

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 25, 2016

L. Wang
Individual
H. Ananthakrishnan
Packet Design
M. Chen
Huawei
A. Dass
S. Kini
Ericsson
N. Bahadur
Bracket Computing
November 22, 2015

**A YANG Data Model for Routing Information Base (RIB)
draft-ietf-i2rs-rib-data-model-04**

Abstract

This document defines a YANG data model for Routing Information Base (RIB) that aligns with the I2RS RIB information model.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 25, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](http://trustee.ietf.org/license-info) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

- [1. Introduction](#) [2](#)
- [1.1. Definitions and Acronyms](#) [3](#)
- [1.2. Tree Diagrams](#) [3](#)
- [2. Model Structure](#) [3](#)
- [2.1. RIB Capability](#) [7](#)
- [2.2. Routing Instance and Rib](#) [8](#)
- [2.3. Route](#) [8](#)
- [2.4. Nexthop](#) [9](#)
- [2.5. RPC Operations](#) [14](#)
- [2.6. Notifications](#) [18](#)
- [3. YANG Modules](#) [19](#)
- [4. IANA Considerations](#) [63](#)
- [5. Security Considerations](#) [63](#)
- [6. Contributors](#) [63](#)
- [7. References](#) [64](#)
- [7.1. Normative References](#) [64](#)
- [7.2. Informative References](#) [64](#)
- Authors' Addresses [64](#)

1. Introduction

The Interface to the Routing System (I2RS) [[I-D.ietf-i2rs-architecture](#)] provides read and write access to the information and state within the routing process that exists inside the routing elements, this is achieved via the protocol message exchange between I2RS clients and I2RS agents associated with the routing system. One of the functions of I2RS is to read and write data of Routing Information Base (RIB). [[I-D.ietf-i2rs-usecase-reqs-summary](#)] introduces a set of RIB use cases. The RIB information model is defined in [[I-D.ietf-i2rs-rib-info-model](#)].

This document defines a YANG [[RFC6020](#)][RFC6991] data model for the RIB that satisfies the RIB use cases and aligns with the RIB information model.

1.1. Definitions and Acronyms

RIB: Routing Information Base

Information Model (IM): An abstract model of a conceptual domain, independent of a specific implementation or data representation.

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Model Structure

The following figure shows an overview of structure tree of the ietf-i2rs-rib module. To give a whole view of the structure tree, some details of the tree are omitted. The relevant details are introduced in the following sub-sections.

```

module: ietf-i2rs-rib
  +--rw routing-instance
    +--rw name string
    +--rw interface-list* [name]
      | +--rw name if:interface-ref
    +--rw router-id? yang:dotted-quad
    +--rw lookup-limit? uint8
    +--rw rib-list* [name]
      +--rw name string
      +--rw rib-family rib-family-def
  
```



```

+--rw ip-rpf-check?  boolean
+--rw route-list* [route-index]
  +--rw route-index          uint64
  +--rw match
    | +--rw (route-type)?
    |   +--:(ipv4)
    |   | ...
    |   +--:(ipv6)
    |   | ...
    |   +--:(mpls-route)
    |   | ...
    |   +--:(mac-route)
    |   | ...
    |   +--:(interface-route)
    |   | ...
  +--rw nexthop
    | +--rw nexthop-id?          uint32
    | +--rw sharing-flag?       boolean
    | +--rw (nexthop-type)?
    |   +--:(nexthop-base)
    |   | ...
    |   +--:(nexthop-chain) {nexthop-chain}?
    |   | ...
    |   +--:(nexthop-replicates) {nexthop-replicates}?
    |   | ...
    |   +--:(nexthop-protection) {nexthop-protection}?
    |   | ...
    |   +--:(nexthop-load-balance) {nexthop-load-balance}?
    |   | ...
  +--rw route-statistic
    | ...
  +--rw route-attributes
    | ...
  +--rw route-vendor-attributes

```

rpcs:

```

+---x rib-add
| +---w input
| | +---w rib-name          string
| | +---w rib-family        rib-family-def
| | +---w ip-rpf-check?     boolean
| +--ro output
|   +--ro result uint32
|   +--ro reason? string
+---x rib-delete
| +---w input
| | +---w rib-name string
| +--ro output
|   +--ro result uint32

```



```

|     +--ro reason? string
+---x route-add
|   +---w input
|   |   +---w return-failure-detail?   boolean
|   |   +---w rib-name                 string
|   |   +---w routes
|   |       +---w route-list* [route-index]
|   |       ...
|   +--ro output
|       +--ro success-count            uint32
|       +--ro failed-count            uint32
|       +--ro failure-detail
|           +--ro failed-routes* [route-index]
|           +--ro route-index uint32
|           +--ro error-code? uint32
+---x route-delete
|   +---w input
|   |   +---w return-failure-detail?   boolean
|   |   +---w rib-name                 string
|   |   +---w routes
|   |       +---w route-list* [route-index]
|   |       ...
|   +--ro output
|       +--ro success-count            uint32
|       +--ro failed-count            uint32
|       +--ro failure-detail
|           +--ro failed-routes* [route-index]
|           +--ro route-index uint32
|           +--ro error-code? uint32
+---x route-update
|   +---w input
|   |   +---w return-failure-detail?   boolean
|   |   +---w rib-name                 string
|   |   +---w (match-options)?
|   |       +--:(match-route-prefix)
|   |       |   ...
|   |       +--:(match-route-attributes)
|   |       |   ...
|   |       +--:(match-route-vendor-attributes) {...}?
|   |       |   ...
|   |       +--:(match-nexthop)
|   |       ...
|   +--ro output
|       +--ro success-count uint32
|       +--ro failed-count uint32
|       +--ro failure-detail
|           +--ro failed-routes* [route-index]
|           +--ro route-index uint32

```



```

|           +--ro error-code? uint32
+---x nh-add
| +---w input
| | +---w rib-name           string
| | +---w nexthop-id?       uint32
| | +---w sharing-flag?     boolean
| | +---w (nexthop-type)?
| |   +--:(nexthop-base)
| |     | ...
| |     +--:(nexthop-chain) {nexthop-chain}?
| |     | ...
| |     +--:(nexthop-replicates) {nexthop-replicates}?
| |     | ...
| |     +--:(nexthop-protection) {nexthop-protection}?
| |     | ...
| |     +--:(nexthop-load-balance) {nexthop-load-balance}?
| |     ...
| +--ro output
|   +--ro result           uint32
|   +--ro reason?         string
|   +--ro nexthop-id?     uint32
+---x nh-delete
  +---w input
  | +---w rib-name           string
  | +---w nexthop-id?       uint32
  | +---w sharing-flag?     boolean
  | +---w (nexthop-type)?
  |   +--:(nexthop-base)
  |     | ...
  |     +--:(nexthop-chain) {nexthop-chain}?
  |     | ...
  |     +--:(nexthop-replicates) {nexthop-replicates}?
  |     | ...
  |     +--:(nexthop-protection) {nexthop-protection}?
  |     | ...
  |     +--:(nexthop-load-balance) {nexthop-load-balance}?
  |     ...
  +--ro output
    +--ro result uint32
    +--ro reason? string
notifications:
+---n nexthop-resolution-status-change
| +--ro nexthop
| | +--ro nexthop-id?       uint32
| | +--ro sharing-flag?     boolean
| | +--ro (nexthop-type)?
| |   +--:(nexthop-base)
| |     | ...

```



```

| |      +--:(nexthop-chain) {nexthop-chain}?
| |      | ...
| |      +--:(nexthop-replicates) {nexthop-replicates}?
| |      | ...
| |      +--:(nexthop-protection) {nexthop-protection}?
| |      | ...
| |      +--:(nexthop-load-balance) {nexthop-load-balance}?
| |      | ...
| |      ...
| +--ro nexthop-state nexthop-state-def
+--n route-change
  +--ro rib-name          string
  +--ro rib-family       rib-family-def
  +--ro route-index      uint64
  +--ro match
    | +--ro (route-type)?
    |   +--:(ipv4)
    |   | ...
    |   +--:(ipv6)
    |   | ...
    |   +--:(mpls-route)
    |   | ...
    |   +--:(mac-route)
    |   | ...
    |   +--:(interface-route)
    |   | ...
    |   ...
  +--ro route-installed-state route-installed-state-def
  +--ro route-state          route-state-def
  +--ro route-change-reason route-reason-def

```

Figure 1: Overview of I2RS Rib Module Structure

2.1. RIB Capability

RIB capability negotiation is very important because not all of the hardware will be able to support all kinds of nexthops and there should be a limitation on how many levels of lookup can be practically performed. Therefore, a RIB data model MUST specify a way for an external entity to learn about the functional capabilities of a network device.

At the same time, nexthop chains can be used to specify multiple headers over a packet, before that particular packet is forwarded. Not every network device will be able to support all kinds of nexthop chains along with the arbitrary number of headers which are chained together. The RIB data model MUST provide a way to expose the nexthop chaining capability supported by a given network device.

This module uses the feature and if-feature statements to achieve above capability negotiation.

2.2. Routing Instance and Rib

A routing instance, in the context of the RIB information model, is a collection of RIBs, interfaces, and routing protocol parameters. A routing instance creates a logical slice of the router and can allow multiple different logical slices; across a set of routers; to communicate with each other. And the routing protocol parameters control the information available in the RIBs. More detail about routing instance can be found in Section 2.2 of [\[I-D.ietf-i2rs-rib-info-model\]](#).

For a routing instance, there will be multiple RIBs. Therefore, this model uses "list" to express the RIBs. The structure tree is shown as following figure.

```
+--rw routing-instance
  +--rw name                string
  +--rw interface-list* [name]
  | +--rw name if:interface-ref
  +--rw router-id?         yang:dotted-quad
  +--rw lookup-limit?     uint8
  +--rw rib-list* [name]
    +--rw name              string
    +--rw rib-family        rib-family-def
    +--rw ip-rpf-check?    boolean
    +--rw route-list* [route-index]
      ... (refer to Section 2.3)
```

Figure 2: Routing Instance Structure

2.3. Route

A route is essentially a match condition and an action following that match. The match condition specifies the kind of route (e.g., IPv4, MPLS, MAC, Interface etc.) and the set of fields to match on.

According to the definition in [\[I-D.ietf-i2rs-rib-info-model\]](#), a route MUST associate with the following attributes:

- o ROUTE_PREFERENCE: See Section 2.3 of [\[I-D.ietf-i2rs-rib-info-model\]](#).
- o ACTIVE: Indicates whether a route is fully resolved and is a candidate for selection.

- o **INSTALLED:** Indicates whether the route got installed in the FIB.

In addition, a route can associate with one or more optional route attributes(e.g., route-vendor-attributes).

For a RIB, there will have a number of routes, so the routes are expressed as a list under a specific rib. Each rib has its own route list.

```

+--rw route-list* [route-index]
  +--rw route-index          uint64
  +--rw match
  | +--rw (route-type)?
  |   +--:(ipv4)
  |     +--rw ipv4
  |       +--rw (ip-route-match-type)?
  |         +--:(dest-ipv4-address)
  |         | ...
  |         +--:(src-ipv4-address)
  |         | ...
  |         +--:(dest-src-ipv4-address)
  |         | ...
  |   +--:(ipv6)
  |     +--rw ipv6
  |       +--rw (ip-route-match-type)?
  |         +--:(dest-ipv6-address)
  |         | ...
  |         +--:(src-ipv6-address)
  |         | ...
  |         +--:(dest-src-ipv6-address)
  |         | ...
  |   +--:(mpls-route)
  |     +--rw mpls-label          uint32
  |   +--:(mac-route)
  |     +--rw mac-address          uint32
  |   +--:(interface-route)
  |     +--rw interface-identifier if:interface-ref
+--rw nexthop
  | ...(refer to Section 2.4)

```

Figure 3: Routes Structure

2.4. Nexthop

A nexthop represents an object resulting from a route lookup. As illustrated in Section 2.4 of [[I-D.ietf-i2rs-rib-info-model](#)], to support various of use cases (e.g., load balance, protection, multicast or the combination of them), the nexthop is modelled as a

multi-level structure and supports recursion. The first level of the next hop includes the following four types:

- o Base: The "base" next hop is the foundation of all other next hop types. It includes the following basic next hops:
 - * next hop-id
 - * IPv4 address
 - * IPv6 address
 - * egress-interface
 - * egress-interface with IPv4 address
 - * egress-interface with IPv6 address
 - * egress-interface with MAC address
 - * logical-tunnel
 - * tunnel-encap
 - * tunnel-decap
 - * rib-name
- o Chain: Provide a way to perform multiple operations on a packet by logically combining them.
- o Load-balance: Designed for load-balance case where it normally will have multiple weighted next hops.
- o Protection: Designed for protection scenario where it normally will have primary and standby next hop.
- o Replicate: Designed for multiple destinations forwarding.

The structure tree of next hop is shown in the following figures.


```

+--rw nexthop
| +--rw nexthop-id?          uint32
| +--rw sharing-flag?       boolean
| +--rw (nexthop-type)?
|   +--:(nexthop-base)
|     | ... (refer to Figure 5)
|     +--:(nexthop-chain) {nexthop-chain}?
|       | +--rw nexthop-chain
|         | +--rw nexthop-list* [nexthop-member-id]
|         |   +--rw nexthop-member-id uint32
|       +--:(nexthop-replicates) {nexthop-replicates}?
|         | +--rw nexthop-replicates
|         |   +--rw nexthop-list* [nexthop-member-id]
|         |     +--rw nexthop-member-id uint32
|       +--:(nexthop-protection) {nexthop-protection}?
|         | +--rw nexthop-protection
|         |   +--rw nexthop-list* [nexthop-member-id]
|         |     +--rw nexthop-member-id uint32
|         |     +--rw nexthop-preference nexthop-preference-def
|       +--:(nexthop-load-balance) {nexthop-load-balance}?
|         +--rw nexthop-lbs
|           +--rw nexthop-list* [nexthop-member-id]
|             +--rw nexthop-member-id uint32
|             +--rw nexthop-lb-weight nexthop-lb-weight-def

```

Figure 4: Nexthop Structure

Figure 5 (as shown below) is a sub-tree of nexthop, it's under the nexthop base node and shows that structure of the "base" nexthop.

```

+--:(nexthop-base)
| +--rw nexthop-base
|   +--rw (nexthop-base-type)?
|     +--:(special-nexthop)
|       | +--rw special? special-nexthop-def
|     +--:(egress-interface-nexthop)
|       | +--rw outgoing-interface if:interface-ref
|     +--:(ipv4-address-nexthop)
|       | +--rw ipv4-address inet:ipv4-address
|     +--:(ipv6-address-nexthop)
|       | +--rw ipv6-address inet:ipv6-address
|     +--:(egress-interface-ipv4-nexthop)
|       | +--rw egress-interface-ipv4-address
|       |   +--rw outgoing-interface if:interface-ref
|       |   +--rw ipv4-address      inet:ipv4-address
|     +--:(egress-interface-ipv6-nexthop)
|       | +--rw egress-interface-ipv6-address
|       |   +--rw outgoing-interface if:interface-ref

```



```

|         |         | +--:(nvgre) {nvgre-tunnel}?
|         |         | | +--rw nvgre-header
|         |         | |   +--rw (nvgre-type)?
|         |         | |   | +--:(ipv4)
|         |         | |   | | +--rw src-ipv4-address inet:ipv4-address
|         |         | |   | | +--rw dest-ipv4-address inet:ipv4-address
|         |         | |   | | +--rw protocol          uint8
|         |         | |   | | +--rw ttl?             uint8
|         |         | |   | | +--rw dscp?           uint8
|         |         | |   | +--:(ipv6)
|         |         | |   |   +--rw src-ipv6-address inet:ipv6-address
|         |         | |   |   +--rw dest-ipv6-address inet:ipv6-address
|         |         | |   |   +--rw next-header      uint8
|         |         | |   |   +--rw traffic-class?   uint8
|         |         | |   |   +--rw flow-label?     uint16
|         |         | |   |   +--rw hop-limit?     uint8
|         |         | |   +--rw virtual-subnet-id  uint32
|         |         | |   +--rw flow-id?          uint16
|         |         | +--:(vxlan) {vxlan-tunnel}?
|         |         | | +--rw vxlan-header
|         |         | | | +--rw (vxlan-type)?
|         |         | | | | +--:(ipv4)
|         |         | | | | | +--rw src-ipv4-address inet:ipv4-address
|         |         | | | | | +--rw dest-ipv4-address inet:ipv4-address
|         |         | | | | | +--rw protocol          uint8
|         |         | | | | | +--rw ttl?             uint8
|         |         | | | | | +--rw dscp?           uint8
|         |         | | | | +--:(ipv6)
|         |         | | | |   +--rw src-ipv6-address inet:ipv6-address
|         |         | | | |   +--rw dest-ipv6-address inet:ipv6-address
|         |         | | | |   +--rw next-header      uint8
|         |         | | | |   +--rw traffic-class?   uint8
|         |         | | | |   +--rw flow-label?     uint16
|         |         | | | |   +--rw hop-limit?     uint8
|         |         | | | +--rw vxlan-identifier   uint32
|         |         | +--:(tunnel-decap-nexthop) {nexthop-tunnel}?
|         |         | | +--rw tunnel-decap
|         |         | | | +--rw (tunnel-type)?
|         |         | | | | +--:(ipv4) {ipv4-tunnel}?
|         |         | | | | | +--rw ipv4-decap
|         |         | | | | |   +--rw ipv4-decap tunnel-decap-action-def
|         |         | | | | |   +--rw ttl-action?  ttl-action-def
|         |         | | | | +--:(ipv6) {ipv6-tunnel}?
|         |         | | | | | +--rw ipv6-decap
|         |         | | | | |   +--rw ipv6-decap tunnel-decap-action-def
|         |         | | | | |   +--rw hop-limit-action? hop-limit-action-def
|         |         | | +--:(mpls) {mpls-tunnel}?
|         |         | | | +--rw label-pop

```



```

|         |         +--rw label-pop      mpls-label-action-def
|         |         +--rw ttl-action?   ttl-action-def
|         +--:(logical-tunnel-nexthop) {nexthop-tunnel}?
|         |   +--rw logical-tunnel
|         |     +--rw tunnel-type tunnel-type-def
|         |     +--rw tunnel-name string
|         +--:(rib-name-nexthop)
|         |   +--rw rib-name?           string
|         +--:(nexthop-identifier)
|         |   +--rw nexthop-ref         nexthop-ref

```

Figure 5: Nexthop Base Structure

2.5. RPC Operations

This module defines the following RPC operations:

- o `rib-add`: It is defined to add a rib to a routing instance. A name of the rib, address family of the rib and whether the RPF check is enabled are passed as the input parameters. The output is the result of the add operation:
 - * `true` - success;
 - * `false` - failed; when failed, the `i2rs` agent may return the specific reason that causes the failure.
- o `rib-delete`: It is defined to delete a rib from a routing instance. When a rib is deleted, all routes installed in the rib will be deleted. A name of the rib is passed as the input parameter. The output is the result of the delete operation:
 - * `true` - success;
 - * `false` - failed; when failed, the `i2rs` agent may return the specific reason that causes the failure.
- o `route-add`: It is defined to add a route or a set of routes to a rib. A rib name, the route prefix(es), route attributes, route vendor attributes, nexthop and whether return failure detail are passed as the input parameters. Before calling the `route-add` rpc, it is required to call the `nh-add` rpc to create and/or return the nexthop identifier. The output is a combination of the route operation states that include:
 - * `success-count`: the numbers of routes that are successfully added;

- * failed-count: the numbers of the routes that are failed to be added;
 - * failure-detail: shows the specific failed routes that failure reason.
- o route-delete: It is defined to delete a route or a set of routes from a rib. A name of the rib, the route prefix(es) and whether return failure detail are passed as the input parameters. The output is combination of the route operation states that include:
- * success-count: the numbers of routes that are successfully deleted;
 - * failed-count: the numbers of the routes that are failed to be deleted;
 - * failure-detail: shows the specific failed routes that failure reason.
- o route-update: It is defined to update a route or a set of routes. A rib name, the route prefix(es), or route attributes, or route vendor attributes, or nexthop are passed as the input parameters. The match conditions can be either route prefix(es), or route attributes, or route vendor attributes, or nexthop. The update actions include: update the nexthop, update the route attributes, update the route vendor attributes. The output is combination of the route operation states that include:
- * success-count: the numbers of routes that are successfully updated;
 - * failed-count: the numbers of the routes that are failed to be updated;
 - * failure-detail: shows the specific failed routes that failure reason.
- o nh-add: It is defined to add a nexthop to a rib. A name of the rib and a nexthop are passed as the input parameters. The network node is required to allocate a nexthop identifier to the nexthop. The outputs include the result of the nexthop add operation.
- * true - success; when success, a nexthop identifier will be returned to the i2rs client.
 - * false - failed; when failed, the i2rs agent may return the specific reason that causes the failure.

- o nh-delete: It is defined to delete a nexthop from a rib. A name of a rib and a nexthop or nexthop identifier are passed as the input parameters. The output is the result of the delete operation:

- * true - success;
- * false - failed; when failed, the i2rs agent may return the specific reason that causes the failure.

The structure tree of rpcs is showing in following figure.

rpcs:

```

+---x rib-add
| +---w input
| | +---w rib-name      string
| | +---w rib-family    rib-family-def
| | +---w ip-rpf-check? boolean
| +--ro output
|   +--ro result uint32
|   +--ro reason? string
+---x rib-delete
| +---w input
| | +---w rib-name string
| +--ro output
|   +--ro result uint32
|   +--ro reason? string
+---x route-add
| +---w input
| | +---w return-failure-detail?  boolean
| | +---w rib-name                string
| | +---w routes
| |   +---w route-list* [route-index]
| |   ...
| +--ro output
|   +--ro success-count      uint32
|   +--ro failed-count      uint32
|   +--ro failure-detail
|   +--ro failed-routes* [route-index]
|   +--ro route-index uint32
|   +--ro error-code? uint32
+---x route-delete
| +---w input
| | +---w return-failure-detail?  boolean
| | +---w rib-name                string
| | +---w routes
| |   +---w route-list* [route-index]
| |   ...

```



```

|   +--ro output
|     +--ro success-count    uint32
|     +--ro failed-count     uint32
|     +--ro failure-detail
|       +--ro failed-routes* [route-index]
|         +--ro route-index uint32
|         +--ro error-code? uint32
+---x route-update
|   +---w input
|     | +---w return-failure-detail?    boolean
|     | +---w rib-name                  string
|     | +---w (match-options)?
|     |   +--:(match-route-prefix)
|     |   | ...
|     |   +--:(match-route-attributes)
|     |   | ...
|     |   +--:(match-route-vendor-attributes) {...}?
|     |   | ...
|     |   +--:(match-nexthop)
|     |   | ...
|     +--ro output
|       +--ro success-count uint32
|       +--ro failed-count uint32
|       +--ro failure-detail
|         +--ro failed-routes* [route-index]
|           +--ro route-index uint32
|           +--ro error-code? uint32
+---x nh-add
|   +---w input
|     | +---w rib-name                string
|     | +---w nexthop-id?             uint32
|     | +---w sharing-flag?           boolean
|     | +---w (nexthop-type)?
|     |   ...
|     +--ro output
|       +--ro result                  uint32
|       +--ro reason?                 string
|       +--ro nexthop-id?             uint32
+---x nh-delete
|   +---w input
|     | +---w rib-name                string
|     | +---w nexthop-id?             uint32
|     | +---w sharing-flag?           boolean
|     | +---w (nexthop-type)?
|     |   ...
|     +--ro output
|       +--ro result uint32
|       +--ro reason? string

```


Figure 6: RPCs Structure

2.6. Notifications

Asynchronous notifications are sent by the RIB manager of a network device to an external entity when some event triggers on the network device. A RIB data-model MUST support sending 2 kind of asynchronous notifications.

1. Route change notification:

- o Installed (Indicates whether the route got installed in the FIB) ;
- o Active (Indicates whether a route is fully resolved and is a candidate for selection) ;
- o Reason - E.g. Not authorized

2. Nexthop resolution status notification

Nexthops can be fully resolved nexthops or an unresolved nexthop.

A resolved nexthop has adequate level of information to send the outgoing packet towards the destination by forwarding it on an interface of a directly connected neighbor.

An unresolved nexthop is something that requires the RIB manager to determine the final resolved nexthop. For example, in a case when a nexthop could be an IP address. The RIB manager would resolve how to reach that IP address, e.g. by checking if that particular IP is address reachable by regular IP forwarding or by a MPLS tunnel or by both. If the RIB manager cannot resolve the nexthop, then the nexthop remains in an unresolved state and is NOT a suitable candidate for installation in the FIB.

The structure tree of notifications is shown in the following figure.

notifications:

```

+---n nexthop-resolution-status-change
| +--ro nexthop
| | +--ro nexthop-id          uint32
| | +--ro sharing-flag       boolean
| | +--ro (nexthop-type)?
| |   +--:(nexthop-base)
| |     | ...
| |     +--:(nexthop-chain) {nexthop-chain}?
| |     | ...
| |     +--:(nexthop-replicates) {nexthop-replicates}?
| |     | ...
| |     +--:(nexthop-protection) {nexthop-protection}?
| |     | ...
| |     +--:(nexthop-load-balance) {nexthop-load-balance}?
| |     ...
| +--ro nexthop-state nexthop-state-def
+---n route-change
+--ro rib-name          string
+--ro rib-family        rib-family-def
+--ro route-index      uint64
+--ro match
| +--ro (route-type)?
|   +--:(ipv4)
|   | ...
|   +--:(ipv6)
|   | ...
|   +--:(mpls-route)
|   | ...
|   +--:(mac-route)
|   | ...
|   +--:(interface-route)
|   ...
+--ro route-installed-state route-installed-state-def
+--ro route-state          route-state-def
+--ro route-change-reason  route-reason-def

```

Figure 7: Notifications Structure

3. YANG Modules

```

<CODE BEGINS> file "ietf-i2rs-rib@2015-11-20.yang"

module ietf-i2rs-rib {
  namespace "urn:ietf:params:xml:ns:yang:ietf-i2rs-rib";
  // replace with iana namespace when assigned
  prefix "iir";

```



```
import ietf-inet-types {
  prefix inet;
  //rfc6991
}

import ietf-interfaces {
  prefix "if";
}

import ietf-yang-types {
  prefix yang;
}

organization
  "IETF I2RS (Interface to Routing System) Working Group";
contact
  "WG Web:   <http://tools.ietf.org/wg/i2rs/>
  WG List:  <mailto:i2rs@ietf.org>

  WG Chair: Susan Hares
            <mailto:shares@ndzh.com>

  WG Chair: Jeffrey Haas
            <mailto:jhaas@pfrfc.org>

  Editor:   Lixing Wang
            <mailto:wang_little_star@sina.com>

  Editor:   Hariharan Ananthkrishnan
            <mailto:hari@packetdesign.com>

  Editor:   Mach(Guoyi) Chen
            <mailto:mach.chen@huawei.com>

  Editor:   Amit Dass
            <mailto:amit.dass@ericsson.com>

  Editor:   Sriganesh Kini
            <mailto:sriganesh.kini@ericsson.com>

  Editor:   Nitin Bahadur
            <mailto:nitin_bahadur@yahoo.com>";
description
  "This module defines a YANG data model for
  Routing Information Base (RIB) that aligns
  with the I2RS RIB information model.";
revision "2015-11-20" {
  description "initial revision";
```



```
    reference "draft-ietf-i2rs-data-model-04";
  }

//Features
feature nexthop-tunnel {
  description
    "This feature means that a node support
    tunnel nexthop capability.";
}

feature nexthop-chain {
  description
    "This feature means that a node support
    chain nexthop capability.";
}

feature nexthop-protection {
  description
    "This feature means that a node support
    protection nexthop capability.";
}

feature nexthop-replicates {
  description
    "This feature means that a node support
    replicates nexthop capability.";
}

feature nexthop-load-balance {
  description
    "This feature means that a node support
    load balance nexthop capability.";
}

feature ipv4-tunnel {
  description
    "This feature means that a node support
    IPv4 tunnel encapsulation capability.";
}

feature ipv6-tunnel {
  description
    "This feature means that a node support
    IPv6 tunnel encapsulation capability.";
}

feature mpls-tunnel {
  description
```



```
        "This feature means that a node support
        MPLS tunnel encapsulation capability.";
    }

feature vxlan-tunnel {
    description
        "This feature means that a node support
        VxLAN tunnel encapsulation capability.";
}

feature gre-tunnel {
    description
        "This feature means that a node support
        GRE tunnel encapsulation capability.";
}

feature nvgre-tunnel {
    description
        "This feature means that a node support
        NVGRE tunnel encapsulation capability.";
}

feature route-vendor-attributes {
    description
        "This feature means that a node support
        route vendor attributes.";
}

//Identities and Type Definitions
identity mpls-label-action {
    description
        "Base identify from which all mpls label
        operations are derived.
        The MPLS label stack operations include:
        push - to add a new label to a label stack,
        pop - to pop the top label from a label stack,
        swap - to change the top label of a label
        stack with new label.";
}

identity label-push {
    base "mpls-label-action";
    description
        "MPLS label stack operation: push.";
}

identity label-pop {
    base "mpls-label-action";
```



```
    description
      "MPLS label stack operation: pop.";
  }

  identity label-swap {
    base "mpls-label-action";
    description
      "MPLS label stack operation: swap.";
  }

  typedef mpls-label-action-def {
    type identityref {
      base "mpls-label-action";
    }
    description
      "MPLS label action def.";
  }

  identity tunnel-decap-action {
    description
      "Base identify from which all tunnel decap
      actions are derived.
      Tunnel decap actions include:
      ipv4-decap - to decap an IPv4 tunnel,
      ipv6-decap - to decap an IPv6 tunnel.";
  }

  identity ipv4-decap {
    base "tunnel-decap-action";
    description
      "IPv4 tunnel decap.";
  }

  identity ipv6-decap {
    base "tunnel-decap-action";
    description
      "IPv4 tunnel decap.";
  }

  typedef tunnel-decap-action-def {
    type identityref {
      base "tunnel-decap-action";
    }
    description
      "Tunnel decap def.";
  }

  identity ttl-action {
```



```
    description
      "Base identify from which all TTL
      actions are derived.";
  }

identity no-action {
  base "ttl-action";
  description
    "Do nothing regarding the TTL.";
}

identity copy-to-inner {
  base "ttl-action";
  description
    "Copy the TTL of the outer header
    to inner header.";
}

identity decrease-and-copy-to-inner {
  base "ttl-action";
  description
    "Decrease TTL by one and copy the TTL
    to inner header.";
}

identity decrease-and-copy-to-next {
  base "ttl-action";
  description
    "Decrease TTL by one and copy the TTL
    to the next header. For example: when
    MPLS label swapping, decrease the TTL
    of the in label and copy it to the out
    label.";
}

typedef ttl-action-def {
  type identityref {
    base "ttl-action";
  }
  description
    "TTL action def.";
}

identity hop-limit-action {
  description
    "Base identify from which all hop limit
    actions are derived.";
}
```



```
identity hop-limit-no-action {
  base "hop-limit-action";
  description
    "Do nothing regarding the hop limit.";
}

identity hop-limit-copy-to-inner {
  base "hop-limit-action";
  description
    "Copy the hop limit of the outer header
    to inner header.";
}

typedef hop-limit-action-def {
  type identityref {
    base "hop-limit-action";
  }
  description
    "IPv6 hop limit action def.";
}

identity special-nextHop {
  description
    "Base identify from which all special
    nextHops are derived.";
}

identity discard {
  base "special-nextHop";
  description
    "This indicates that the network
    device should drop the packet and
    increment a drop counter.";
}

identity discard-with-error {
  base "special-nextHop";
  description
    "This indicates that the network
    device should drop the packet,
    increment a drop counter and send
    back an appropriate error message
    (like ICMP error).";
}

identity receive {
  base "special-nextHop";
  description
```



```
    "This indicates that that the traffic is
    destined for the network device. For
    example, protocol packets or OAM packets.
    All locally destined traffic SHOULD be
    throttled to avoid a denial of service
    attack on the router's control plane. An
    optional rate-limiter can be specified
    to indicate how to throttle traffic
    destined for the control plane.";
}

identity cos-value {
    base "special-nexthop";
    description
        "Cos-value special nexthop.";
}

typedef special-nexthop-def {
    type identityref {
        base "special-nexthop";
    }
    description
        "Special nexthop def.";
}

identity ip-route-match-type {
    description
        "Base identify from which all route
        match types are derived.
        Route match type could be:
        match source, or
        match destination, or
        match source and destination.";
}

identity match-ip-src {
    base "ip-route-match-type";
    description
        "Source route match type.";
}

identity match-ip-dest {
    base "ip-route-match-type";
    description
        "Destination route match type";
}

identity match-ip-src-dest {
    base "ip-route-match-type";
    description
```



```
    "Src and Dest route match type";
}

typedef ip-route-match-type-def {
    type identityref {
        base "ip-route-match-type";
    }
    description
        "IP route match type def.";
}

identity rib-family {
    description
        "Base identify from which all rib
        address families are derived.";
}

identity ipv4-rib-family {
    base "rib-family";
    description
        "IPv4 rib address family.";
}

identity ipv6-rib-family {
    base "rib-family";
    description
        "IPv6 rib address family.";
}

identity mpls-rib-family {
    base "rib-family";
    description
        "MPLS rib address family.";
}

identity ieee-mac-rib-family {
    base "rib-family";
    description
        "MAC rib address family.";
}

typedef rib-family-def {
    type identityref {
        base "rib-family";
    }
    description
        "Rib address family def.";
}
```



```
identity route-type {
  description
    "Base identify from which all route types
    are derived.";
}

identity ipv4-route {
  base "route-type";
  description
    "IPv4 route type.";
}

identity ipv6-route {
  base "route-type";
  description
    "IPv6 route type.";
}

identity mpls-route {
  base "route-type";
  description
    "MPLS route type.";
}

identity ieee-mac {
  base "route-type";
  description
    "MAC route type.";
}

identity interface {
  base "route-type";
  description
    "Interface route type.";
}

typedef route-type-def {
  type identityref {
    base "route-type";
  }
  description
    "Route type def.";
}

identity tunnel-type {
  description
    "Base identify from which all tunnel
    types are derived.";
```



```
}  
  
identity ipv4-tunnel {  
    base "tunnel-type";  
    description  
        "IPv4 tunnel type";  
}  
  
identity ipv6-tunnel {  
    base "tunnel-type";  
    description  
        "IPv6 Tunnel type";  
}  
  
identity mpls-tunnel {  
    base "tunnel-type";  
    description  
        "MPLS tunnel type";  
}  
  
identity gre-tunnel {  
    base "tunnel-type";  
    description  
        "GRE tunnel type";  
}  
  
identity vxlan-tunnel {  
    base "tunnel-type";  
    description  
        "VxLAN tunnel type";  
}  
  
identity nvgre-tunnel {  
    base "tunnel-type";  
    description  
        "NVGRE tunnel type";  
}  
  
typedef tunnel-type-def {  
    type identityref {  
        base "tunnel-type";  
    }  
    description  
        "Tunnel type def.";  
}  
  
identity route-state {  
    description
```



```
    "Base identify from which all route
      states are derived.";
  }

  identity active {
    base "route-state";
    description
      "Active state.";
  }

  identity inactive {
    base "route-state";
    description
      "Inactive state.";
  }

  typedef route-state-def {
    type identityref {
      base "route-state";
    }
    description
      "Route state def.";
  }

  identity nexthop-state {
    description
      "Base identify from which all nexthop
        states are derived.";
  }

  identity resolved {
    base "nexthop-state";
    description
      "Reolved nexthop state.";
  }

  identity unresolved {
    base "nexthop-state";
    description
      "Unresolved nexthop state.";
  }

  typedef nexthop-state-def {
    type identityref {
      base "nexthop-state";
    }
    description
      "Nexthop state def.";
```



```
}

identity route-installed-state {
  description
    "Base identify from which all route
    installed states are derived.";
}

identity uninstalled {
  base "route-installed-state";
  description
    "Uninstalled state.";
}

identity installed {
  base "route-installed-state";
  description
    "Installed state.";
}

typedef route-installed-state-def {
  type identityref {
    base "route-installed-state";
  }
  description
    "Route installed state def.";
}

identity route-reason {
  description
    "Base identify from which all route
    reasons are derived.";
}

identity low-preference {
  base "route-reason";
  description
    "Low preference";
}

identity unresolved-nextthop {
  base "route-reason";
  description
    "Unresolved nextthop";
}

identity higher-metric {
  base "route-reason";
```



```
    description
      "Higher metric";
  }

typedef route-reason-def {
  type identityref {
    base "route-reason";
  }
  description
    "Route reason def.";
}

typedef nexthop-preference-def {
  type uint8 {
    range "1..99";
  }
  description
    "Nexthop-preference is used for protection schemes.
    It is an integer value between 1 and 99. A lower
    value indicates higher preference. To download N
    nexthops to the FIB, the N nexthops with the lowest
    value are selected.";
}

typedef nexthop-lb-weight-def {
  type uint8 {
    range "1..99";
  }
  description
    "Nhop-lb-weight is a number between 1 and 99.";
}

typedef nexthop-ref {
  type leafref {
    path "/iir:routing-instance" +
        "/iir:rib-list" +
        "/iir:route-list" +
        "/iir:nexthop" +
        "/iir:nexthop-id";
  }
  description
    "A nexthop reference that provides
    an indirection reference to a nexthop.";
}

//Groupings
grouping route-prefix {
  description
```



```
"The common attributes used for all types of route prefix.";
leaf route-index {
  type uint64 ;
  mandatory true;
  description
    "Route index.";
}
container match {
  description
    "The match condition specifies the
    kind of route (IPv4, MPLS, etc.)
    and the set of fields to match on.";
  choice route-type {
    description
      "Route types: IPv4, IPv6, MPLS, MAC etc.";
    case ipv4 {
      description
        "IPv4 route case.";
      container ipv4 {
        description
          "IPv4 route match.";
        choice ip-route-match-type {
          description
            "IP route match type options:
            match source, or
            match destination, or
            match source and destination.";
          case dest-ipv4-address {
            leaf dest-ipv4-prefix {
              type inet:ipv4-prefix;
              mandatory true;
              description
                "An IPv4 destination address as the match.";
            }
          }
          case src-ipv4-address {
            leaf src-ipv4-prefix {
              type inet:ipv4-prefix;
              mandatory true;
              description
                "An IPv4 source address as the match.";
            }
          }
          case dest-src-ipv4-address {
            container dest-src-ipv4-address {
              description
                "A combination of an IPv4 source and
                an IPv4 destination address as the match.";
            }
          }
        }
      }
    }
  }
}
```



```

        "A combination of an IPv6 source and
        an IPv6 destination address as the match.";
leaf dest-ipv6-prefix {
  type inet:ipv6-prefix;
  mandatory true;
  description
    "The IPv6 destination address of the match";
}
leaf src-ipv6-prefix {
  type inet:ipv6-prefix;
  mandatory true;
  description
    "The IPv6 source address of the match.";
}
}
}
}
}
}
}
case mpls-route {
  description
    "MPLS route case.";
  leaf mpls-label {
    type uint32 ;
    mandatory true;
    description
      "The label used for matching.";
  }
}
case mac-route {
  description
    "MAC route case.";
  leaf mac-address {
    type uint32 ;
    mandatory true;
    description
      "The MAC address used for matching.";
  }
}
case interface-route {
  description
    "Interface route case.";
  leaf interface-identifier {
    type if:interface-ref;
    mandatory true;
    description
      "The interface used for matching.";
  }
}

```



```
    }
  }
}

grouping route {
  description
    "The common attributes used for all types of route.";
  uses route-prefix;
  container nexthop {
    description
      "The nexthop of the route.";
    uses nexthop;
  }
  container route-statistic {
    description
      "The statistic information of the route.";
    leaf route-state {
      type route-state-def;
      config false;
      description
        "Indicate a route's state: Active or Inactive.";
    }
    leaf route-installed-state {
      type route-installed-state-def;
      config false;
      description
        "Indicate that a route's installed states:
        Installed or uninstalled.";
    }
    leaf route-reason {
      type route-reason-def;
      config false;
      description
        "Indicate the route reason.";
    }
  }
  container route-attributes {
    description
      "Route attributes.";
    uses route-attributes;
  }
  container route-vendor-attributes {
    description
      "Route vendor attributes.";
    uses route-vendor-attributes;
  }
}
```



```
grouping nexthop-list {
  description
    "A generic nexthop list.";
  list nexthop-list {
    key "nexthop-member-id";
    description
      "A list of nexthop.";
    leaf nexthop-member-id {
      type uint32;
      mandatory true;
      description
        "A nexthop identifier that points
         to a nexthop list member.
         A nexthop list member is a nexthop.";
    }
  }
}

grouping nexthop-list-p {
  description
    "A nexthop list with preference parameter.";
  list nexthop-list {
    key "nexthop-member-id";
    description
      "A list of nexthop.";
    leaf nexthop-member-id {
      type uint32;
      mandatory true;
      description
        "A nexthop identifier that points
         to a nexthop list member.
         A nexthop list member is a nexthop.";
    }
    leaf nexthop-preference {
      type nexthop-preference-def;
      mandatory true;
      description
        "Nexthop-preference is used for protection schemes.
         It is an integer value between 1 and 99. A lower
         value indicates higher preference. To download a
         primary/standby/tertiary group to the FIB, the
         nexthops that are resolved and have two highest
         preferences are selected.";
    }
  }
}

grouping nexthop-list-w {
```



```
description
  "A nexthop list with weight parameter.";
list nexthop-list {
  key "nexthop-member-id";
  description
    "A list of nexthop.";
  leaf nexthop-member-id {
    type uint32;
    mandatory true;
    description
      "A nexthop identifier that points
      to a nexthop list member.
      A nexthop list member is a nexthop.";
  }
  leaf nexthop-lb-weight {
    type nexthop-lb-weight-def;
    mandatory true;
    description
      "The weight of a nexthop of
      the load balance nexthops.";
  }
}
}

grouping nexthop {
  description
    "The nexthop structure.";
  leaf nexthop-id {
    type uint32;
    description
      "An identifier that refers to a nexthop.";
  }
  leaf sharing-flag {
    type boolean;
    description
      "To indicate whether a nexthop is sharable
      or non-sharable.
      true - sharable, means the nexthop can be shared
      with other routes
      false - non-sharable, means the nexthop can not
      be shared with other routes.";
  }
}
choice nexthop-type {
  description
    "Nexthop type options.";
  case nexthop-base {
    container nexthop-base {
      description
```



```
        "The base nexthop.";
        uses nexthop-base;
    }
}
case nexthop-chain {
    if-feature nexthop-chain;
    container nexthop-chain {
        description
            "A chain nexthop.";
        uses nexthop-list;
    }
}
case nexthop-replicates {
    if-feature nexthop-replicates;
    container nexthop-replicates {
        description
            "A replicates nexthop.";
        uses nexthop-list;
    }
}
case nexthop-protection {
    if-feature nexthop-protection;
    container nexthop-protection {
        description
            "A protection nexthop.";
        uses nexthop-list-p;
    }
}
case nexthop-load-balance {
    if-feature nexthop-load-balance;
    container nexthop-lbs {
        description
            "A load balance nexthop.";
        uses nexthop-list-w;
    }
}
}
}

grouping nexthop-base {
    description
        "The base nexthop.";
    choice nexthop-base-type {
        description
            "Nexthop base type options.";
        case special-nexthop {
            leaf special {
                type special-nexthop-def;
            }
        }
    }
}
```



```
        description
            "A special nexthop.";
    }
}
case egress-interface-nexthop {
    leaf outgoing-interface {
        type if:interface-ref;
        mandatory true;
        description
            "The nexthop is an outgoing interface.";
    }
}
case ipv4-address-nexthop {
    leaf ipv4-address {
        type inet:ipv4-address;
        mandatory true;
        description
            "The nexthop is an IPv4 address.";
    }
}
case ipv6-address-nexthop {
    leaf ipv6-address {
        type inet:ipv6-address;
        mandatory true;
        description
            "The nexthop is an IPv6 address.";
    }
}
case egress-interface-ipv4-nexthop {
    container egress-interface-ipv4-address{
        leaf outgoing-interface {
            type if:interface-ref;
            mandatory true;
            description
                "Name of the outgoing interface.";
        }
        leaf ipv4-address {
            type inet:ipv4-address;
            mandatory true;
            description
                "The nexthop points to an interface with
                an IPv4 address.";
        }
        description
            "The nexthop is an Egress-interface and an ip
            address.This can be used in cases e.g.where
            the ip address is a link-local address.";
    }
}
```



```
}
case egress-interface-ipv6-nexthop {
  container egress-interface-ipv6-address {
    leaf outgoing-interface {
      type if:interface-ref;
      mandatory true;
      description
        "Name of the outgoing interface.";
    }
    leaf ipv6-address {
      type inet:ipv6-address;
      mandatory true;
      description
        "The nexthop points to an interface with
        an IPv6 address.";
    }
    description
      "The nexthop is an Egress-interface and an ip
      address.This can be used in cases e.g.where
      the ip address is a link-local address.";
  }
}
case egress-interface-mac-nexthop {
  container egress-interface-mac-address {
    leaf outgoing-interface {
      type if:interface-ref;
      mandatory true;
      description
        "Name of the outgoing interface.";
    }
    leaf ieee-mac-address {
      type uint32;
      mandatory true;
      description
        "The nexthop points to an interface with
        a specific mac-address.";
    }
    description
      "The egress interface must be an ethernet
      interface. Address resolution is not required
      for this nexthop.";
  }
}
case tunnel-encap-nexthop {
  if-feature nexthop-tunnel;
  container tunnel-encap {
    uses tunnel-encap;
    description
```



```
        "This can be an encap representing an IP tunnel or
        MPLS tunnel or others as defined in info model.
        An optional egress interface can be chained to the
        tunnel encap to indicate which interface to send
        the packet out on. The egress interface is useful
        when the network device contains Ethernet interfaces
        and one needs to perform address resolution for the
        IP packet.";
    }
}
case tunnel-decap-nextthop {
    if-feature nextthop-tunnel;
    container tunnel-decap {
        uses tunnel-decap;
        description
            "This is to specify decapsulating a tunnel header.";
    }
}
case logical-tunnel-nextthop {
    if-feature nextthop-tunnel;
    container logical-tunnel {
        uses logical-tunnel;
        description
            "This can be a MPLS LSP or a GRE tunnel (or others
            as defined in This document), that is represented
            by a unique identifier (e.g. name).";
    }
}
case rib-name-nextthop {
    leaf rib-name {
        type string;
        description
            "A nextthop pointing to a rib indicates that the
            route lookup needs to continue in The specified
            rib. This is a way to perform chained lookups.";
    }
}
case nextthop-identifier {
    leaf nextthop-ref {
        type nextthop-ref;
        mandatory true;
        description
            "A nextthop reference that points to a nextthop.";
    }
}
}
```



```
grouping route-vendor-attributes {
  description
    "Route vendor attributes.";
}

grouping logical-tunnel {
  description
    "A logical tunnel that is identified
    by a type and a tunnel name.";
  leaf tunnel-type {
    type tunnel-type-def;
    mandatory true;
    description
      "A tunnel type.";
  }
  leaf tunnel-name {
    type string;
    mandatory true;
    description
      "A tunnel name that points to a logical tunnel.";
  }
}

grouping ipv4-header {
  description
    "The IPv4 header encapsulation information.";
  leaf src-ipv4-address {
    type inet:ipv4-address;
    mandatory true;
    description
      "The source ip address of the header.";
  }
  leaf dest-ipv4-address {
    type inet:ipv4-address;
    mandatory true;
    description
      "The destination ip address of the header.";
  }
  leaf protocol {
    type uint8;
    mandatory true;
    description
      "The protocol id of the header.";
  }
  leaf ttl {
    type uint8;
    description
      "The TTL of the header.";
  }
}
```



```
    }
    leaf dscp {
      type uint8;
      description
        "The DSCP field of the header.";
    }
  }
}

grouping ipv6-header {
  description
    "The IPv6 header encapsulation information.";
  leaf src-ipv6-address {
    type inet:ipv6-address;
    mandatory true;
    description
      "The source ip address of the header.";
  }
  leaf dest-ipv6-address {
    type inet:ipv6-address;
    mandatory true;
    description
      "The destination ip address of the header.";
  }
  leaf next-header {
    type uint8;
    mandatory true;
    description
      "The next header of the IPv6 header.";
  }
  leaf traffic-class {
    type uint8;
    description
      "The traffic class value of the header.";
  }
  leaf flow-label {
    type uint16;
    description
      "The flow label of the header.";
  }
  leaf hop-limit {
    type uint8;
    description
      "The hop limit the header.";
  }
}

grouping nvgre-header {
  description
```



```
    "The NVGRE header encapsulation information.";
  choice nvgre-type {
    description
      "NVGRE can use either IPv4
       or IPv6 header for encapsulation.";
    case ipv4 {
      uses ipv4-header;
    }
    case ipv6 {
      uses ipv6-header;
    }
  }
  leaf virtual-subnet-id {
    type uint32;
    mandatory true;
    description
      "The subnet identifier of the NVGRE header.";
  }
  leaf flow-id {
    type uint16;
    description
      "The flow identifier of the NVGRE header.";
  }
}

grouping vxlan-header {
  description
    "The VxLAN encapsulation header information.";
  choice vxlan-type {
    description
      "NVGRE can use either IPv4
       or IPv6 header for encapsulation.";
    case ipv4 {
      uses ipv4-header;
    }
    case ipv6 {
      uses ipv6-header;
    }
  }
  leaf vxlan-identifier {
    type uint32;
    mandatory true;
    description
      "The VxLAN identifier of the VxLAN header.";
  }
}

grouping gre-header {
```



```
description
  "The GRE encapsulation header information.";
choice dest-address-type {
  description
    "GRE options: IPv4 and IPv6";
  case ipv4 {
    leaf ipv4-dest {
      type inet:ipv4-address;
      mandatory true;
      description
        "The destination ip address of the GRE header.";
    }
  }
  case ipv6 {
    leaf ipv6-dest {
      type inet:ipv6-address;
      mandatory true;
      description
        "The destination ip address of the GRE header.";
    }
  }
}
leaf protocol-type {
  type uint16;
  mandatory true;
  description
    "The protocol type of the GRE header.";
}
leaf key {
  type uint64;
  description
    "The GRE key of the GRE header.";
}
}

grouping mpls-header {
  description
    "The MPLS encapsulation header information.";
  list label-operations {
    key "label-oper-id";
    description
      "Label operations.";
    leaf label-oper-id {
      type uint32;
      description
        "An optional identifier that points
        to a label operation.";
    }
  }
}
```



```
choice label-actions {
  description
    "Label action options.";
  case label-push {
    container label-push {
      description
        "Label push operation.";
      leaf label {
        type uint32;
        mandatory true;
        description
          "The label to be pushed.";
      }
      leaf s-bit {
        type boolean;
        description
          "The s-bit of the label to be pushed. ";
      }
      leaf tc-value {
        type uint8;
        description
          "The traffic class value of the label to be pushed.";
      }
      leaf ttl-value {
        type uint8;
        description
          "The TTL value of the label to to be pushed.";
      }
    }
  }
}
case label-swap {
  container label-swap {
    description
      "Label swap operation.";
    leaf in-label {
      type uint32;
      mandatory true;
      description
        "The label to be swapped.";
    }
    leaf out-label {
      type uint32;
      mandatory true;
      description
        "The out MPLS label.";
    }
    leaf ttl-action {
      type ttl-action-def;
    }
  }
}
```



```
        description
        "The label ttl actions:
        - No-action, or
        - Copy to inner label,or
        - Decrease (the in label) by 1 and
          copy to the out label.";
    }
}
}
}
}
```

```
grouping tunnel-encap{
  description
  "Tunnel encapsulation inforamtion.";
  choice tunnel-type {
    description
    "Tunnel options for next-hops.";
    case ipv4 {
      if-feature ipv4-tunnel;
      container ipv4-header {
        uses ipv4-header;
        description
        "IPv4 header.";
      }
    }
    case ipv6 {
      if-feature ipv6-tunnel;
      container ipv6-header {
        uses ipv6-header;
        description
        "IPv6 header.";
      }
    }
    case mpls {
      if-feature mpls-tunnel;
      container mpls-header {
        uses mpls-header;
        description
        "MPLS header.";
      }
    }
    case gre {
      if-feature gre-tunnel;
      container gre-header {
        uses gre-header;
        description
```



```
        "GRE header.";
    }
}
case nvgre {
    if-feature nvgre-tunnel;
    container nvgre-header {
        uses nvgre-header;
        description
            "NvGRE header.";
    }
}
case vxlan {
    if-feature vxlan-tunnel;
    container vxlan-header {
        uses vxlan-header;
        description
            "VxLAN header.";
    }
}
}
}

grouping tunnel-decap {
    description
        "Tunnel decapsulation inforamtion.";
    choice tunnel-type {
        description
            "Nexthop tunnel type options.";
        case ipv4 {
            if-feature ipv4-tunnel;
            container ipv4-decap {
                description
                    "IPv4 decap.";
                leaf ipv4-decap {
                    type tunnel-decap-action-def;
                    mandatory true;
                    description
                        "IPv4 decap operations.";
                }
                leaf ttl-action {
                    type ttl-action-def;
                    description
                        "The ttl actions:
                        no-action or copy to inner header.";
                }
            }
        }
    }
}
case ipv6 {
```



```
    if-feature ipv6-tunnel;
    container ipv6-decap {
      description
        "IPv6 decap.";
      leaf ipv6-decap {
        type tunnel-decap-action-def;
        mandatory true;
        description
          "IPv6 decap operations.";
      }
      leaf hop-limit-action {
        type hop-limit-action-def;
        description
          "The hop limit actions:
           no-action or copy to inner header.";
      }
    }
  }
}
case mpls {
  if-feature mpls-tunnel;
  container label-pop {
    description
      "MPLS decap.";
    leaf label-pop {
      type mpls-label-action-def;
      mandatory true;
      description
        "Pop a label from the label stack.";
    }
    leaf ttl-action {
      type ttl-action-def;
      description
        "The label ttl actions:
         no-action or copy to inner label";
    }
  }
}
}
}

grouping route-attributes {
  description
    "Route attributes.";
  leaf route-preference {
    type uint32;
    mandatory true;
    description
      "ROUTE_PREFERENCE: This is a numerical value that
```



```
        allows for comparing routes from different
        protocols. Static configuration is also
        considered a protocol for the purpose of this
        field. It is also known as administrative-distance.
        The lower the value, the higher the preference.";
    }
    leaf local-only {
        type boolean ;
        mandatory true;
        description
            "Indicate whether the attributes is local only.";
    }
    container address-family-route-attributes{
        description
            "Address family related route attributes.";
        choice route-type {
            description
                "Address family related route attributes.";
            case ip-route-attributes {
            }
            case mpls-route-attributes {
            }
            case ethernet-route-attributes {
            }
        }
    }
}

container routing-instance {
    description
        "A routing instance, in the context of
        the RIB information model, is a collection
        of RIBs, interfaces, and routing parameters";
    leaf name {
        type string;
        mandatory true;
        description
            "The name of the routing instance.This MUST
            be unique across all routing instances in
            a given network device.";
    }
    list interface-list {
        key "name";
        description
            "This represents the list of interfaces associated
            with this routing instance. The interface list helps
            constrain the boundaries of packet forwarding.
            Packets coming on these interfaces are directly
```



```
    associated with the given routing instance. The
    interface list contains a list of identifiers, with
    each identifier uniquely identifying an interface.";
  leaf name {
    type if:interface-ref;
    description
      "A reference to the name of a network layer interface.";
  }
}
leaf router-id {
  type yang:dotted-quad;
  description
    "Router ID - 32-bit number in the form of a dotted quad.";
}
leaf lookup-limit {
  type uint8;
  description
    "A limit on how many levels of a lookup can be performed.";
}
list rib-list {
  key "name";
  description
    "A list of RIBs that are associated with the routing
    instance.";
  leaf name {
    type string;
    mandatory true;
    description
      "A reference to the name of each rib.";
  }
  leaf rib-family {
    type rib-family-def;
    mandatory true;
    description
      "The address family of a rib.";
  }
  leaf ip-rpf-check {
    type boolean;
    description
      "Each RIB can be optionally associated with a
      ENABLE_IP_RPF_CHECK attribute that enables Reverse
      path forwarding (RPF) checks on all IP routes in that
      RIB. Reverse path forwarding (RPF) check is used to
      prevent spoofing and limit malicious traffic.";
  }
}
list route-list {
  key "route-index";
  description
```



```
        "A list of routes of a rib.";
    uses route;
}
}
}

/*RPC Operations*/
rpc rib-add {
    description
        "To add a rib to a instance";
    input {
        leaf rib-name {
            type string;
            mandatory true;
            description
                "A reference to the name of the rib
                that is to be added.";
        }
        leaf rib-family {
            type rib-family-def;
            mandatory true;
            description
                "The address family of the rib.";
        }
        leaf ip-rpf-check {
            type boolean;
            description
                "Each RIB can be optionally associated with a
                ENABLE_IP_RPF_CHECK attribute that enables Reverse
                path forwarding (RPF) checks on all IP routes in that
                RIB. Reverse path forwarding (RPF) check is used to
                prevent spoofing and limit malicious traffic.";
        }
    }
}
output {
    leaf result {
        type boolean;
        mandatory true;
        description
            "Return the result of the rib-add operation.
            true - success;
            false - failed";
    }
    leaf reason {
        type string;
        description
            "The specific reason that causes the failure.";
    }
}
```



```
    }
  }

  rpc rib-delete {
    description
      "To delete a rib from a routing instance.
      After deleting the rib, all routes installed
      in the rib will be deleted as well.";
    input {
      leaf rib-name {
        type string;
        mandatory true;
        description
          "A reference to the name of the rib
          that is to be deleted.";
      }
    }
    output {
      leaf result {
        type boolean;
        mandatory true;
        description
          "Return the result of the rib-delete operation.
          true - success;
          false - failed";
      }
      leaf reason {
        type string;
        description
          "The specific reason that causes failure.";
      }
    }
  }
}

grouping route-operation-state {
  description
    "Route operation state.";
  leaf success-count {
    type uint32;
    mandatory true;
    description
      "The numbers of routes that are successfully
      added/deleted/updated.";
  }
  leaf failed-count {
    type uint32;
    mandatory true;
    description
```



```
        "The numbers of the routes that are failed
          to be added/deleted/updated.";
    }
    container failure-detail {
        description
            "The failure detail reflects the reason why a route
            operation fails. It is a array that includes the route
            index and error code of the failed route.";
        list failed-routes {
            key "route-index";
            description
                "The list of failed routes.";
            leaf route-index {
                type uint32;
                description
                    "The route index of the failed route.";
            }
            leaf error-code {
                type uint32;
                description
                    "The error code that reflects the failure reason.";
            }
        }
    }
}

rpc route-add {
    description
        "To add a route or a list of route to a rib";
    input {
        leaf return-failure-detail {
            type boolean;
            default false;
            description
                "Whether return the failure detail.
                true - return the failure detail;
                false - do not return the failure detail;
                the default is false.";
        }
        leaf rib-name {
            type string;
            mandatory true;
            description
                "A reference to the name of a rib.";
        }
        container routes {
            description
                "The routes to be added to the rib.";
        }
    }
}
```



```
list route-list {
  key "route-index";
  description
    "The list of routes to be added.";
  uses route-prefix;
  container route-attributes {
    uses route-attributes;
    description
      "The route attributes.";
  }
  container route-vendor-attributes {
    if-feature route-vendor-attributes;
    uses route-vendor-attributes;
    description
      "The route vendor attributes.";
  }
  container nexthop {
    uses nexthop;
    description
      "The nexthop of the added route.";
  }
}
}
}
output {
  uses route-operation-state;
}
}

rpc route-delete {
  description
    "To delete a route or a list of route from a rib";
  input {
    leaf return-failure-detail {
      type boolean;
      default false;
      description
        "Whether return the failure detail.
         true - return the failure detail;
         false - do not return the failure detail;
         the default is false.";
    }
    leaf rib-name {
      type string;
      mandatory true;
      description
        "A reference to the name of a rib.";
    }
  }
}
```



```
    container routes {
      description
        "The routes to be added to the rib.";
      list route-list{
        key "route-index";
        description
          "The list of routes to be deleted.";
        uses route-prefix;
      }
    }
  }
}
output {
  uses route-operation-state;
}
}

grouping route-update-options {
  description
    "Update options:
    1. update the nexthop
    2. update the route attributes
    3. update the route-vendor-attributes.";
  choice update-options {
    description
      "Update options:
      1. update the nexthop
      2. update the route attributes
      3. update the route-vendor-attributes.";
    case update-nexthop {
      container updated-nexthop {
        uses nexthop;
        description
          "The nexthop used for updating.";
      }
    }
    case update-route-attributes {
      container updated-route-attr {
        uses route-attributes;
        description
          "The route attributes used for updating.";
      }
    }
    case update-route-vendor-attributes {
      container updated-route-vendor-attr {
        uses route-vendor-attributes;
        description
          "The vender route attributes used for updating.";
      }
    }
  }
}
```



```
    }
  }
}

rpc route-update {
  description
    "To update a route or a list of route of a rib.
    The inputs:
      1. The match conditions, could be:
        a. route prefix, or
        b. route attributes, or
        c. nexthop;
      2. The update parameters to be used:
        a. new nexthop;
        b. new route attributes;nexthop
    Actions:
      1. update the nexthop
      2. update the route attributes
    The outputs:
      success-count - the number of routes updated;
      failed-count - the number of routes fail to update
      failure-detail - the detail failure info.
    ";
  input {
    leaf return-failure-detail {
      type boolean;
      default false;
      description
        "Whether return the failure detail.
        true - return the failure detail;
        false - do not return the failure detail;
        the default is false.";
    }
    leaf rib-name {
      type string;
      mandatory true;
      description
        "A reference to the name of a rib.";
    }
  }
  choice match-options {
    description
      "Match options.";
    case match-route-prefix {
      description
        "Update the routes that match route
        prefix(es) condition.";
      container input-routes {
        description
```



```
        "The matched routes to be updated.";
    list route-list {
        key "route-index";
        description
            "The list of routes to be updated.";
        uses route-prefix;
        uses route-update-options;
    }
}
}
case match-route-attributes {
    description
        "Update the routes that match the
        route attributes condition.";
    container input-route-attributes {
        description
            "The route attributes are used for matching.";
        uses route-attributes;
    }
    container update-parameters {
        description
            "Update options:
            1. update the nexthop
            2. update the route attributes
            3. update the route-vendor-attributes.";
        uses route-update-options;
    }
}
}
case match-route-vendor-attributes {
    if-feature route-vendor-attributes;
    description
        "Update the routes that match the
        vendor attributes condition";
    container input-route-vendor-attributes {
        description
            "The vendor route attributes are used for matching.";
        uses route-vendor-attributes;
    }
    container update-parameters-vendor {
        description
            "Update options:
            1. update the nexthop
            2. update the route attributes
            3. update the route-vendor-attributes.";
        uses route-update-options;
    }
}
}
case match-nexthop {
```



```
    description
      "Update the routes that match the nexthop.";
    container input-nexthop {
      description
        "The nexthop used for matching.";
      uses nexthop;
    }
    container update-parameters-nexthop {
      description
        "Update options:
        1. update the nexthop
        2. update the route attributes
        3. update the route-vendor-attributes.";
      uses route-update-options;
    }
  }
}
}
output {
  uses route-operation-state;
}
}
```

```
rpc nh-add {
  description
    "To add a nexthop to a rib.
    Inputs parameters:
      1. rib name
      2. nexthop;
    Actions:
      Add the nexthop to the rib
    Outputs:
      1.Operation result:
        true - success
        false - failed;
      2. nexthop identifier.";
  input {
    leaf rib-name {
      type string;
      mandatory true;
      description
        "A reference to the name of a rib.";
    }
    uses nexthop;
  }
  output {
    leaf result {
      type boolean;
    }
  }
}
```



```
        mandatory true;
        description
            "Return the result of the rib-add operation.
            true - success;
            false - failed;";
    }
    leaf reason {
        type string;
        description
            "The specific reason that causes the failure.";
    }
    leaf nexthop-id {
        type uint32;
        description
            "A nexthop identifier that is allocated to the nexthop.";
    }
}
}
}

rpc nh-delete {
    description
        "To delete a nexthop from a rib";
    input {
        leaf rib-name {
            type string;
            mandatory true;
            description
                "A reference to the name of a rib.";
        }
        uses nexthop;
    }
    output {
        leaf result {
            type boolean;
            mandatory true;
            description
                "Return the result of the rib-add operation.
                true - success;
                false - failed.";
        }
        leaf reason {
            type string;
            description
                "The specific reason that causes the failure.";
        }
    }
}
}
```



```
/*Notifications*/
notification nexthop-resolution-status-change {
  description
    "Nexthop resolution status (resolved/unresolved)
    notification.";
  container nexthop{
    description
      "The nexthop.";
    uses nexthop;
  }
  leaf nexthop-state {
    type nexthop-state-def;
    mandatory true;
    description
      "Nexthop resolution status (resolved/unresolved)
      notification.";
  }
}

notification route-change {
  description
    "Route change notification.";
  leaf rib-name {
    type string;
    mandatory true;
    description
      "A reference to the name of a rib.";
  }
  leaf rib-family {
    type rib-family-def;
    mandatory true;
    description
      "A reference to address family of a rib.";
  }
  uses route-prefix;
  leaf route-installed-state {
    type route-installed-state-def;
    mandatory true;
    description
      "Indicates whether the route got installed in the FIB.";
  }
  leaf route-state {
    type route-state-def;
    mandatory true;
    description
      "Indicates whether a route is fully resolved and
      is a candidate for selection.";
  }
}
```



```
    leaf route-change-reason {
      type route-reason-def;
      mandatory true;
      description
        "Return the reason that causes the route change.";
    }
  }
}
```

<CODE ENDS>

4. IANA Considerations

This document requests to register a URI in the "IETF XML registry" [[RFC3688](#)]:

```
-----
URI: urn:ietf:params:xml:ns:yang:ietf-i2rs-rib
Registrant Contact: The IESG.XML:
N/A, the requested URI is an XML namespace.
-----
```

This document requests to register a YANG module in the "YANG Module Names registry" [[RFC6020](#)]:

```
-----
name:          ietf-i2rs-rib
namespace:     urn:ietf:params:xml:ns:yang:ietf-i2rs-rib
prefix:        iir
reference:     RFC XXXX
-----
```

5. Security Considerations

This document introduces no extra new security threat and SHOULD follow the security requirements as stated in [[I-D.ietf-i2rs-architecture](#)].

6. Contributors

The following individuals also contribute to this document.

- o Zekun He, Tencent Holdings Ltd
- o Sujian Lu, Tencent Holdings Ltd
- o Jeffery Zhang, Juniper Networks

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", [RFC 6991](#), DOI 10.17487/RFC6991, July 2013, <<http://www.rfc-editor.org/info/rfc6991>>.

7.2. Informative References

- [I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", [draft-ietf-i2rs-architecture-09](#) (work in progress), March 2015.
- [I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Kini, S., and J. Medved, "Routing Information Base Info Model", [draft-ietf-i2rs-rib-info-model-08](#) (work in progress), October 2015.
- [I-D.ietf-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", [draft-ietf-i2rs-usecase-reqs-summary-01](#) (work in progress), May 2015.

Authors' Addresses

Lixing Wang
Individual

Email: wang_little_star@sina.com

Hariharan Ananthakrishnan
Packet Design

Email: hari@packetdesign.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Amit Dass
Ericsson
Torshamnsgatan 48.
Stockholm 16480
Sweden

Email: amit.dass@ericsson.com

Sriganesh Kini
Ericsson

Email: sriganesh.kini@ericsson.com

Nitin Bahadur
Bracket Computing

Email: nitin_bahadur@yahoo.com

