Implementing RIP

The Routing Information Protocol (RIP) is a classic distance vector Interior Gateway Protocol (IGP) designed to exchange information within an autonomous system (AS) of a small network.

This module describes the concepts and tasks to implement basic RIP routing. Cisco IOS XR software supports a standard implementation of RIP Version 2 (RIPv2) that supports backward compatibility with RIP Version 1 (RIPv1) as specified by RFC 2453.

Feature History for Implementing RIP

Release	Modification
Release 6.0.1	This feature was introduced.

Prerequisites for Implementing RIP

You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

Information About Implementing RIP

RIP Functional Overview

RIP Version 1 (RIP v1) is a classful, distance-vector protocol that is considered the easiest routing protocol to implement. Unlike OSPF, RIP broadcasts User Datagram Protocol (UDP) data packets to exchange routing information in internetworks that are flat rather than hierarchical. Network complexity and network management

time is reduced. However, as a classful routing protocol, RIP v1 allows only contiguous blocks of hosts, subnets or networks to be represented by a single route, severely limiting its usefulness.

RIP v2 allows more information carried in RIP update packets, such as support for:

- Route summarization
- Classless interdomain routing (CIDR)
- Variable-length subnet masks (VLSMs)
- Autonomous systems and the use of redistribution

The metric that RIP uses to rate the value of different routes is *hop count.* The hop count is the number of routers that can be traversed in a route. A directly connected network has a metric of zero; an unreachable network has a metric of 16. This small range of metrics makes RIP an unsuitable routing protocol for large networks.

Routing information updates are advertised every 30 seconds by default, and new updates discovered from neighbor routers are stored in a routing table.

Only RIP Version 2 (RIP v2), as specified in RFC 2453, is supported on Cisco IOS XR software and, by default, the software only sends and receives RIP v2 packets. However, you can configure the software to send, or receive, or both, only Version 1 packets or only Version 2 packets or both version type packets per interface.

Here are some good reasons to use RIP:

- Compatible with diverse network devices
- Best for small networks, because there is very little overhead, in terms of bandwidth used, configuration, and management time
- Support for legacy host systems

Because of RIP's ease of use, it is implemented in networks worldwide.

Note VRF does not allow configuration of a VRF group applied directly under router RIP. A VRF group can be configured if it is applied globally or under VRF.

Split Horizon for RIP

Normally, routers that are connected to broadcast-type IP networks and that use distance-vector routing protocols employ the *split horizon* mechanism to reduce the possibility of routing loops. Split horizon blocks information about routes from being advertised by a router out of any interface from which that information originated. This behavior usually optimizes communications among multiple routers, particularly when links are broken.

If an interface is configured with secondary IP addresses and split horizon is enabled, updates might not be sourced by every secondary address. One routing update is sourced per network number unless split horizon is disabled.

Note The split horizon feature is enabled by default. In general, we recommend that you do not change the default state of split horizon unless you are certain that your operation requires the change in order to properly advertise routes.

Route Timers for RIP

RIP uses several timers that determine such variables as the frequency of routing updates, the length of time before a route becomes invalid, and other parameters. You can adjust these timers to tune routing protocol performance to better suit your internetwork needs, by making the following timer adjustments to:

- The rate (time in seconds between updates) at which routing updates are sent
- The interval of time (in seconds) after which a route is declared invalid
- The interval (in seconds) during which routing information regarding better paths is suppressed
- The amount of time (in seconds) that must pass before a route is removed from the RIP topology table
- The amount of time delay between RIP update packets

The first four timer adjustments are configurable by the **timers basic** command. The **output-delay** command changes the amount of time delay between RIP update packets. See Customizing RIP, on page 8 for configuration details.

It also is possible to tune the IP routing support in the software to enable faster convergence of the various IP routing algorithms and quickly drop back to redundant routers, if necessary. The total result is to minimize disruptions to end users of the network in situations in which quick recovery is essential.

Route Redistribution for RIP

Redistribution is a feature that allows different routing domains, to exchange routing information. Networking devices that route between different routing domains are called *boundary routers*, and it is these devices that inject the routes from one routing protocol into another. Routers within a routing domain only have knowledge of routes internal to the domain unless route redistribution is implemented on the boundary routers.

When running RIP in your routing domain, you might find it necessary to use multiple routing protocols within your internetwork and redistribute routes between them. Some common reasons are:

- To advertise routes from other protocols into RIP, such as static, connected, OSPF, and BGP.
- To migrate from RIP to a new Interior Gateway Protocol (IGP).
- To retain routing protocol on some routers to support host systems, but upgrade routers for other department groups.
- To communicate among a mixed-router vendor environment. Basically, you might use a protocol specific to Cisco in one portion of your network and use RIP to communicate with devices other than Cisco devices.

Further, route redistribution gives a company the ability to run different routing protocols in work groups or areas in which each is particularly effective. By not restricting customers to using only a single routing protocol,

Cisco IOS XR route redistribution is a powerful feature that minimizes cost, while maximizing technical advantage through diversity.

When it comes to implementing route redistribution in your internetwork, it can be very simple or very complex. An example of a simple one-way redistribution is to log into a router on which RIP is enabled and use the **redistribute static** command to advertise only the static connections to the backbone network to pass through the RIP network. For complex cases in which you must consider routing loops, incompatible routing information, and inconsistent convergence time, you must determine why these problems occur by examining how Cisco routers select the best path when more than one routing protocol is running administrative cost.

Default Administrative Distances for RIP

Administrative distance is used as a measure of the trustworthiness of the source of the IP routing information. When a dynamic routing protocol such as RIP is configured, and you want to use the redistribution feature to exchange routing information, it is important to know the default administrative distances for other route sources so that you can set the appropriate distance weight.

This table lists the Default Administrative Distances of Routing Protocols.

Table 1: Default Administrative Distances of Routing Protocols

Routing Protocols	Administrative Distance Value
Connected interface	0
Static route out an interface	0
Static route to next hop	1
External BGP	20
OSPF	110
IS-IS	115
RIP version 1 and 2	120
Internal BGP	200
Unknown	255

An administrative distance is an integer from 0 to 255. In general, the higher the value, the lower the trust rating. An administrative distance of 255 means the routing information source cannot be trusted at all and should be ignored. Administrative distance values are subjective; there is no quantitative method for choosing them.

Routing Policy Options for RIP

Route policies comprise series of statements and expressions that are bracketed with the **route-policy** and **end-policy** keywords. Rather than a collection of individual commands (one for each line), the statements within a route policy have context relative to each other. Thus, instead of each line being an individual command, each policy or set is an independent configuration object that can be used, entered, and manipulated as a unit.

Each line of a policy configuration is a logical subunit. At least one new line must follow the **then** , **else** , and **end-policy** keywords. A new line must also follow the closing parenthesis of a parameter list and the name string in a reference to an AS path set, community set, extended community set, or prefix set. At least one new line must precede the definition of a route policy, AS path set, community set, extended community set, or prefix set. One or more new lines can follow an action statement. One or more new lines can follow a comma separator in a named AS path set, community set, extended community set, or prefix set. A new line must appear at the end of a logical unit of policy expression and may not appear anywhere else.

Authentication Using Keychain in RIP

Authentication using keychain in Cisco IOS XR Routing Information Protocol (RIP) provides mechanism to authenticate all RIP protocol traffic on RIP interface, based keychain authentication. This mechanism uses the Cisco IOS XR security keychain infrastructure to store and retrieve secret keys and use it to authenticate in-bound and out-going traffic on per-interface basis.

Keychain management is a common method of authentication to configure shared secrets on all entities that exchange secrets such as keys, before establishing trust with each other. Routing protocols and network management applications on Cisco IOS XR software often use authentication to enhance security while communicating with peers.

Tip The Cisco IOS XR software system security component implements various system security features including keychain management. Refer these documents for detailed information on keychain management concepts, configuration tasks, examples, and command used to configure keychain management.

- *Implementing Keychain Management* module in *System Security Configuration Guide for Cisco NCS 5000 Series Routers*
- *Keychain Management Commands* module in *System Security Command Reference for Cisco NCS 5000 Series Routers*

Note The keychain by itself has no relevance; therefore, it must be used by an application that needs to communicate by using the keys (for authentication) with its peers. The keychain provides a secure mechanism to handle the keys and rollover based on the lifetime. The Cisco IOS XR keychain infrastructure takes care of the hit-less rollover of the secret keys in the keychain.

Once you have configured a keychain in the IOS XR keychain database and if the same has been configured on a particular RIP interface, it will be used for authenticating all incoming and outgoing RIP traffic on that interface. Unless an authentication keychain is configured on a RIP interface (on the default VRF or a non-default VRF), all RIP traffic will be assumed to be authentic and authentication mechanisms for in-bound RIP traffic and out-bound RIP traffic will be not be employed to secure it.

RIP employs two modes of authentication: keyed message digest mode and clear text mode. Use the **authentication keychain** *keychain-name* **mode** {**md5** | **text**} command to configure authentication using the keychain mechanism.

In cases where a keychain has been configured on RIP interface but the keychain is actually not configured in the keychain database or keychain is not configured with MD5 cryptographic algorithm, all incoming RIP packets on the interface will be dropped. Outgoing packets will be sent without any authentication data.

In-bound RIP Traffic on an Interface

These are the verification criteria for all in-bound RIP packets on a RIP interface when the interface is configured with a keychain.

If...	Then...
The keychain configured on the RIP interface does not exist in the keychain database...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The keychain is not configured with a MD5 cryptographic algorithm...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The Address Family Identifier of the first (and only the first) entry in the message is not 0xFFFF, then authentication is not in use...	The packet will be dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
The MD5 digest in the 'Authentication Data' is found to be invalid...	The packet is dropped. A RIP component-level debug message is be logged to provide the specific details of the authentication failure.
Else, the packet is forwarded for the rest of the processing.	

Out-bound RIP Traffic on an Interface

These are the verification criteria for all out-bound RIP packets on a RIP interface when the interface is configured with a keychain.

If...	Then
The keychain configured on the RIP interface exists in the keychain database ...	The RIP packet passes authentication check at the remote/peer end, provided the remote router is also configured to authenticate the packets using the same keychain.
The keychain is configured with a MD5 cryptographic algorithm...	The RIP packet passes authentication check at the remote/peer end, provided the remote router is also configured to authenticate the packets using the same keychain.
Else, RIP packets fail authentication check.	

How to Implement RIP

This section contains instructions for the following tasks:

Note To save configuration changes, you must commit changes when the system prompts you.

Enabling RIP

This task enables RIP routing and establishes a RIP routing process.

Before you begin

Although you can configure RIP before you configure an IP address, no RIP routing occurs until at least one IP address is configured.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **neighbor** *ip-address*
4. **broadcast-for-v2**
5. **interface** *type interface-path-id*
6. **receive version** { **1** | **2** | **1 2** }
7. **send version** { **1** | **2** | **1 2** }
8. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Configures a RIP routing process.
Step 3	**neighbor** *ip-address* **Example:** `RP/0/RP0/CPU0:router(config-rip)# neighbor 172.160.1.2`	(Optional) Defines a neighboring router with which to exchange RIP protocol information.
Step 4	**broadcast-for-v2** **Example:** `RP/0/RP0/CPU0:router(config-rip)# broadcast-for-v2`	(Optional) Configures RIP to send only Version 2 packets to the broadcast IP address. This command can be applied at the interface or level.
Step 5	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-rip)# interface HundredGigE 0/1/0/3`	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 6	**receive version** { **1** | **2** | **1 2** } **Example:**	(Optional) Configures an interface to accept packets that are:

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-rip-if)# receive version 1 2`	• Only RIP v1 • Only RIP v2 • Both RIP v1 and RIP v2
Step 7	**send version** { **1** | **2** | **1 2** } **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# send version 1 2`	(Optional) Configures an interface to send packets that are: • Only RIP v1 • Only RIP v2 • Both RIP v1 and RIP v2
Step 8	**commit**	

Customizing RIP

This task describes how to customize RIP for network timing and the acceptance of route entries.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **auto-summary**
4. **timers basic** *update invalid holddown flush*
5. **output-delay** *delay*
6. **nsf**
7. **interface** *type interface-path-id*
8. **metric-zero-accept**
9. **split-horizon disable**
10. **poison-reverse**
11. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Configures a RIP routing process.
Step 3	**auto-summary** **Example:** `RP/0/RP0/CPU0:router(config-rip)# auto-summary`	(Optional) Enables automatic route summarization of subnet routes into network-level routes. • By default, auto-summary is disabled.

	Command or Action	Purpose
		Note If you have disconnected subnets, use the **no** keyword to disable automatic route summarization and permit software to send subnet and host routing information across classful network boundaries.
Step 4	**timers basic** *update invalid holddown flush* **Example:** `RP/0/RP0/CPU0:router(config-rip)# timers basic 5 15 15 30`	(Optional) Adjusts RIP network timers. **Note** To view the current and default timer values, view output from the **show rip** command.
Step 5	**output-delay** *delay* **Example:** `RP/0/RP0/CPU0:router(config-rip)# output-delay 10`	(Optional) Changes the interpacket delay for the RIP updates sent. **Note** Use this command if you have a high-end router sending at high speed to a low-speed router that might not be able to receive at that fast a rate.
Step 6	**nsf** **Example:** `RP/0/RP0/CPU0:router(config-rip)# nsf`	(Optional) ensures continuous forwarding even after a RIP process is shutdown or restart.
Step 7	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-rip)# interface HundredGigE 0/1/0/3`	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 8	**metric-zero-accept** **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# metric-zero-accept`	(Optional) Allows the networking device to accept route entries received in update packets with a metric of zero (0). The received route entry is set to a metric of one (1).
Step 9	**split-horizon disable** **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# split-horizon disable`	(Optional) Disables the split horizon mechanism. • By default, split horizon is enabled. • In general, we do not recommend changing the state of the default for the **split-horizon** command, unless you are certain that your application requires a change to properly advertise routes.
Step 10	**poison-reverse** **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# poison-reverse`	Enables poison reverse processing of RIP router updates.

	Command or Action	Purpose
Step 11	**commit**	

Control Routing Information

This task describes how to control or prevent routing update exchange and propagation.

Some reasons to control or prevent routing updates are:

- To slow or stop the update traffic on a WAN link—If you do not control update traffic on an on-demand WAN link, the link remains up constantly. By default, RIP routing updates occur every 30 seconds.
- To prevent routing loops—If you have redundant paths or are redistributing routes into another routing domain, you may want to filter the propagation of one of the paths.
- To filter network received in updates — If you do not want other routers from learning a particular device's interpretation of one or more routes, you can suppress that information.
- To prevent other routers from processing routes dynamically— If you do not want to process routing updates entering the interface, you can suppress that information.
- To preserve bandwidth—You can ensure maximum bandwidth availability for data traffic by reducing unnecessary routing update traffic.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **neighbor** *ip-address*
4. **interface** *type interface-path-id*
5. **passive-interface**
6. exit
7. **interface** *type interface-path-id*
8. **route-policy** { **in** | **out** }
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Configures a RIP routing process.
Step 3	**neighbor** *ip-address* **Example:**	(Optional) Defines a neighboring router with which to exchange RIP protocol information.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-rip)# neighbor 172.160.1.2`	
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-rip)# interface HundredGigE 0/1/0/3`	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 5	**passive-interface** **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# passive-interface`	(Optional) Suppresses the sending of RIP updates on an interface, but not to explicitly configured neighbors.
Step 6	exit **Example:** `RP/0/ /CPU0:router(config-rip-if)# exit`	(Optional) Returns the router to the next higher configuration mode.
Step 7	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-rip)# interface HundredGigE 0/1/0/4`	(Optional) Defines the interfaces on which the RIP routing protocol runs.
Step 8	**route-policy** { **in** | **out** } **Example:** `RP/0/RP0/CPU0:router(config-rip-if)# route-policy out`	(Optional) Applies a routing policy to updates advertised to or received from a RIP neighbor.
Step 9	**commit**	

Creating a Route Policy for RIP

This task defines a route policy and shows how to attach it to an instance of a RIP process. Route policies can be used to:

- Control routes sent and received
- Control which routes are redistributed
- Control origination of the default route

A route policy definition consists of the **route-policy** command and *name* argument followed by a sequence of optional policy statements, and then closes with the **end-policy** command.

A route policy is not useful until it is applied to routes of a routing protocol.

SUMMARY STEPS

1. **configure**
2. **route-policy** *name*
3. **set rip-metric** *number*
4. **end-policy**
5. **commit**
6. **configure**
7. **router rip**
8. **route-policy** *route-policy-name* { **in** | **out** }
9. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**route-policy** *name* **Example:** `RP/0/RP0/CPU0:router(config)# route-policy IN-IPv4`	Defines a route policy and enters route-policy configuration mode.
Step 3	**set rip-metric** *number* **Example:** `RP/0/RP0/CPU0:router(config-rpl)# set rip metric 42`	(Optional) Sets the RIP metric attribute.
Step 4	**end-policy** **Example:** `RP/0/RP0/CPU0:router(config-rpl)# end-policy`	Ends the definition of a route policy and exits route-policy configuration mode.
Step 5	**commit**	
Step 6	**configure**	
Step 7	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)# router rip`	Configures a RIP routing process.
Step 8	**route-policy** *route-policy-name* { **in**	**out** } **Example:** `RP/0/RP0/CPU0:router(config-rip)# route-policy rp1 in`	Applies a routing policy to updates advertised to or received from an RIP neighbor.
Step 9	**commit**	

Configuring RIP Authentication Keychain

Configuring RIP Authentication Keychain for IPv4 Interface on a Non-default VRF

Perform this task to configure a RIP authentication keychain for IPv4 interface on a non-default VRF.

Before you begin

All keychains need to be configured in Cisco IOS XR keychain database using configuration commands described in *Implementing Keychain Management* module of *System Security Configuration Guide for Cisco NCS 5000 Series Routers* before they can be applied to a RIP interface/VRF.

The **authentication keychain** *keychain-name* and **mode md5** configurations will accept the name of a keychain that has not been configured yet in the IOS XR keychain database or a keychain that has been configured in IOS XR keychain database without MD5 cryptographic algorithm. However, in both these cases, all incoming packets on the interface will be dropped and outgoing packets will be sent without authentication data.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **vrf** *vrf_name*
4. **interface** *type interface-path-id*
5. Use one of these commands:
   - **authentication keychain** *keychain-name* **mode md5**
   - **authentication keychain** *keychain-name* **mode text**
6. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)#router rip`	Configures a RIP routing process.
Step 3	**vrf** *vrf_name* **Example:** `RP/0/RP0/CPU0:router(config-rip)#vrf vrf_rip_auth`	Configures a non-default VRF
Step 4	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-rip-vrf)#interface HundredGigE 0/1/0/3`	Defines the interface on which the RIP routing protocol runs.

	Command or Action	Purpose

Step 5	Use one of these commands:
	• **authentication keychain** *keychain-name* **mode md5**
	• **authentication keychain** *keychain-name* **mode text**
	Example:
	`RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode md5`
	Or
	`RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode text`
		Configures an authentication keychain mode for RIP.
		• **md5**—Keyed message digest (md5) authentication mode
		• **text**—Clear text authentication mode

Step 6	**commit**	

Configuring RIP Authentication Keychain for IPv4 Interface on Default VRF

Perform this task to configure a RIP authentication keychain for IPv4 interface (on the default VRF).

Before you begin

All keychains need to be configured in Cisco IOS XR keychain database using configuration commands described in *Implementing Keychain Management* module of *System Security Configuration Guide for Cisco NCS 5000 Series Routers* before they can be applied to a RIP interface/VRF.

The **authentication keychain** *keychain-name* and **mode md5** configurations will accept the name of a keychain that has not been configured yet in the IOS XR keychain database or a keychain that has been configured in IOS XR keychain database without MD5 cryptographic algorithm. However, in both these cases, all incoming packets on the interface will be dropped and outgoing packets will be sent without authentication data.

SUMMARY STEPS

1. **configure**
2. **router rip**
3. **interface** *type interface-path-id*
4. Use one of these commands:
   - **authentication keychain** *keychain-name* **mode md5**
   - **authentication keychain** *keychain-name* **mode text**
5. **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure**	
Step 2	**router rip** **Example:** `RP/0/RP0/CPU0:router(config)#router rip`	Configures a RIP routing process.

	Command or Action	Purpose
Step 3	**interface** *type interface-path-id* **Example:** `RP/0/RP0/CPU0:router(config-rip)#interface HundredGigE 0/1/0/3`	Defines the interface on which the RIP routing protocol runs.

Step 4	Use one of these commands:
	• **authentication keychain** *keychain-name* **mode md5**
	• **authentication keychain** *keychain-name* **mode text**
	Example:
	`RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode md5`
	Or
	`RP/0/RP0/CPU0:router(config-rip-if)#authentication keychain key1 mode text`
		Configures an authentication keychain mode for RIP.
		• **md5**—Keyed message digest (md5) authentication mode
		• **text**—Clear text authentication mode

Step 5	**commit**	

Configuration Examples for Implementing RIP

This section provides the following configuration examples:

Configuring a Basic RIP Configuration: Example

The following example shows two Gigabit Ethernet interfaces configured with RIP.

```
interface TenGigE 0/3/0/0
 ipv4 address 172.16.0.1 255.255.255.0
!

interface TenGigE 0/3/0/1
 ipv4 address 172.16.2.12 255.255.255.0
!

router rip
 interface TenGigE 0/3/0/0
 !
 interface TenGigE 0/3/0/1
 !
!
```

Configuring RIP on the Provider Edge: Example

The following example shows how to configure basic RIP on the PE with two VPN routing and forwarding (VRF) instances.

```
router rip
 interface HundredGigE 0/1/0/3
```

```
 !
 vrf vpn0
  interface HundredGigE 0/1/0/4
  !
 !
 vrf vpn1
  interface HundredGigE 0/1/0/5
  !
 !
!
```

Adjusting RIP Timers for each VRF Instance: Example

The following example shows how to adjust RIP timers for each VPN routing and forwarding (VRF) instance.

For VRF instance vpn0, the **timers basic** command sets updates to be broadcast every 10 seconds. If a router is not heard from in 30 seconds, the route is declared unusable. Further information is suppressed for an additional 30 seconds. At the end of the flush period (45 seconds), the route is flushed from the routing table.

For VRF instance vpn1, timers are adjusted differently: 20, 60, 60, and 70 seconds.

The **output-delay** command changes the interpacket delay for RIP updates to 10 milliseconds on vpn1. The default is that interpacket delay is turned off.

```
router rip
 interface HundredGigE 0/1/0/3
 !
 vrf vpn0
  interface HundredGigE 0/1/0/4
  !
  timers basic 10 30 30 45
 !
 vrf vpn1
  interface HundredGigE 0/1/0/5
  !
  timers basic 20 60 60 70
  output-delay 10
 !
!
```

Configuring Redistribution for RIP: Example

The following example shows how to redistribute Border Gateway Protocol (BGP) and static routes into RIP.

The RIP metric used for redistributed routes is determined by the route policy. If a route policy is not configured or the route policy does not set RIP metric, the metric is determined based on the redistributed protocol. For VPNv4 routes redistributed by BGP, the RIP metric set at the remote PE router is used, if valid.

In all other cases (BGP, IS-IS, OSPF, connected, static), the metric set by the **default-metric** command is used. If a valid metric cannot be determined, then redistribution does not happen.

```
route-policy ripred
  set rip-metric 5
end-policy
!

router rip
```

```
 vrf vpn0
  interface HundredGigE 0/1/0/3
  !
  redistribute connected
  default-metric 3
 !
 vrf vpn1
  interface HundredGigE 0/1/0/4
  !
  redistribute bgp 100 route-policy ripred
  redistribute static
  default-metric 3
 !
!
```

Configuring Route Policies for RIP: Example

The following example shows how to configure inbound and outbound route policies that are used to control which route updates are received by a RIP interface or sent out from a RIP interface.

```
prefix-set pf1
  10.1.0.0/24
end-set
!

prefix-set pf2
  150.10.1.0/24
end-set
!

route-policy policy_in
  if destination in pf1 then
    pass
  endif
end-policy
!

route-policy pass-all
  pass
end-policy
!

route-policy infil
  if destination in pf2 then
    add rip-metric 2
    pass
  endif
end-policy
!

router rip
 interface HundredGigE 0/1/0/3
  route-policy policy_in in
 !
 interface HundredGigE 0/1/0/4
 !
 route-policy infil in
 route-policy pass-all out
```

Configuring Passive Interfaces and Explicit Neighbors for RIP: Example

The following example shows how to configure passive interfaces and explicit neighbors. When an interface is passive, it only accepts routing updates. In other words, no updates are sent out of an interface except to neighbors configured explicitly.

```
router rip
 interface HundredGigE 0/1/0/3
  passive-interface
 !
 interface HundredGigE 0/1/0/4
 !
 neighbor 172.17.0.1
 neighbor 172.18.0.5
!
```