Implementing and Monitoring RIB

Routing Information Base (RIB) is a distributed collection of information about routing connectivity among all nodes of a network. Each router maintains a RIB containing the routing information for that router. RIB stores the best routes from all routing protocols that are running on the system.

Each routing protocol selects its own set of best routes and installs those routes and their attributes in RIB. RIB stores these routes and selects the best ones from among all routing protocols. Those routes are downloaded to the line cards for use in forwarding packets. The acronym RIB is used both to refer to RIB processes and the collection of route data contained within RIB. Within a protocol, routes are selected based on the metrics in use by that protocol. A protocol downloads its best routes (lowest or tied metric) to RIB. RIB selects the best overall route by comparing the administrative distance of the associated protocol.

This module describes how to implement and monitor RIB on your network.

Note VPNv4, VPNv6 and VPN routing and forwarding (VRF) address families will be supported in a future release.

Verify RIB Configuration Using Routing Table

Perform this task to verify the RIB configuration to ensure that RIB is running on the RP and functioning properly by checking the routing table summary and details.

SUMMARY STEPS

1. **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **summary** [**detail**] [**standby**]
2. **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

DETAILED STEPS

Step 1 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **summary** [**detail**] [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route summary
```

Displays route summary information about the specified routing table.

- The default table summarized is the IPv4 unicast routing table.

Step 2 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast
```

Displays more detailed route information about the specified routing table.

- This command is usually issued with an IP address or other optional filters to limit its display. Otherwise, it displays all routes from the default IPv4 unicast routing table, which can result in an extensive list, depending on the configuration of the network.

Output of show route best-local Command: Example

The following is sample output from the **show route backup** command:

```
show route backup

Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
       U - per-user static route, o - ODR, L - local
S    172.73.51.0/24 is directly connected, 2d20h, HundredGigE 4/0/0/1
                Backup  O E2 [110/1] via 10.12.12.2, HundredGigE 3/0/0/1
```

Verify Networking and Routing Problems

Perform this task to verify the operation of routes between nodes.

SUMMARY STEPS

1. **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

2. **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **backup** [*ip-address*] [**standby**]
3. **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **best-local** *ip-address* [**standby**]
4. **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **connected** [**standby**]
5. **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **local** [*interface*] [**standby**]
6. **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **longer-prefixes** { *ip-address mask* | *ip-address* / *prefix-length* } [**standby**]
7. **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **next-hop** *ip-address* [**standby**]

DETAILED STEPS

Step 1 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] [*protocol* [*instance*] | *ip-address mask*] [**standby**] [**detail**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast 192.168.1.11/8
```

Displays the current routes in RIB.

Step 2 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **backup** [*ip-address*] [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast backup 192.168.1.11/8
```

Displays backup routes in RIB.

Step 3 **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **best-local** *ip-address* [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast best-local 192.168.1.11/8
```

Displays the best-local address to use for return packets from the given destination.

Step 4 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **connected** [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast connected
```

Displays the current connected routes of the routing table.

Step 5 **show route** [**vrf** { *vrf-name* | **all** }] [**afi-all** | **ipv4** | **ipv6**] [**unicast** | **safi-all**] **local** [*interface*] [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast local
```

Displays local routes for receive entries in the routing table.

Step 6 **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **longer-prefixes** { *ip-address mask* | *ip-address* / *prefix-length* } [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast longer-prefixes 192.168.1.11/8
```

Displays the current routes in RIB that share a given number of bits with a given network.

Step 7 **show route** [**vrf** { *vrf-name* | **all** }] [**ipv4** | **ipv6**] [**unicast** | **safi-all**] **next-hop** *ip-address* [**standby**]

Example:

```
RP/0/RP0/CPU0:router# show route ipv4 unicast next-hop 192.168.1.34
```

Displays the next-hop gateway or host to a destination address.

Output of show route Command: Example

The following is sample output from the **show route** command when entered without an address:

```
show route

  Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
         O - OSPF, IA - OSPF inter area
         N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
         E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
         i - ISIS, L1 - IS-IS level-1, L2 - IS-IS level-2
         ia - IS-IS inter area, su - IS-IS summary null, * - candidate default
         U - per-user static route, o - ODR, L - local

  Gateway of last resort is 172.23.54.1 to network 0.0.0.0

  C    10.2.210.0/24 is directly connected, 1d21h, Ethernet0/1/0/0
  L    10.2.210.221/32 is directly connected, 1d21h, Ethernet0/1/1/0
  C    172.20.16.0/24 is directly connected, 1d21h, ATM4/0.1
  L    172.20.16.1/32 is directly connected, 1d21h, ATM4/0.1
  C    10.6.100.0/24 is directly connected, 1d21h, Loopback1
  L    10.6.200.21/32 is directly connected, 1d21h, Loopback0
  S    192.168.40.0/24 [1/0] via 172.20.16.6, 1d21h
```

Disable RIB Next-hop Dampening

Perform this task to disable RIB next-hop dampening.

SUMMARY STEPS

1. **router rib**
2. **address-family** { **ipv4** | **ipv6** } **next-hop dampening disable**
3. **commit**

DETAILED STEPS

Step 1 **router rib**

Example:

```
RP/0/RP0/CPU0:router# route rib
```

Enters RIB configuration mode.

Step 2 **address-family { ipv4 | ipv6 } next-hop dampening disable**

Example:

```
RP/0/RP0/CPU0:router(config-rib)# address family ipv4 next-hop dampening disable
```

Disables next-hop dampening for IPv4 address families.

Step 3 **commit**

Output of show route next-hop Command: Example

The following is sample output from the **show route resolving-next-hop** command:

```
  show route resolving-next-hop 10.0.0.1

  Nexthop matches 0.0.0.0/0
    Known via "static", distance 200, metric 0, candidate default path
    Installed Aug 18 00:59:04.448
    Directly connected nexthops
      172.29.52.1, via MgmtEth0/

/CPU0/0
        Route metric is 0
```

Enable RCC and LCC On-demand Scan

Perform this task to trigger route consistency checker (RCC) and Label Consistency Checker (LCC) on-demand scan. The on-demand scan can be run on a particular address family (AFI), sub address family (SAFI), table and prefix, vrf, or all prefixes in the table.

SUMMARY STEPS

1. Use one of these commands.
   - **show rcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*]
   - **show lcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*]
2. Use one of these commands.
   - **clear rcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*] **log**

- **clear lcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*] **log**

DETAILED STEPS

Step 1 Use one of these commands.

- **show rcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*]
- **show lcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*]

Example:

```
RP/0/RP0/CPU0:router#show rcc ipv6 unicast 2001:DB8::/32 vrf vrf_1
```

Or

```
RP/0/RP0/CPU0:router#show lcc ipv6 unicast 2001:DB8::/32 vrf vrf_1
```

Runs on-demand Route Consistency Checker (RCC) or Label Consistency Checker (LCC).

Step 2 Use one of these commands.

- **clear rcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*] **log**
- **clear lcc** {**ipv4** | **ipv6**} **unicast** [**all**] [*prefix*/*mask*] [**vrf** *vrf-name*] **log**

Example:

```
RP/0/RP0/CPU0:router#clear rcc ipv6 unicast log
```

Or

```
RP/0/RP0/CPU0:router#show lcc ipv6 unicast log
```

Clears the log of previous scans.

Enable RCC and LCC Background Scan

Perform this task to run a background scan for Route Consistency Checker (RCC) and Label Consistency Checker (LCC).

SUMMARY STEPS

1. **configure**
2. Use one of these commands:
   - **rcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}
   - **lcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}
3. **commit**
4. Use one of these commands.
   - **show rcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]

- **show lcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]

DETAILED STEPS

Step 1 **configure**

Step 2 Use one of these commands:

- **rcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}
- **lcc** {**ipv4** | **ipv6**} **unicast** {**enable** | **period** *milliseconds*}

Example:

```
RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast enable

RP/0/RP0/CPU0:router(config)#rcc ipv6 unicast period 500
```

Or

```
RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast enable

RP/0/RP0/CPU0:router(config)#lcc ipv6 unicast period 500
```

Triggers RCC or LCC background scan. Use the **period** option to control how often the verification be triggered. Each time the scan is triggered, verification is resumed from where it was left out and one buffer's worth of routes or labels are sent to the forwarding information base (FIB).

Step 3 **commit**

Step 4 Use one of these commands.

- **show rcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]
- **show lcc** {**ipv4**| **ipv6**} **unicast** [**summary** | **scan-id** *scan-id-value*]

Example:

```
RP/0/RP0/CPU0:router#show rcc ipv6 unicast statistics scan-id 120
```

Or

```
RP/0/RP0/CPU0:router#show lcc ipv6 unicast statistics scan-id 120
```

Displays statistics about background scans.

- **summary**—Displays the current ongoing scan id and a summary of the previous few scans.
- **scan-id** *scan-id-value*—Displays details about a specific scan.

Enabling RCC and LCC: Example

This example shows how to enable Route Consistency Checker (RCC) background scan with a period of 500 milliseconds between buffers in scans for IPv6 unicast tables:

```
rcc ipv6 unicast period 500
```

This example shows how to enable Label Consistency Checker (LCC) background scan with a period of 500 milliseconds between buffers in scans for IPv6 unicast tables:

```
lcc ipv6 unicast period 500
```

This example shows how to run Route Consistency Checker (RCC) on-demand scan for subnet 10.10.0.0/16 in vrf1:

```
show rcc ipv4 unicast 10.10.0.0/16 vrf vrf 1
```

This example shows how to run Label Consistency Checker (LCC) on-demand scan on all labels for IPv6 prefixes:

```
show lcc ipv6 unicast all
```

References for RIB

This section provides additional conceptual information on RIB. It includes the following topics:

RIB Data Structures in BGP and Other Protocols

RIB uses processes and maintains data structures distinct from other routing applications, such as Border Gateway Protocol (BGP) and other unicast routing protocols. However, these routing protocols use internal data structures similar to what RIB uses, and may internally refer to the data structures as a RIB. For example, BGP routes are stored in the BGP RIB (BRIB). RIB processes are not responsible for the BRIB, which are handled by BGP.

The table used by the line cards and RP to forward packets is called the Forwarding Information Base (FIB). RIB processes do not build the FIBs. Instead, RIB downloads the set of selected best routes to the FIB processes, by the Bulk Content Downloader (BCDL) process, onto each line card. FIBs are then constructed.

RIB Administrative Distance

Forwarding is done based on the longest prefix match. If you are forwarding a packet destined to 10.0.2.1, you prefer 10.0.2.0/24 over 10.0.0.0/16 because the mask /24 is longer (and more specific) than a /16. Routes from different protocols that have the same prefix and length are chosen based on administrative distance. For instance, the Open Shortest Path First (OSPF) protocol has an administrative distance of 110, and the Intermediate System-to-Intermediate System (IS-IS) protocol has an administrative distance of 115. If IS-IS and OSPF both download 10.0.1.0/24 to RIB, RIB would prefer the OSPF route because OSPF has a lower administrative distance. Administrative distance is used only to choose between multiple routes of the same length.

This table lists default administrative distances for the common protocols.

Table 1: Default Administrative Distances

Protocol	Administrative Distance Default
Connected or local routes	0
Static routes	1
External BGP routes	20
OSPF routes	110
IS-IS routes	115
Internal BGP routes	200

The administrative distance for some routing protocols (for instance IS-IS, OSPF, and BGP) can be changed. See the protocol-specific documentation for the proper method to change the administrative distance of that protocol.

Note Changing the administrative distance of a protocol on some but not all routers can lead to routing loops and other undesirable behavior. Doing so is not recommended.

RIB Statistics

RIB supports statistics for messages (requests) flowing between the RIB and its clients. Protocol clients send messages to the RIB (for example, route add, route delete, and next-hop register, and so on). RIB also sends messages (for example, redistribute routes, advertisements, next-hop notifications, and so on). These statistics are used to gather information about what messages have been sent and the number of messages that have been sent. These statistics provide counters for the various messages that flow between the RIB server and its clients. The statistics are displayed using the **show rib statistics** command.

RIB maintains counters for all requests sent from a client including:

- Route operations
- Table registrations
- Next-hop registrations
- Redistribution registrations
- Attribute registrations
- Synchronization completion

RIB also maintains counters for all requests sent by the RIB. The configuration will disable the RIB next-hop dampening feature. As a result, RIB notifies client immediately when a next hop that client registered for is resolved or unresolved. RIB also maintains the results of the requests.

RIB Quarantining

RIB quarantining solves the problem in the interaction between routing protocols and the RIB. The problem is a persistent oscillation between the RIB and routing protocols that occurs when a route is continuously inserted and then withdrawn from the RIB, resulting in a spike in CPU use until the problem is resolved. If there is no damping on the oscillation, then both the protocol process and the RIB process have high CPU use, affecting the rest of the system as well as blocking out other protocol and RIB operations. This problem occurs when a particular combination of routes is received and installed in the RIB. This problem typically happens as a result of a network misconfiguration. However, because the misconfiguration is across the network, it is not possible to detect the problem at configuration time on any single router.

The quarantining mechanism detects mutually recursive routes and quarantines the last route that completes the mutual recursion. The quarantined route is periodically evaluated to see if the mutual recursion has gone away. If the recursion still exists, the route remains quarantined. If the recursion has gone away, the route is released from its quarantine.

The following steps are used to quarantine a route:

1. RIB detects when a particular problematic path is installed.
2. RIB sends a notification to the protocol that installed the path.
3. When the protocol receives the quarantine notification about the problem route, it marks the route as being “quarantined.” If it is a BGP route, BGP does not advertise reachability for the route to its neighbors.
4. Periodically, RIB tests all its quarantined paths to see if they can now safely be installed (moved from quarantined to "Ok to use" state). A notification is sent to the protocol to indicate that the path is now safe to use.

Route and Label Consistency Checker

The Route Consistency Checker and Label Consistency Checker (RCC/LCC) are command-line tools that can be used to verify consistency between control plane and data plane route and label programming in IOS XR software.

Routers in production networks may end up in a state where the forwarding information does not match the control plane information. Possible causes of this include fabric or transport failures between the Route Processor (RP) and the line cards (LCs), or issues with the Forwarding Information Base (FIB). RCC/LCC can be used to identify and provide detailed information about resultant inconsistencies between the control plane and data plane. This information can be used to further investigate and diagnose the cause of forwarding problems and traffic loss.

RCC/LCC can be run in two modes. It can be triggered from using the appropriate command modes as an on-demand, one-time scan (On-demand Scan), or be configured to run at defined intervals in the background during normal router operation (Background Scan). RCC compares the Routing Information Base (RIB) against the Forwarding Information Base (FIB) while LCC compares the Label Switching Database (LSD) against the FIB. When an inconsistency is detected, RCC/LCC output will identify the specific route or label and identify the type of inconsistency detected as well as provide additional data that will assist with further troubleshooting.

RCC runs on the Route Processor. FIB checks for errors on the line card and forwards first the 20 error reports to RCC. RCC receives error reports from all nodes, summarizes them (checks for exact match), and adds it to two queues, soft or hard. Each queue has a limit of 1000 error reports and there is no prioritization in the

queue. RCC/LCC logs the same errors (exact match) from different nodes as one error. RCC/LCC compares the errors based on prefix/label, version number, type of error, etc.

On-demand Scan

In On-demand Scan, user requests scan through the command line interface on a particular prefix in a particular table or all the prefixes in the table. The scan is run immediately and the results are published right away. LCC performs on-demand scan on the LSD, where as RCC performs it per VRF.

Background Scan

In Background Scan, user configures the scan that is then left to run in the background. The configuration consists of the time period for the periodic scan. This scan can be configured on either a single table or multiple tables. LCC performs background scan on the LSD, where as RCC performs it either for default or other VRFs.