Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 15, 2018

L. Wang
Individual
M. Chen
Huawei
A. Dass
Ericsson
H. Ananthakrishnan
Packet Design
S. Kini
Individual
N. Bahadur
Bracket Computing
May 14, 2018

A YANG Data Model for Routing Information Base (RIB)
draft-ietf-i2rs-rib-data-model-15

Abstract

This document defines a YANG data model for the 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 https://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 November 15, 2018.

Table of Contents

1. Introduction

The Interface to the Routing System (I2RS) [RFC7921] provides read and write access to the information and state within the routing process that exists inside the routing elements, this is achieved via 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 the 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 [RFC7950][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

FIB: Forwarding Information Base

RPC: Remote Procedure Call

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

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

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 subsequent 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 address-family address-family-definition
         +--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-status
            |  |  ...
            |  +--rw route-attributes
            |  |  ...
            |  +--rw route-vendor-attributes
            +--rw nexthop-list* [nexthop-member-id]
               +--rw nexthop-member-id uint32
   rpcs:
      +---x rib-add
      |  +---w input
      |  |  +---w name        string
      |  |  +---w address-family      address-family-definition
      |  |  +---w ip-rpf-check?   boolean
      |  +--ro output
      |     +--ro result uint32
      |     +--ro reason? string
      +---x rib-delete
      |  +---w input
      |  |  +---w 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-definition
```

```
      +---n route-change
         +--ro rib-name                string
         +--ro address-family              address-family-definition
         +--ro route-index             uint64
         +--ro match
         |  +--ro (route-type)?
         |     +--:(ipv4)
         |     |  ...
         |     +--:(ipv6)
         |     |  ...
         |     +--:(mpls-route)
         |     |  ...
         |     +--:(mac-route)
         |     |  ...
         |     +--:(interface-route)
         |        ...
         +--ro route-installed-state route-installed-state-definition
         +--ro route-state         route-state-definition
         +--ro route-change-reason route-change-reason-definition
```

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 might be a limitation on how many levels of lookup can be practically performed. Therefore, a RIB data model needs to 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 needs 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 advertisement.

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. The routing protocol parameters control

the information available in the RIBs. More details about routing instance can be found in Section 2.2 of [I-D.ietf-i2rs-rib-info-model].

For a routing instance, there can be multiple RIBs. Therefore, this model uses "list" to express the RIBs. The structure tree is shown below:

```
+--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 address-family      address-family-definition
      +--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:

- ROUTE_PREFERENCE: See Section 2.3 of [I-D.ietf-i2rs-rib-info-model].
- ACTIVE: Indicates whether a route has at least one fully resolved nexthop and is therefore eligible for installation in the FIB.
- INSTALLED: Indicates whether the route got installed in the FIB.
- REASON - Indicates the specific reason that caused the failure, E.g. Not authorized.

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

A RIB 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 use cases (e.g., load balancing, protection, multicast or a combination of them), the nexthop is modeled as a multi-level structure and supports recursion. The first level of the nexthop includes the following four types:

- Base: The "base" nexthop is the foundation of all other nexthop types. It includes the follow basic nexthops:
  - nexthop-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-encapsulation

      * tunnel-decapsulation

      * 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 nexthops.

   o Protection: Designed for protection scenario where it normally will have primary and standby nexthop.

   o Replicate: Designed for multiple destinations forwarding.

   The structure tree of nexthop 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-definition
   |     +--:(nexthop-load-balance) {nexthop-load-balance}?
   |        +--rw nexthop-lb
   |           +--rw nexthop-list* [nexthop-member-id]
   |              +--rw nexthop-member-id uint32
   |              +--rw nexthop-lb-weight nexthop-lb-weight-definition
```

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-definition
|        +--:(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
```

```
|        |     +--rw ipv6-address       inet:ipv6-address
|        +--:(egress-interface-mac-nexthop)
|        |  +--rw egress-interface-mac-address
|        |     +--rw outgoing-interface if:interface-ref
|        |     +--rw ieee-mac-address yang:mac-address
|        +--:(tunnel-encap-nexthop) {nexthop-tunnel}?
|        |  +--rw tunnel-encap
|        |     +--rw (tunnel-type)?
|        |        +--:(ipv4) {ipv4-tunnel}?
|        |        |  +--rw ipv4-header
|        |        |     +--rw src-ipv4-address inet:ipv4-address
|        |        |     +--rw dest-ipv4-address inet:ipv4-address
|        |        |     +--rw protocol          uint8
|        |        |     +--rw ttl?              uint8
|        |        |     +--rw dscp?             uint8
|        |        +--:(ipv6) {ipv6-tunnel}?
|        |        |  +--rw ipv6-header
|        |        |     +--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?       inet:ipv6-flow-label
|        |        |     +--rw hop-limit?        uint8
|        |        +--:(mpls) {mpls-tunnel}?
|        |        |  +--rw mpls-header
|        |        |     +--rw label-operations* [label-oper-id]
|        |        |        +--rw label-oper-id uint32
|        |        |        +--rw (label-actions)?
|        |        |           +--:(label-push)
|        |        |           |  +--rw label-push
|        |        |           |     +--rw label        uint32
|        |        |           |     +--rw s-bit?       boolean
|        |        |           |     +--rw tc-value? uint8
|        |        |           |     +--rw ttl-value? uint8
|        |        |           +--:(label-swap)
|        |        |              +--rw label-swap
|        |        |                 +--rw out-label     uint32
|        |        |                 +--rw ttl-action? ttl-action-
definition
|        |        +--:(gre) {gre-tunnel}?
|        |        |  +--rw gre-header
|        |        |     +--rw (dest-address-type)?
|        |        |     |  +--:(ipv4)
|        |        |     |  |  +--rw ipv4-dest inet:ipv4-address
|        |        |     |  +--:(ipv6)
|        |        |     |     +--rw ipv6-dest inet:ipv6-address
|        |        |     +--rw protocol-type uint16
|        |        |     +--rw key?          uint64
```

```
|        |        +--:(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?       inet:ipv6-flow-label
|        |        |     |     +--rw hop-limit?        uint8
|        |        |     +--rw virtual-subnet-id uint32
|        |        |     +--rw flow-id?          uint8
|        |        +--:(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? inet:ipv6-flow-label
|        |              |     +--rw hop-limit?           uint8
|        |              +--rw vxlan-identifier     uint32
|        +--:(tunnel-decapsulation-nexthop) {nexthop-tunnel}?
|        |  +--rw tunnel-decapsulation
|        |     +--rw (tunnel-type)?
|        |        +--:(ipv4) {ipv4-tunnel}?
|        |        |  +--rw ipv4-decapsulation
|        |        |     +--rw ipv4-decapsulation tunnel-decapsulation-
action-definition
|        |        |     +--rw ttl-action?   ttl-action-definition
|        |        +--:(ipv6) {ipv6-tunnel}?
|        |        |  +--rw ipv6-decapsulation
|        |        |     +--rw ipv6-decapsulation tunnel-decapsulation-
action-definition
|        |        |     +--rw hop-limit-action? hop-limit-action-
```

```
definition
|        |        +--:(mpls) {mpls-tunnel}?
|        |           +--rw label-pop
|        |              +--rw label-pop     mpls-label-action-definition
|        |              +--rw ttl-action?   ttl-action-definition
|        +--:(logical-tunnel-nexthop) {nexthop-tunnel}?
|        |  +--rw logical-tunnel
|        |     +--rw tunnel-type tunnel-type-definition
|        |     +--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:

- rib-add: Add a RIB to a routing instance. A name of the RIB, address family of the RIB and (optionally) 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 caused the failure.
- rib-delete: 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 caused the failure.
- route-add: 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 details 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. However, in situations when the nexthop already exists and the nexthop-id is known, this action is not expected.

The output is a combination of the route operation states while querying the appropriate node in the data tree that include:

* success-count: the number of routes that were successfully added;

* failed-count: the number of the routes that failed to be added;

* failure-detail: shows the specific routes that failed to be added.

- route-delete: Delete a route or a set of routes from a RIB. A name of the RIB, the route prefix(es) and whether to return failure details are passed as the input parameters. The output is a combination of route operation states that include:

  * success-count: the number of routes that were successfully deleted;

  * failed-count: the number of the routes that failed to be deleted;

  * failure-detail: shows the specific routes that failed to be deleted.

- route-update: 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 number of routes that were successfully updated;

  * failed-count: the number of the routes that failed to be updated;

  * failure-detail: shows the specific routes that failed to be updated.

- nh-add: 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 caused the failure.

o nh-delete: 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 caused the failure.

The structure tree of rpcs is shown in following figure.

```
rpcs:
   +---x rib-add
   |  +---w input
   |  |  +---w rib-name        string
   |  |  +---w address-family      address-family-definition
   |  |  +---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. An implementation of this RIB data model MUST support sending two kinds of asynchronous notifications.

1. Route change notification:

o Installed (Indicates whether the route got installed in the FIB) ;

o Active (Indicates whether a route has at least one fully resolved nexthop and is therefore eligible for installation in the FIB) ;

o Reason - E.g. Not authorized

2. Nexthop resolution status notification

Nexthops can be fully resolved or unresolved.

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

An unresolved nexthop is something that requires the RIB manager to determine the final resolved nexthop. In one example, 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 address is 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.

An implementation of this RIB data model MUST support sending route-change notifications whenever a route transitions between the following states:

o from the active state to the inactive state

o from the inactive state to the active state

o from the installed state to the uninstalled state

o from the uninstalled state to the installed state

A single notification MAY be used when a route transitions from inactive/uninstalled to active/installed or in the other direction.

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

Figure 7: Notifications Structure

3. YANG Modules

<CODE BEGINS> file "ietf-i2rs-rib@2018-04-23.yang"

```
module ietf-i2rs-rib {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-i2rs-rib";
  prefix "iir";

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991";
  }

  import ietf-interfaces {
    prefix if;
    reference "RFC 8344";
  }

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991";
  }

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

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

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

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

     Editor:   Hariharan Ananthakrishnan
               <mailto:hari@packetdesign.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.
      Copyright (c) <2018> IETF Trust and the persons
      identified as authors of the code.  All rights reserved.";
   revision "2018-04-23" {
     description "initial revision";
     reference "RFC XXXX: draft-ietf-i2rs-data-model-10";
    // RFC Ed.: replace XXXX with actual RFC number and remove
    // this note
   }

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

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

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

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

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

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

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

   feature mpls-tunnel {
     description
       "This feature means that a node supports
        MPLS tunnel encapsulation capability.";
   }

   feature vxlan-tunnel {
     description
       "This feature means that a node supports
        VXLAN tunnel encapsulation capability.";
     reference "RFC7348";
   }

   feature gre-tunnel {
     description
       "This feature means that a node supports
        GRE tunnel encapsulation capability.";
     reference "RFC2784";
   }

   feature nvgre-tunnel {
     description
       "This feature means that a node supports
        NvGRE tunnel encapsulation capability.";
     reference "RFC7637";
   }

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

   //Identities and Type Definitions
   identity mpls-label-action {
     description
       "Base identity 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 exchange 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-definition {
     type identityref {
       base "mpls-label-action";
     }
     description
       "MPLS label action definition.";
   }

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

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

   identity ipv6-decapsulation {
     base "tunnel-decapsulation-action";
     description
       "IPv6 tunnel decapsulation.";
```

```
   }

   typedef tunnel-decapsulation-action-definition {
     type identityref {
       base "tunnel-decapsulation-action";
     }
     description
       "Tunnel decapsulation definition.";
   }

   identity ttl-action {
     description
       "Base identity 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 the inner header.";
   }

   identity decrease-and-copy-to-inner {
     base "ttl-action";
     description
       "Decrease TTL by one and copy the TTL
        to the 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-definition {
     type identityref {
```

```
      base "ttl-action";
    }
    description
      "TTL action definition.";
  }

  identity hop-limit-action {
    description
      "Base identity 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 the inner header.";
  }

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

  identity special-nexthop {
    description
      "Base identity 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 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-definition {
     type identityref {
       base "special-nexthop";
     }
     description
       "Special nexthop definition.";
   }

   identity ip-route-match-type {
     description
       "Base identity 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
      "Source and Destination route match type";
  }

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

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

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

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

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

  identity ieee-mac-address-family {
```

```
    base "address-family";
    description
      "MAC RIB address family.";
  }

  typedef address-family-definition {
    type identityref {
      base "address-family";
    }
    description
      "RIB address family definition.";
  }

  identity route-type {
    description
      "Base identity 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-definition {
     type identityref {
       base "route-type";
     }
     description
       "Route type definition.";
   }

   identity tunnel-type {
     description
       "Base identity 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-definition {
     type identityref {
       base "tunnel-type";
     }
     description
       "Tunnel type definition.";
   }

   identity route-state {
     description
       "Base identity 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-definition {
     type identityref {
       base "route-state";
     }
     description
       "Route state definition.";
   }

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

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

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

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

   identity route-installed-state {
     description
       "Base identity 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-definition {
     type identityref {
       base "route-installed-state";
     }
     description
       "Route installed state definition.";
   }

 //Route change reason identities

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

```
   identity lower-route-preference {
     base "route-change-reason";
     description
       "This route was installed in the FIB because it had
        a lower route preference value (and thus was more
        preferred) than the route it replaced.";
   }

   identity higher-route-preference {
     base "route-change-reason";
     description
       "This route was uninstalled from the FIB because it had
        a higher route preference value (and thus was less
        preferred) than the route that replaced it.";
   }

   identity resolved-nexthop {
     base "route-change-reason";
     description
       "This route was made active because at least
        one of its nexthops was resolved.";
   }

   identity unresolved-nexthop {
     base "route-change-reason";
     description
       "This route was made inactive because all of
        its nexthops are unresolved.";
   }

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

   typedef nexthop-preference-definition {
     type uint8 {
       range "1..99";
     }
     description
       "Nexthop-preference is used for protection schemes.
        It is an integer value between 1 and 99. Lower
        values are more preferred. To download N
        nexthops to the FIB, the N nexthops with the lowest
        value are selected. If there are more than N
```

```
        nexthops that have the same preference, an
        implementation of i2rs client should select N
        nexthops and download them, as for how to select
        the nexthops is left to the implementations.";
    }

    typedef nexthop-lb-weight-definition {
      type uint8 {
        range "1..99";
      }
      description
        "Nexthop-lb-weight is used for load-balancing.
         Each list member SHOULD be assigned a weight
         between 1 and 99. The weight determines the
         proportion of traffic to be sent over a nexthop
         used for forwarding as a ratio of the weight of
         this nexthop divided by the sum of the weights
         of all the nexthops of this route that are used
         for forwarding. To perform equal load-balancing,
         one MAY specify a weight of 0 for all the member
         nexthops. The value 0 is reserved for equal
         load-balancing and if applied, MUST be applied
         to all member nexthops.
         Note: The weight of 0 is specially because of
         historical reasons. It's typically used in
         hardware devices to signify ECMP";
    }

    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.";
                  leaf dest-ipv4-prefix {
                    type inet:ipv4-prefix;
                    mandatory true;
```

```
                    description
                      "The IPv4 destination address of the match.";
                  }
                  leaf src-ipv4-prefix {
                    type inet:ipv4-prefix;
                    mandatory true;
                    description
                      "The IPv4 source address of the match";
                  }
                }
              }
            }
          }
        }
        case ipv6 {
          description
            "IPv6 route case.";
          container ipv6 {
            description
              "IPv6 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-ipv6-address {
                leaf dest-ipv6-prefix {
                  type inet:ipv6-prefix;
                  mandatory true;
                  description
                    "An IPv6 destination address as the match.";
                }
              }
              case src-ipv6-address {
                leaf src-ipv6-prefix {
                  type inet:ipv6-prefix;
                  mandatory true;
                  description
                    "An IPv6 source address as the match.";
                }
              }
              case dest-src-ipv6-address {
                container dest-src-ipv6-address {
                  description
                    "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 yang:mac-address;
            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 routes.";
     uses route-prefix;
     container nexthop {
       description
         "The nexthop of the route.";
       uses nexthop;
     }
     //In the information model, it is called route-statistic
     container route-status {
       description
         "The status information of the route.";
       leaf route-state {
         type route-state-definition;
         config false;
         description
           "Indicate a route's state: Active or Inactive.";
       }
       leaf route-installed-state {
         type route-installed-state-definition;
         config false;
         description
           "Indicate that a route's installed states:
            Installed or uninstalled.";
       }
       leaf route-reason {
         type route-change-reason-definition;
         config false;
         description
           "Indicate the reason that caused the route change.";
       }
     }
     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 nexthops.";
      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-definition;
        mandatory true;
        description
          "Nexthop-preference is used for protection schemes.
           It is an integer value between 1 and 99.  Lower
           values are more preferred. To download a
           primary/standby/tertiary group to the FIB, the
           nexthops that are resolved and are most preferred
           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-definition;
        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-replicate {
        if-feature nexthop-replicate;
        container nexthop-replicate {
          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-lb {
          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-definition;
          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 yang:mac-address;
            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 encapsulation representing an IP
```

```
              tunnel or MPLS tunnel or others as defined in info
              model. An optional egress interface can be chained
              to the tunnel encapsulation 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-decapsulation-nexthop {
          if-feature nexthop-tunnel;
          container tunnel-decapsulation {
            uses tunnel-decapsulation;
            description
              "This is to specify the decapsulation of a tunnel header.";
          }
        }
        case logical-tunnel-nexthop {
          if-feature nexthop-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-nexthop {
          leaf rib-name {
            type string;
            description
              "A nexthop 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 nexthop-identifier {
          leaf nexthop-ref {
            type nexthop-ref;
            mandatory true;
            description
              "A nexthop reference that points to a nexthop.";
          }
        }
      }
    }

    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-definition;
      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 inet:ipv6-flow-label;
        description
          "The flow label of the header.";
      }
      leaf hop-limit {
        type uint8 {
          range "1..255";
        }
        description
          "The hop limit of 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 uint8;
      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 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-definition;
              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 information.";
    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-decapsulation {
    description
      "Tunnel decapsulation information.";
    choice tunnel-type {
      description
        "Nexthop tunnel type options.";
      case ipv4 {
        if-feature ipv4-tunnel;
        container ipv4-decapsulation {
          description
            "IPv4 decapsulation.";
          leaf ipv4-decapsulation  {
            type tunnel-decapsulation-action-definition;
            mandatory true;
            description
              "IPv4 decapsulation operations.";
          }
          leaf ttl-action {
            type ttl-action-definition;
            description
              "The ttl actions:
               no-action or copy to inner header.";
          }
        }
      }
```

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

   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;
        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 address-family {
          type address-family-definition;
          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;
      }
      // This is a list that maintains the nexthops added to the RIB.
      uses nexthop-list;
    }
  }

  //RPC Operations
  rpc rib-add {
    description
      "To add a RIB to a instance";
    input {
      leaf name {
        type string;
        mandatory true;
        description
          "A reference to the name of the RIB
           that is to be added.";
      }
      leaf address-family {
        type address-family-definition;
        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 caused 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 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 caused 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.
              0 - Reserved.
              1 - Trying to add a repeat route;
              2 - Trying to delete or update a route that is not exist;
              3 - Malformed route attribute;
              ";
         }
       }
     }
   }

   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 vendor 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-parametors {
            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 caused 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 caused 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-definition;
      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 address-family {
      type address-family-definition;
      mandatory true;
      description
        "The address family of a RIB.";
    }
    uses route-prefix;
    leaf route-installed-state {
      type route-installed-state-definition;
      mandatory true;
      description
        "Indicates whether the route got installed in the FIB.";
    }
    leaf route-state {
```

```
        type route-state-definition;
        mandatory true;
        description
          "Indicates whether a route is active or inactive.";
      }
      list route-change-reasons {
        key "route-change-reason";
        description
          "The reasons that cause the route change. A route
           change that may result from several reasons. For
           example, a nexthop becoming resolved will make a
           route A active which is of better preference than
           a currently active route B, which results in the
           route A being installed";
        leaf route-change-reason {
          type route-change-reason-definition;
          mandatory true;
          description
            "The reason that caused the route change.";
        }
      }
    }
  }

 <CODE ENDS>
```

4. IANA Considerations

This document registers a URI in the "ns" registry with 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" [RFC7950]:

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

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The YANG modules define information that can be configurable in certain instances, for example, a RIB, a route, a nexthop can be created or deleted by client applications, the YANG modules also define RPCs that can be used by client applications to add/delete RIBs, routes and nexthops. In such cases, a malicious client could attempt to remove, add or update a RIB, a route, a nexthop, by creating or deleting corresponding elements in the RIB, route and nexthop lists, respectively. Removing a RIB or a route could lead to disruption or impact in performance of a service, updating a route may lead to suboptimal path and degradation of service levels as well as possibly disruption of service. For those reasons, it is important that the NETCONF access control model is vigorously applied to prevent misconfiguration by unauthorized clients.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability in the ietf-i2rs-rib module:

- RIB: A malicious client could attempt to remove a RIB from a routing instance, for example in order to sabotage the services provided by the RIB, or to add a RIB to a routing instance, hence to inject unauthorized traffic into the nexthop.

- route:A malicious client could attempt to remove or add a route from/to a RIB, for example in order to sabotage the services provided by the RIB.

- nexthop: A malicious client could attempt to remove or add a nexthop from/to RIB, which may lead to suboptimal path and

degradation of service levels as well as possibly disruption of service.

6. Contributors

The following individuals also contribute to this document.

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

7. Acknowledgements

The authors would like to thank Chris Bowers, John Scudder, Tom Petch, Mike McBride and Ebben Aries for his review, suggestion and comments to this document.

8. References

8.1. Normative References

[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-17 (work in progress), May 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

8.2. Informative References

[I-D.ietf-i2rs-usecase-reqs-summary] Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", draft-ietf-i2rs-usecase-reqs-summary-03 (work in progress), November 2016.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <https://www.rfc-editor.org/info/rfc2784>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <https://www.rfc-editor.org/info/rfc7637>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <https://www.rfc-editor.org/info/rfc7921>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Authors' Addresses

Lixing Wang
Individual

Email: wang_little_star@sina.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Amit Dass
Ericsson

Email: amit.dass@ericsson.com

Hariharan Ananthakrishnan
Packet Design

Email: hari@packetdesign.com

Sriganesh Kini
Individual

Email: sriganeshkini@gmail.com

Nitin Bahadur
Bracket Computing

Email: nitin_bahadur@yahoo.com