Chapter 23
Routing Information Protocol, next generation (RIPng)

defaultmetric

Name

`defaultmetric` - defines the metric Multi-Exit Discriminator (MED) used when advertising routes via RIPng

Syntax

`defaultmetric` *`metric`* **`;`**

Parameters

`metric` **-** a RIPng metric from 0 to 15

Description

When RIPng learns a route from another protocol, the metrics of the two protocols are almost certainly incompatible. Most protocols have a much larger metric space (for example, 0 to 65535) than RIPng does, so there must be a mechanism for specifying the initial metric that the RIPng advertiser will use.

`defaultmetric` is used to set the default metric for advertising into the RIPng cloud routes learned from other protocols, including **`static`**. To set this metric for exporting routes to RIPng on a per-protocol basis, use the **`export`** statement.

Default

`defaultmetric 1 ;`

Context

`ripng` statement

Examples

The following example sets the default metric for routes exported to RIPng to be 2.

```
ripng on {
    defaultmetric 2 ;
} ;
```

See Also

`export` on page 623

`ripng` on page 568

expire-time

Name

`expire-time` - sets the expiration time for routes received via RIPng

Syntax

`expire-time` *`expire_time`***`;`**

Parameters

`expire_time` - an integer from 5 to 180, inclusive

Description

`expire-time` is initialized when a route is established, and any time an update message is received for the route. If *`expire_time`* seconds elapse from the last time the **`expire-time`** was initialized, the route is considered to have expired, and the deletion process begins for that route. This is a **`group`** option in the **`ripng`** clause. For example, it is used in the same context as **`preference`**.

Defaults

```
expire-time 180;
```

Context

`ripng` statement

Examples

```
ripng yes {
     expire-time 100;
     interface fxp0;
};
```

See Also

`update-time` on page 575

interface

Name

`interface` - controls various attributes of sending RIPng on specific interfaces

Syntax

```
interface interface_list [ noripin ] | [ ripin ] [ noripout ] |
  [ ripout ] [ metricin ] | [ metricout ] ;
```

Parameters

`interface_list` - one or more interface names, including wildcard names (names without a number) and names that may specify more than one interface or address, or the token **`all`** for all interfaces

`noripin` - specifies that RIPng packets received via the specified interface will be ignored.

`ripin -` specifies that RIPng packets on all non-loopback interfaces will be listened to. Specifying **`ripin`** may be necessary when **`noripin`** is used on a wildcard interface descriptor.

`noripout -` specifies that no RIPng packets will be sent on the specified interfaces. The sending of RIPng on point-to-point interfaces must be manually configured.

`ripout -` Specifying **`ripout`** is necessary to send RIPng on point-to-point interfaces and can be necessary when **`noripin`** is used on a wildcard interface descriptor.

`metricin -` sets an additional metric on incoming RIPng routes

`metricout -` specifies an additional cost to be added to outgoing routes

Description

`interface` controls various attributes of sending RIPng on specific interfaces. It is used both to set interface-specific parameters in RIPng and to determine on which interfaces RIPng will run. By default, RIPng will run on all interfaces; however, if any specific interfaces (or subsets of interfaces) are explicitly configured with **`interface`**, then all non-configured interfaces will not run RIPng.

If multiple interfaces (either physical or logical) have addresses on the same subnet, then (regardless of configuration) RIPng will send updates only on the first one for which RIPng is configured to do so.

Although it is possible to specify a loopback interface or loopback address in an interface statement, RIPng will not normally send packets to a loopback. To override this behavior, use **`source-gateways`** with the loopback address included in the *`gatewaylist`*.

Notes:

- When specifying a link-local address as the interface, the interface index must be stripped from the interface.
- If there are multiple interfaces configured on the same subnet, RIPng updates will only be sent from the first one for which RIPng output is configured. This limitation is required because of the way the UNIX kernel operates.

Default

interface *interface_list* **ripin ripout ;**

Context

ripng statement

Examples

Example 1

The following will configure RIPng to run on just the eth0 interface (and not any others).

```
ripng on {
    interface eth0;
};
```

Example 2

Alternatively, this example will set RIPng to run on all interfaces except eth0.

```
ripng on {
    interface all ;
    interface eth0 noripin noripout;
};
```

See Also

metricin on page 562

metricout on page 564

noripin on page 570

noripout on page 571

ripng on page 568

"Chapter 5 Interface Statements" on page 15 of the *GateD Command Reference Guide*

"Chapter 7 Interface Statement" on page 23 of *Configuring GateD*

metricin

Name

`metricin` - sets an additional metric on incoming RIPng routes

Syntax

`metricin` *`ripngmetric`*

Parameters

`ripngmetric` - a number signifying hop count, from 0 to 15

Description

It is often the case that a router should prefer routes received on one set of interfaces over those received on another. For example, given two point-to-point links, one can be more expensive than the other and should, therefore, be less preferred. **`metricin`** is used for exactly this purpose: to make routes learned from certain interfaces less preferable.

`metricin` is the default manner by which RIPng increments hop count. That is to say, RIPng works by adding a hop every time a route is received and before it is sent back out to other interfaces. This implementation adds the hop when the route is received (for example, before decisions regarding whether the route should be used are made). By default, a metric of 1 plus the kernel interface metric is added as the hop count. Normally, this interface metric is zero, but some operating systems allow it to be specified on interface configuration. If **`metricin`** is explicitly given, it is added as an absolute value (for example, without the interface metric).

Defaults

If left unspecified, a metric of 1 plus the kernel interface metric (if any) is the default.

Context

`ripng interface` statement

Examples

Example 1

To override any specified interface metric, the following adds exactly 1 to the metric of all received routes.

```
ripng on {
    interface all metricin 1;
};
```

Example 2

In somewhat more normal usage, the following example increases the cost of routes by 2 that are received over a point-to-point link more than those received on other interfaces.

```
ripng on {
        interface all metricin 1;
        interface ppp0 metricin 3;
    };
```

See Also

`interface` on page 560

`metricout` on page 564

`ripin` on page 570

`ripout` on page 571

metricout

Name

`metricout` - specifies an additional cost to be added to outgoing routes

Syntax

`metricout` *`ripngmetric`*

Parameters

`ripngmetric` - a number signifying hop count, from 0 to 15

Description

Normally, this RIPng implementation adds to the hop count only on incoming routes. There are times, however, when the user wants to cause other routers not to prefer routes from a given origin. For example, if the router is a backup router, it might be desirable for its routes to always be less preferred. **`metricout`** accomplishes this by adding to the RIPng metric on top of any metric specified by **`metricin`** before RIPng updates are sent out the specified interface.

Defaults

```
metricout 0 ;
```

Context

`ripng interface` statement

Examples

Example 1

If this router (host 128) is acting as backup on 192.168.10/24 for all other interfaces, the following will add 2 to all metrics advertised out the 192.168.10.128 interface.

```
ripng on {
    interface all ripin ripout;
    interface 192.168.10.128 metricout 2;
};
```

Example 2

The following example has no effect. It is contrived to point out that **`metricout`** does not impact routing based on the receipt of updates, so if updates do not go out an interface on which **`metricout`** is specified, the behavior of the entire network will be identical to what it would have been without **`metricout`** being issued.

```
ripng on {
    interface all ripin ripout;
    interface eth0 ripin noripout metricout 15;
    };
```

See Also

`interface` on page 560

`metricin` on page 562

`noripin` on page 570

`noripout` on page 571

`ripin` on page 570

`ripout` on page 571

preference

Name

`preference` - sets the preference for RIPng routes

Syntax

`preference` *`ripngpreference`*

Parameters

`ripngpreference` - an assigned integer between 0 (directly connected) and 255 (for EGP)

Description

`preference` specifies how active routes that are learned from RIPng (compared to other protocols) will be selected. When a route has been learned from more than one protocol, the active route will be selected from the protocol with the lowest preference. Each protocol has a default preference in this selection. **`preference`** can be used to change the default value for RIPng. **`preference`** can be overridden by a **`preference`** value specified in import policy.

Default

`preference 100 ;`

Context

`ripng` statement

Examples

Example 1

The following example makes RIPng routes more preferable than IS-IS routes, but less preferable than OSPF routes.

```
ripng on {
    preference 13;
};
```

Example 2

The following example makes RIPng routes less preferable than OSPFase routes, but still more preferable than BGP routes.

```
ripng on {
    preference 160;
};
```

See Also

"Preferences and Route Selection" on page 11 in *Configuring GateD*

`ripng` on page 568

ripng

Name

`ripng` - an implementation of a distance-vector routing protocol for local networks

Syntax

`ripng ( on | off ) [ {` *`ripargs`* **`} ] ;`**

Parameters

`ripngargs` **-** RIPng configuration parameters, which are described throughout this chapter.

Description

`ripng` is an implementation of a distance-vector, or Bellman-Ford, routing protocol for local networks. RIPng is based off the RIP protocol and inherits the limitations and constraints that are in RIP.

A router running RIPng sends an update to its neighbor routers every 30 seconds. Each update contains paired values where each pair consists of an IPv6 network address and an integer distance to that network. RIPng uses a hop count metric to measure the distance to a destination. In the RIPng metric, a router advertises directly connected networks at a metric of 1 by default. Networks that are reachable through one other gateway are 2 hops, and so on. 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. 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 than 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.

At startup, RIPng issues a request for routing information and then listens for responses to the request. If a system configured to supply RIPng 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 RIPng response packet is received, the routing daemon takes the information and rebuilds the routing database, adding new routes and "better" (lower metric) routes to destinations already listed in the database. RIPng also deletes routes from the database if the next router to that destination reports that 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.

Default

`ripng` is run on all interfaces.

Context

global

Examples

The following example configures the identical behavior as not having a `gated.conf` file would (run RIPng on all interfaces).

```
ripng on ;
```

See Also

`aggregate` on page 673

`dampen-flap` on page 707

`export` on page 623

"Routing Information Protocol, next generation (RIPng)" on page 127 in *Configuring GateD*

ripin, noripin

Name

ripin, noripin - specifies whether RIPng will listen to RIPng updates

Syntax

ripin

noripin

Parameters

none

Description

ripin specifies that RIPng will process RIPng updates received on a given interface. Since this is also the default, it is not normally required, except to override a **noripin** specified on a wildcard list of interfaces. **noripin** does exactly the opposite. Although it would almost certainly be a misconfiguration, it is important to note that RIPng can send RIPng updates on a superset of those interfaces on which it receives updates. This can be a valid configuration if, for example, the user receives RIPng updates from an ISP, redistributing those onto the LAN, and does not want to send the LAN topology back to the ISP. But this would be highly unusual.

Defaults

ripin

Context

ripng interface statement

Examples

This somewhat contrived example processes RIPng updates received on all eth devices except eth0.

```
ripng on {
    interface eth ripin;
    interface eth0 noripin;
};
```

See Also

interface on page 560

metricin on page 562

metricout on page 564

ripout on page 571

ripout, noripout

Name

ripout, noripout - specifies whether RIPng will send RIPng updates

Syntax

ripout

noripout

Parameters

none

Description

noripout specifies that RIPng updates should not be sent on the specified interfaces. **ripout** specifies the converse and is the default on broadcast-enabled interfaces.

Defaults

ripout on broadcast interfaces

noripout on nonbroadcast interfaces (including point-to-point interfaces)

Context

ripng interface statement

Examples

Example 1

The following example specifies that RIPng updates should not be sent on any eth interfaces except eth0.

```
ripng on {
    interface eth noripout;
    interface eth0 ripout;
};
```

Example 2

The following example specifies that RIPng updates should be sent on all eth and ppp interfaces.

```
ripng on {
    interface eth ripout;
    interface ppp ripout;
};
```

See Also

`interface` on page 560

`metricin` on page 562

`metricout` on page 564

`ripin` on page 570

traceoptions

Name

`traceoptions` - specifies the tracing options for RIPng

Syntax

`traceoptions` *`trace_options`*

Parameters

Trace options include:

`packets` - Trace all RIPng packets.

`request` - Trace RIPng information request packets, such as **`request`**, **`poll`**, and **`pollentry`**.

`response -` Trace RIPng **`response`** packets, which are the type of packet that actually contains routing information.

`other -` Trace any other type of packet. The only valid ones are **`trace_on`** and **`trace_off`**, both of which are ignored.

Description

`traceoptions` specifies the tracing options for this group. By default, these are inherited from the global trace options.

Default

inherited from global **`traceoptions`**

Context

`ripng` statement

Examples

Example 1

```
traceoptions none;
```

Example 2

```
traceoptions /var/tmp/ripng_peer1 detail packets;
```

Example 3

```
traceoptions receive request;
```

Example 4

```
traceoptions send response;
```

See Also

`ripng` on page 568

update-time

Name

`update-time` - sets the update time for unsolicited route response

Syntax

`update-time` *`update_time`***`;`**

Parameters

`update_time` - an integer from 0 to 30, inclusive

Description

This is a group option in the RIPng clause. For example, it is used in the same context as **`preference`**.

Defaults

```
update-time 30;
```

Context

`ripng` statement

Examples

```
ripng on {
    update-time 20;
    interface fxp0;
};
```

See Also

`expire-time` on page 559