple network using GateD and RIP to route information, with an example of configuration files for each router:




Routers 1, 2, and 3 are part of a LAN, with each router responsible for a separate network. All three routers have two interfaces configured. RIP traffic from Router 1 is not very heavy, while connections to Router 3 are known to be slow. Network 204.1.1.0 needs to be excluded from participating in RIP updates. All routers are using RIP version 2, and simple authentication has been activated.

The configuration files for the routers could look like the following:

## Router 1

```
rip on {
    interface 1.1.1.1 version 2
    authentication simple test secondary authenticate simple “test2”;
};
```

- RIP is activated with the **rip on** statement.
- Interface **1.1.1.1** is defined.
- Two simple authentications are in place for interface 1.1.1.1, one with a password of **test** and the other with a password of **test2**.

## Router 2

```
interfaces {
    1.1.1.1 passive;
} ;
rip on {
    interface 1.1.1.2 version 2
    authentication simple test secondary authenticate simple “test2”;
    interface 172.1.1.3 version 2 metricin 5
    authentication simple test secondary authenticate simple “test2”;
};
```

- Since RIP traffic on Router 1 is light, GateD may mark interface **1.1.1.1** as down. To prevent this, an interface statement is used to mark interface **1.1.1.1** as **passive**. This prevents the interface from being labeled down due to lack of traffic.
- RIP is activated with the **rip on** statement.
- Interface 1.1.1.2 is defined.

- Two simple authentications are in place for interface 1.1.1.2, one with a password of **test** and the other with a password of **test2**.
- Interface 172.1.1.3 is defined.
- Since Router 3 is known to be slow, the metricin statement artificially inflates the routes learned from interface 172.1.1.3 (as it is connected to Router 3), because these routes may not be the best path for traffic.
- Two simple authentications are in place for interface 1.1.1.2, one with a password of **test** and the other with a password of **test2**.

## Router 3

```
martians {
    204.1.1.0;
} ;
rip on {
    interface 1.1.1.1 version 2
    authentication simple “test” secondary authenticate simple “test2”;
    interface 204.1.1.2 version 2
    authentication simple “test” secondary authenticate simple “test2”;
};
```

- Traffic from address **204.1.1.0** must be excluded from the RIP routing table. The **martian** statement prevents any routes from this interface from being learned.
- RIP is activated with the **rip on** statement.
- Interfaces **172.1.1.1** and **204.1.1.2** are defined.
- Two simple authentications are in place for interface **1.1.1.2** and **204.1.1.2**, one with a password of **test** and the other with a password of **test2**.

The syntax conventions for the GateD configuration file are described in detail in Chapter 3 of this manual.

**♦ Note ♦**

Some of the statements in the above example have been separated with **<enter>** (i.e., they are on separate lines). It is important to remember that GateD does not recognize spaces or line feeds. The end of a statement is determined by the placement of the semi-colon (;).