Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

L. Wang
Huawei
H. Ananthakrishnan
Packet Design
M. Chen
Huawei
A. Dass
S. Kini
Ericsson
N. Bahadur
Bracket Computing
March 09, 2015

Data Model for RIB I2RS protocol
draft-wang-i2rs-rib-data-model-02

Abstract

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

Requirements Language

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on September 10, 2015.

Table of Contents

1. Introduction

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

This document defines a YANG [RFC6020][RFC6021] data model for the RIB that satisfies the RIB use cases and aligns with the RIB information model.

1.1. Definitions and Acronyms

RIB: Routing Information Base

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

1.2. Tree Diagrams

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

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

2. Model Structure

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

```
module: i2rs-rib
   +--rw nexthop-capacity
   |  ...
   +--rw nexthop-tunnel-encap-capacity
   |  ...
   +--rw routing-instance
      +--rw instance-name     string
      +--rw interface-list* [name]
      |  +--rw name        if:interface-ref
      +--rw router-id?        yang:dotted-quad
```

```
            +--rw rib-list* [rib-name]
               +--rw rib-name               string
               +--rw rib-family             rib-family-def
               +--rw enable-ip-rpf-check?        boolean
               +--rw route-list* [route-index]
                  +--rw route-index                uint64
                  +--rw route-type                 route-type-def
                  +--rw match
                  |  +--rw (rib-route-type)?
                  |     +--:(ipv4)
                  |     |  ...
                  |     +--:(ipv6)
                  |     |  ...
                  |     +--:(mpls-route)
                  |     |  ...
                  |     +--:(mac-route)
                  |     |  ...
                  |     +--:(interface-route)
                  |        ...
                  +--rw nexthop
                  |  +--rw nexthop-id           uint32
                  |  +--rw (nexthop-type)?
                  |     +--:(nexthop-base)
                  |     |  ...
                  |     +--:(nexthop-protection)
                  |     |  ...
                  |     +--:(nexthop-load-balance)
                  |     |  ...
                  |     +--:(nexthop-replicate)
                  |        ...
                  +--rw route-statistic
                  |  ...
                  +--rw route-attributes
                  |  +--rw route-preference           uint32
                  |  +--rw local-only                 boolean
                  |  +--rw address-family-route-attributes
                  |     +--rw (route-type)?
                  |        +--:(ip-route-attributes)
                  |        +--:(mpls-route-attributes)
                  |        +--:(eThernet-route-attributes)
                  +--rw route-vendor-attributes
      notifications:
         +---n nexthop-resolution-status-change
         |  +--ro nexthop
         |  |  +--ro nexthop-id           uint32
         |  |  +--ro (nexthop-type)?
         |  |     +--:(nexthop-base)
         |  |     |  ...
```

```
|  |     +--:(nexthop-protection)
|  |     |  ...
|  |     +--:(nexthop-load-balance)
|  |     |  ...
|  |     +--:(nexthop-replicate)
|  |        ...
|  +--ro nexthop-state        nexthop-state-def
+---n route-change
   +--ro instance-name            string
   +--ro rib-name                 string
   +--ro rib-family               rib-family-def
   +--ro route-index              uint64
   +--ro route-type               route-type-def
   +--ro match
   |  +--ro (rib-route-type)?
   |     +--:(ipv4)
   |     |  ...
   |     +--:(ipv6)
   |     |  ...
   |     +--:(mpls-route)
   |     |  +--ro mpls-label             uint32
   |     +--:(mac-route)
   |     |  +--ro mac-address            uint32
   |     +--:(interface-route)
   |        +--ro interface-identifier if:interface-ref
   +--ro route-installed-state route-installed-state-def
   +--ro route-state           route-state-def
   +--ro route-reason          route-reason-def
```

Figure 1 Overview of I2RS module

2.1. RIB Capability

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

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

The structure of the next-hop-capacity and the nexthop-tunnel-encap-capacity is shown in the following figure:

Editor Notes: this version only includes the nexthop-hop and nexthop-tunnel-encap capabilities, there may also need to define RIB capabilities in future revision.

```
+--rw nexthop-capacity
|  +--rw support-tunnel?         boolean
|  +--rw support-chains?         boolean
|  +--rw support-list-of-list?   boolean
|  +--rw support-replication?    boolean
|  +--rw support-weighted?       boolean
|  +--rw support-protection?     boolean
|  +--rw lookup-limit?           uint8
+--rw nexthop-tunnel-encap-capacity
|  +--rw support-ipv4?    boolean
|  +--rw support-ipv6?    boolean
|  +--rw support-mpls?    boolean
|  +--rw support-gre?     boolean
|  +--rw support-vxlan?   boolean
|  +--rw support-nvgre?   boolean
```

Figure 2 RIB Capability

2.2. Routing Instance and Rib

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

As described in [I-D.ietf-i2rs-rib-info-model], there will be multiple routing instances for a router. At the same time, for a routing instance, there would be multiple RIBs as well. Therefore, this model uses "list" to express the RIBs. The structure tree is shown as following figure.

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

Figure 3 Routing Instance

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 is fully resolved and is a candidate for selection.
- INSTALLED: Indicates whether the route got installed in the FIB.

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

For a RIB, there will have a number of routes, so the routes are expressed as a list under the rib list.

```
 +--rw route-list* [route-index]
    +--rw route-index                uint64
    +--rw route-type                 route-type-def
    +--rw match
    |  +--rw (rib-route-type)?
    |     +--:(ipv4)
    |     |  +--rw ipv4
    |     |     +--rw ipv4-route-type                  ip-route-type-def
    |     |     +--rw (ip-route-type)?
    |     |        +--:(destination-ipv4-address)
    |     |        |  +--rw destination-ipv4-prefix inet:ipv4-prefix
    |     |        +--:(source-ipv4-address)
    |     |        |  +--rw source-ipv4-prefix         inet:ipv4-prefix
    |     |        +--:(destination-source-ipv4-address)
    |     |           +--rw destination-source-ipv4-address
    |     |              +--rw destination-ipv4-prefix inet:ipv4-prefix
    |     |              +--rw source-ipv4-prefix inet:ipv4-prefix
    |     +--:(ipv6)
    |     |  +--rw ipv6
    |     |     +--rw ipv6-route-type                  ip-route-type-def
    |     |     +--rw (ip-route-type)?
    |     |        +--:(destination-ipv6-address)
    |     |        |  +--rw destination-ipv6-prefix inet:ipv6-prefix
    |     |        +--:(source-ipv6-address)
    |     |        |  +--rw source-ipv6-prefix        inet:ipv6-prefix
    |     |        +--:(destination-source-ipv6-address)
    |     |           +--rw destination-source-ipv6-address
    |     |              +--rw destination-ipv6-prefix inet:ipv6-prefix
    |     |              +--rw source-ipv6-prefix inet:ipv6-prefix
    |     +--:(mpls-route)
    |     |  +--rw mpls-label              uint32
    |     +--:(mac-route)
    |     |  +--rw mac-address             uint32
    |     +--:(interface-route)
    |        +--rw interface-identifier        if:interface-ref
    +--rw nexthop
       ... (refer to sec.2.4)
```

Figure 4 Route

2.4. Nexthop

A nexthop represents an object resulting from a route lookup. As illustrated in Section 2.4 of [I-D.ietf-i2rs-rib-info-model], to support various of use cases (e.g., load balance, protection, multicast or the combination of them), the nexthop is modelled as a multi-level structure and supports recursion. The first level of the nexthop includes the following four types:

- Base: The "base" nexthop itself is a hierarchical structure, it is the base of all other nexthop types. The first level of the base nexthop includes special-nexthop and nexthop-chain. The nexthop-chain can have one or more nexthop chain members, each member is one of the four types (as listed below) of specific nexthop. Other first level nexthop (e.g., load-balance, protection and replicate) will finally be iterated to a "base" nexthop.
  - nexthop-id
  - egress-interface
  - logical-tunnel
  - tunnel-encap
- Load-balance: Designed for load-balance case.
- Protection: Designed for protection scenario where it normally will have primary and standby nexthop.
- Replicate: Designed for multiple destinations forwarding.

The structure tree of nexthop is shown in the following figures.

```
+--rw nexthop
|  +--rw nexthop-id           uint32
|  +--rw (nexthop-type)?
|     +--:(nexthop-base)
|     |  +--rw nexthop-base
|     |     +--rw nexthop-chain* [nexthop-chain-id]
|     |        +--rw nexthop-chain-id        uint32
|     |        +--rw (nexthop-chain-type)?
|     |              ... (refer to Figure 6)
|     +--:(nexthop-protection)
|     |  +--rw nexthop-protection-list* [nexthop-protection-id]
|     |     +--rw nexthop-protection-id     uint32
|     |     +--rw nexthop-preference       nexthop-preference-def
|     |     +--rw nexthop                  nexthop-ref
|     +--:(nexthop-load-balance)
|     |  +--rw nexthop-lb
|     |     +--rw nexthop-lbs* [nexthop-lbs-id]
|     |        +--rw nexthop-lbs-id        uint32
|     |        +--rw nhop-lb-weight        nhop-lb-weight-def
|     |        +--rw nexthop-lb-member        nexthop-ref
|     +--:(nexthop-replicate)
|        +--rw nexthop-replicate
|           +--rw nexthop-replicates* [nexthop-replicates-id]
|              +--rw nexthop-replicates-id        uint32
|              +--rw nexthop-replicate?        nexthop-ref
```

Figure 5 Nexhop

Figure 6 (as shown blow) is a sub-tree of nexthop, it’s under the nexthop chain node.

```
+--rw (nexthop-chain-type)?
   +--:(nexthop-chain-member-special)
   |  +--rw nexthop-chain-member-special
   |     +--rw nexthop-chain-member-special? special-nexthop-def
   +--:(nexthop-chain-member-identifier)
   |  +--rw (nexthop-identifier-type)?
   |     +--:(nexthop-chain-name)
   |     |  +--rw nexthop-chain-name         string
   |     +--:(nexthop-chain-id)
   |        +--rw nexthop-chain-id           uint32
   +--:(egress-interface-next-hop)
   |  +--rw outgoing-interface               if:interface-ref
   +--:(ipv4-address-next-hop)
   |  +--rw next-hop-ipv4-address            inet:ipv4-address
   +--:(ipv6-address-next-hop)
   |  +--rw next-hop-ipv6-address            inet:ipv6-address
   +--:(egress-interface-ipv4-next-hop)
```

```
|  +--rw next-hop-egress-interface-ipv4-address
|     +--rw outgoing-interface            if:interface-ref
|     +--rw next-hop-egress-ipv4-address inet:ipv4-address
+--:(egress-interface-ipv6-next-hop)
|  +--rw next-hop-egress-interface-ipv6-address
|     +--rw outgoing-interface           if:interface-ref
|     +--rw next-hop-egress-ipv6-address inet:ipv4-address
+--:(egress-interface-mac-next-hop)
|  +--rw next-hop-egress-interface-mac-address
|     +--rw outgoing-interface        if:interface-ref
|     +--rw ieee-mac-address          uint32
+--:(tunnel-encap-next-hop)
|  +--rw tunnel-encap
|     +--rw (tunnel-type)?
|     |  +--:(ipv4)
|     |  |  +--rw source-ipv4-address inet:ipv4-address
|     |  |  +--rw destination-ipv4-address inet:ipv4-address
|     |  |  +--rw protocol                 uint8
|     |  |  +--rw ttl?                     uint8
|     |  |  +--rw dscp?                    uint8
|     |  +--:(ipv6)
|     |  |  +--rw source-ipv6-address inet:ipv6-address
|     |  |  +--rw destination-ipv6-address inet:ipv6-address
|     |  |  +--rw next-header              uint8
|     |  |  +--rw traffic-class?           uint8
|     |  |  +--rw flow-label?              uint16
|     |  |  +--rw hop-limit?               uint8
|     |  +--:(mpls)
|     |  |  +--rw (mpls-action-type)?
|     |  |     +--:(mpls-push)
|     |  |     |  +--rw mpls-push          boolean
|     |  |     |  +--rw mpls-label         uint32
|     |  |     |  +--rw s-bit?             boolean
|     |  |     |  +--rw tos-value?         uint8
|     |  |     |  +--rw ttl-value?         uint8
|     |  |     +--:(mpls-pop)
|     |  |        +--rw mpls-pop           boolean
|     |  |        +--rw ttl-action?        uint8
|     |  +--:(gre)
|     |  |  +--rw gre-ip-destination        inet:ipv4-address
|     |  |  +--rw gre-protocol-type         inet:ipv4-address
|     |  |  +--rw gre-key?                  uint64
|     |  +--:(nvgre)
|     |     +--rw (nvgre-type)?
|     |     |  +--:(ipv4)
|     |     |  |  +--rw source-ipv4-address inet:ipv4-address
|     |     |  |  +--rw destination-ipv4-address inet:ipv4-address
|     |     |  |  +--rw protocol                 uint8
```

```
|     |     |  |  +--rw ttl?                      uint8
|     |     |  |  +--rw dscp?                     uint8
|     |     |  +--:(ipv6)
|     |     |     +--rw source-ipv6-address inet:ipv6-address
|     |     |     +--rw destination-ipv6-address inet:ipv6-address
|     |     |     +--rw next-header              uint8
|     |     |     +--rw traffic-class?           uint8
|     |     |     +--rw flow-label?              uint16
|     |     |     +--rw hop-limit?               uint8
|     |     +--rw virtual-subnet-id           uint32
|     |     +--rw flow-id?                    uint16
|     +--rw outgoing-interface?         string
+--:(logical-tunnel-next-hop)
|  +--rw logical-tunnel
|     +--rw tunnel-type        tunnel-type-def
|     +--rw tunnel-name        string
+--:(rib-name)
    +--rw rib-name?                             string
```

Figure 6 Nexthop Chain

2.5. Notifications

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

1. Route change notification:

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

o Active (Indicates whether a route is fully resolved and is a candidate for selection) ;

o Reason - E.g. Not authorized

2. Nexthop resolution status notification

Nexthops can be fully resolved nexthops or an unresolved nexthop.

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

An unresolved nexthop is something that requires the RIB manager to determine the final resolved nexthop. For example, in a case when a nexthop could be an IP address. The RIB manager would resolve how to

reach that IP address, e.g. by checking if that particular IP is address reachable by regular IP forwarding or by a MPLS tunnel or by both. If the RIB manager cannot resolve the nexthop, then the nexthop remains in an unresolved state and is NOT a suitable candidate for installation in the FIB.

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

```
 notifications:
    +---n nexthop-resolution-status-change
    |  +--ro nexthop
    |  |  +--ro nexthop-id           uint32
    |  |  +--ro (nexthop-type)?
    |  |     +--:(nexthop-base)
    |  |     |  +--ro nexthop-base
    |  |     |     +--ro nexthop-chain* [nexthop-chain-id]
    |  |     |        +--ro nexthop-chain-id     uint32
    |  |     |        +--ro (nexthop-chain-type)?
    |  |     |           ...
    |  |     +--:(nexthop-protection)
    |  |     |  +--ro nexthop-protection-list* [nexthop-protection-id]
    |  |     |     +--ro nexthop-protection-id     uint32
    |  |     |     +--ro nexthop-preference       nexthop-preference-def
    |  |     |     +--ro rw nexthop                  nexthop-ref
    |  |     +--:(nexthop-load-balance)
    |  |     |  +--ro nexthop-lb
    |  |     |     +--ro nexthop-lbs* [nexthop-lbs-id]
    |  |     |        +--ro nexthop-lbs-id       uint32
    |  |     |        +--ro nhop-lb-weight     nhop-lb-weight-def
    |  |     |        +--ro nexthop-lb-member nexthop-ref
    |  |     +--:(nexthop-replicate)
    |  |        +--ro nexthop-replicate
    |  |           +--ro nexthop-replicates* [nexthop-replicates-id]
    |  |              +--ro nexthop-replicates-id     uint32
    |  |              +--ro nexthop-replicate?       nexthop-ref
    |  +--ro nexthop-state     nexthop-state-def
    +---n route-change
       +--ro instance-name            string
       +--ro rib-name                 string
       +--ro rib-family               rib-family-def
       +--ro route-index              uint64
       +--ro route-type               route-type-def
       +--ro match
       |  +--ro (rib-route-type)?
       |     +--:(ipv4)
       |     |  +--ro ipv4
       |     |     ...
       |     +--:(ipv6)
```

```
       |     |  +--ro ipv6
       |     |     ...
       |     +--:(mpls-route)
       |     |  +--ro mpls-label             uint32
       |     +--:(mac-route)
       |     |  +--ro mac-address            uint32
       |     +--:(interface-route)
       |        +--ro interface-identifier if:interface-ref
       +--ro route-installed-state route-installed-state-def
       +--ro route-state              route-state-def
       +--ro route-reason             route-reason-def
```

Figure 7 Notifications

3. YANG Modules

```
//<code begins> file "i2rs rib@2015-03-09.yang"

module i2rs-rib {
  namespace "urn:TBD1:params:xml:ns:yang:rt:i2rs:rib";
    // replace with iana namespace when assigned
    prefix "i2rs-rib";

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-routing {
    prefix "rt";
  }

  organization
    "TBD2";
  contact
     "email: wang_little_star@sina.com
      email: hari@packetdesign.com
      email: mach.chen@huawei.com
      email: amit.dass@ericsson.com
      email: sriganesh.kini@ericsson.com
      email: nitin_bahadur@yahoo.com";

  description
    "
```

```
      terms and acronyms

      isis (isis):intermediate system to intermediate system

      ip (ip): internet protocol

      ipv4 (ipv4):internet protocol version 4

      ipv6 (ipv6): internet protocol version 6

      metric(metric): multi exit discriminator

      igp (igp): interior gateway protocol

      mtu (mtu) maximum transmission uint
     ";


  revision "2015-03-09" {
    description "initial revision";
    reference "draft-ietf-i2rs-rib-info-model-06";
  }

  container nexthop-capacity{
    leaf support-tunnel{
      type boolean;
    }
    leaf support-chains{
      type boolean;
    }
    leaf support-list-of-list{
      type boolean;
    }
    leaf support-replication{
      type boolean;
    }
    leaf support-weighted{
      type boolean;
    }
    leaf support-protection{
      type boolean;
    }
    leaf lookup-limit{
      type uint8;
    }
  }
```

```
  container nexthop-tunnel-encap-capacity{
    leaf support-ipv4{
      type boolean;
    }
    leaf support-ipv6{
      type boolean;
    }
    leaf support-mpls{
      type boolean;
    }
    leaf support-gre{
      type boolean;
    }
    leaf support-vxlan{
      type boolean;
    }
    leaf support-nvgre{
      type boolean;
    }
  }

  container routing-instance {
    description
      "Configuration of a 'i2rs' pseudo-protocol instance
       consists of a list of routes.";
    leaf instance-name {
      description
        "A routing instance is identified by its name,
        INSTANCE_name.  This MUST be unique across all routing
        instances in a given network device.";
      type string ;
      mandatory true;
    }
    list interface-list {
      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.";
      key "name";
      leaf name {
        type if:interface-ref;
         description
         "A reference to The name of a configured network layer
         interface.";
      }
```

```
    }
    uses rt:router-id ;
    list rib-list {
      description
        "This is the list of RIBs associated with this routing
        instance.  Each routing instance can have multiple RIBs to
        represent routes of different types.";
      key "rib-name";
      leaf rib-name {
        description
         "A reference to The name of a rib.";
       type string;
        mandatory true;
      }
      leaf rib-family {
        type rib-family-def;
        mandatory true;
      }
      leaf enable-ip-rpf-check {
        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.";
        type boolean;
      }
      list route-list{
        key "route-index";
        uses route;
      }
    }
  }

  grouping route-prefix{
    description
      "The common attributes used for all routes";
    leaf route-index {
      type uint64 ;
      mandatory true;
    }
    leaf route-type {
      type route-type-def ;
      mandatory true;
    }
    container match {
      choice rib-route-type {
        case ipv4 {
```

```
          description
            "Match on destination IP address in the IPv4 header";
          container ipv4{
            leaf ipv4-route-type {
              type ip-route-type-def ;
              mandatory true;
            }
            choice ip-route-type {

              case destination-ipv4-address {
                leaf destination-ipv4-prefix {
                  type inet:ipv4-prefix;
                  mandatory true;
                }
              }
              case source-ipv4-address {
                leaf source-ipv4-prefix {
                  type inet:ipv4-prefix;
                  mandatory true;
                }
              }
              case destination-source-ipv4-address {
                container destination-source-ipv4-address {
                  leaf destination-ipv4-prefix {
                    type inet:ipv4-prefix;
                    mandatory true;
                  }
                  leaf source-ipv4-prefix {
                    type inet:ipv4-prefix;
                    mandatory true;
                  }
                }
              }
            }
          }
        }
        case ipv6 {
          description
            "Match on destination IP address in the IPv6 header";
          container ipv6{
            leaf ipv6-route-type {
              type ip-route-type-def ;
              mandatory true;
            }
            choice ip-route-type {
              case destination-ipv6-address {
                leaf destination-ipv6-prefix {
                  type inet:ipv6-prefix;
```

```
                mandatory true;
              }
            }
            case source-ipv6-address {
              leaf source-ipv6-prefix {
                type inet:ipv6-prefix;
                mandatory true;
              }
            }
            case destination-source-ipv6-address {
              container destination-source-ipv6-address {
                leaf destination-ipv6-prefix {
                  type inet:ipv6-prefix;
                  mandatory true;
                }
                leaf source-ipv6-prefix {
                  type inet:ipv6-prefix;
                  mandatory true;
                }
              }
            }
          }
        }
        case mpls-route {
          description
            "Match on a MPLS label at the top of the MPLS label stack";
          leaf mpls-label {
            type uint32 ;
            mandatory true;
          }
        }

        case mac-route {
          description
            "Match on MAC destination addresses in the ethernet header";
          leaf mac-address {
            type uint32 ;
            mandatory true;
          }
        }
        case interface-route {
          description
            "Match on incoming interface of the packet";
          leaf interface-identifier {
            type if:interface-ref;
            mandatory true;
          }
```

```
        }
      }
    }
  }

  grouping route{
    description
      "The common attributes usesd for all routes";
    uses route-prefix;
    container nexthop{
      uses nexthop;
    }
    container route-statistic{
      leaf route-state {
        type route-state-def ;
        config false;
      }
      leaf route-installed-state {
        type route-installed-state-def ;
        config false;
      }
      leaf route-reason {
        type route-reason-def ;
        config false;
      }
    }
    container route-attributes{
      uses route-attributes;
    }
    container route-vendor-attributes{
      uses route-vendor-attributes;
    }
  }

  typedef nexthop-ref {
    type leafref {
      path  "/i2rs-rib:routing-instance/i2rs-rib:rib-list" +
            "/i2rs-rib:route-list/i2rs-rib:nexthop/i2rs-rib:nexthop-id";

    }
  }

  grouping nexthop {
    leaf nexthop-id {
      mandatory true;
      type uint32;
    }
    choice nexthop-type {
```

```
      case nexthop-base {
        container nexthop-base {
          list nexthop-chain {
            key "nexthop-chain-id";
            uses nexthop-chain-member;
          }
        }
      }

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

      case nexthop-load-balance {
        container nexthop-lb {
          list nexthop-lbs {
            key "nexthop-lbs-id";
            leaf nexthop-lbs-id {
              mandatory true;
              type uint32;
            }
            leaf nhop-lb-weight {
              mandatory true;
              type nhop-lb-weight-def;
            }
            leaf nexthop-lb-member {
              mandatory true;
```

```
              type nexthop-ref;
            }
          }
        }
      }

      case nexthop-replicate {
        container nexthop-replicate {
          list nexthop-replicates{
            key "nexthop-replicates-id";
            leaf nexthop-replicates-id {
              mandatory true;
              type uint32;
            }
            leaf nexthop-replicate {
              type nexthop-ref;
            }
          }
        }
      }
    }
  }

  grouping nexthop-chain-member {
    description
      "One Nexthop content for routes.";
    leaf nexthop-chain-id{
      type uint32;
      mandatory true;
    }
    choice nexthop-chain-type {
      case nexthop-chain-member-special {
        container nexthop-chain-member-special {
          leaf nexthop-chain-member-special{
            type special-nexthop-def;
          }
        }
      }

      case nexthop-chain-member-identifier{
        uses nexthop-chain-member-identifier;
      }

      case egress-interface-next-hop {
         description
           "Simple next-hop is specified as an outgoing interface,
            next-hop address or both.";
         leaf outgoing-interface {
```

```
           type if:interface-ref;
           mandatory true;
           description
             "Name of The outgoing interface.";
         }
       }

       case ipv4-address-next-hop {
         leaf next-hop-ipv4-address {
           type inet:ipv4-address;
           mandatory true;
           description
             "Ipv4 address of The next-hop.";
         }
       }

       case ipv6-address-next-hop {
         leaf next-hop-ipv6-address {
           type inet:ipv6-address;
           mandatory true;
           description
             "Ipv6 address of The next-hop.";
         }
       }

       case egress-interface-ipv4-next-hop {
         container next-hop-egress-interface-ipv4-address{
           leaf outgoing-interface {
             type if:interface-ref;
             mandatory true;
             description
               "Name of The outgoing interface.";
           }
           leaf next-hop-egress-ipv4-address {
             type inet:ipv4-address;
             mandatory true;
             description
               "Ipv4 address of The next-hop.";
           }
           description
             "Egress-interface and ip address: This can be usesd
              in cases e.g.where The ip address is a link-local
              address.";
         }
       }

       case egress-interface-ipv6-next-hop {
         container next-hop-egress-interface-ipv6-address{
```

```
          leaf outgoing-interface {
            type if:interface-ref;
            mandatory true;
            description
              "Name of The outgoing interface.";
          }
          leaf next-hop-egress-ipv6-address {
            type inet:ipv4-address;
            mandatory true;
            description
              "Ipv4 address of The next-hop.";
          }
          description
            "Egress-interface and ip address: This can be usesd
             in cases e.g.where The ip address is a link-local
             address.";
        }
      }

      case egress-interface-mac-next-hop {
        container next-hop-egress-interface-mac-address{
          leaf outgoing-interface {
            type if:interface-ref;
            mandatory true;
            description
              "Name of The outgoing interface.";
          }
          leaf ieee-mac-address {
            type uint32;
            mandatory true;
            description
              "Name of The mac-address.";
          }
          description
            "Egress-interface and ip address: This can be usesd
             in cases e.g.where The ip address is a link-local
             address.";
        }
      }

      case tunnel-encap-next-hop {
        container tunnel-encap {
          uses tunnel-encap;
            leaf outgoing-interface {
              type string;
          }
          description
            "This can be an encap representing an ip tunnel or
```

```
            mpls tunnel or others as defined in this document.
            An optional egress interface can be specified to
            indicate which interface to send The packet out on.
            The egress interface is usesful when the network
            device contains eThernet interfaces and one needs
            to perform address resolution for The ip packet.";
        }
      }

      case logical-tunnel-next-hop {
        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 {
        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.";
        }
      }
    }
  }

  grouping nexthop-chain-member-identifier{
    choice nexthop-identifier-type{
      case nexthop-chain-name {
        leaf nexthop-chain-name {
          type string;
          mandatory true;
        }
      }
      case nexthop-chain-id {
        leaf nexthop-chain-id {
          type uint32;
          mandatory true;
        }
      }
    }
  }
```

```
  grouping route-vendor-attributes{
  }

  grouping logical-tunnel{
    leaf tunnel-type {
      type tunnel-type-def ;
      mandatory true;
    }
    leaf tunnel-name {
      type string ;
      mandatory true;
    }
  }

  grouping ipv4-header{
    leaf source-ipv4-address {
      type inet:ipv4-address;
      mandatory true;
    }
    leaf destination-ipv4-address {
      type inet:ipv4-address;
      mandatory true;
    }
    leaf protocol {
      type uint8;
      mandatory true;
    }
    leaf ttl {
      type uint8;
    }
    leaf dscp {
      type uint8;
    }
  }

  grouping ipv6-header{
    leaf source-ipv6-address {
      type inet:ipv6-address;
      mandatory true;
    }
    leaf destination-ipv6-address {
      type inet:ipv6-address;
      mandatory true;
    }
    leaf next-header {
      type uint8;
      mandatory true;
    }
```

```
    leaf traffic-class {
      type uint8;
    }
    leaf flow-label {
      type uint16;
    }
    leaf hop-limit {
      type uint8;
    }
  }

  grouping nvgre-header{
    choice nvgre-type {
      description
        "nvgre-header.";
      case ipv4 {
        uses ipv4-header;
      }
      case ipv6 {
        uses ipv6-header;
      }
    }
    leaf virtual-subnet-id {
      type uint32;
      mandatory true;
    }
    leaf flow-id {
      type uint16;
    }
  }

  grouping vxlan-header{
    choice vxlan-type {
      description
        "vxlan-header.";
      case ipv4 {
        uses ipv4-header;
      }
      case ipv6 {
        uses ipv6-header;
      }
    }
    leaf vxlan-identifier {
      type uint32;
    }
  }

  grouping gre-header{
```

```
    leaf gre-ip-destination {
      type inet:ipv4-address;
      mandatory true;
    }
    leaf gre-protocol-type {
      type inet:ipv4-address;
      mandatory true;
    }
    leaf gre-key {
      type uint64;
    }
  }

  grouping mpls-header{
    choice mpls-action-type {
      description
        "mpls-header.";
      case mpls-push {
        leaf mpls-push {
          type boolean;
          mandatory true;
        }
        leaf mpls-label {
          type uint32;
          mandatory true;
        }
        leaf s-bit {
          type boolean;
        }
        leaf tos-value {
          type uint8;
        }
        leaf ttl-value {
          type uint8;
        }
          }
      case mpls-pop {
        leaf mpls-pop {
          type boolean;
          mandatory true;
        }
        leaf ttl-action {
          type uint8;
        }
      }
    }
  }
```

```
  grouping tunnel-encap{
    choice tunnel-type {
      description
        "options for next-hops.";
      case ipv4 {
        uses ipv4-header;
      }
      case ipv6 {
        uses ipv6-header;
      }
      case mpls {
        uses mpls-header;
      }
      case gre {
        uses gre-header;
      }
      case nvgre {
        uses nvgre-header;
      }
    }
  }

  grouping route-attributes{
    leaf route-preference {
      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 iss also known as administrative-distance.
         The lower the value, the higher the preference.";
        type uint32 ;
      mandatory true;
    }
    leaf local-only {
      type boolean ;
      mandatory true;
    }
    container address-family-route-attributes{
      choice route-type {
        case ip-route-attributes {
        }
        case mpls-route-attributes {
        }
        case eThernet-route-attributes {
        }
      }
    }
```

```
  }
  typedef nexthop-preference-def {
    description
      "Nexthop-preference is used for protection schemes.
       It is an integer value between 1 and 99.  A lower
       value indicates higher preference.  To download a
       primary/standby/tertiary group to the FIB, the
       nexthops that are resolved and have two highest
       preferences are selected.";
    type uint8 {
      range "1..99";
    }
  }
  typedef nhop-lb-weight-def {
    description
      "Nhop-lb-weight is a number between 1 and 99.";
    type uint8 {
      range "1..99";
    }
  }

  identity mpls-action {
    description
      "The mpls-action. ";
  }

  identity push {
    base "mpls-action";
  }

  identity pop {
    base "mpls-action";
  }

  identity swap {
    base "mpls-action";
  }

  typedef mpls-action-def {
    type identityref {
      base "mpls-action";
    }
  }

  identity special-nexthop {
    description
      "special-nexthop. ";
  }
```

```
  identity discard {
    base "special-nexthop";
  }

  identity discard-with-error {
    base "special-nexthop";
  }

  identity receive {
    base "special-nexthop";
  }

  identity cos-value {
    base "special-nexthop";
  }

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

  identity ip-route-type {
    description
      "The ip route type. ";
  }

  identity src {
    base "ip-route-type";
  }

  identity dest {
    base "ip-route-type";
  }

  identity dest-src {
    base "ip-route-type";
  }

  typedef ip-route-type-def {
    type identityref {
      base "ip-route-type";
    }
  }

  identity rib-family {
    description
      "The rib-family.";
```

```
  }

  identity ipv4-rib-family {
    base "rib-family";
  }

  identity ipv6-rib-family {
    base "rib-family";
  }

  identity mpls-rib-family {
    base "rib-family";
  }

  identity ieee-mac-rib-family {
    base "rib-family";
  }

  typedef rib-family-def {
    type identityref {
      base "rib-family";
    }
  }

  identity route-type {
    description "The route type. ";
  }

  identity ipv4-route {
    base "route-type";
  }

  identity ipv6-route {
    base "route-type";
  }

  identity mpls-route {
    base "route-type";
  }

  identity ieee-mac {
    base "route-type";
  }

  identity interface {
    base "route-type";
  }
```

```
  typedef route-type-def {
    type identityref {
      base "route-type";
    }
  }

  identity tunnel-type {
    description
      "The tunnel type.";
  }

  identity ipv4-tunnel {
    base "tunnel-type";
    description
      "IPv4 tunnel type";
  }

  identity ipv6-tunnel {
    base "tunnel-type";
    description
      "IPv6 Tunnel type";
  }

  identity mpls-tunnel {
    base "tunnel-type";
    description
      "MPLS tunnel type";
  }

  identity gre-tunnel {
    base "tunnel-type";
    description
      "GRE tunnel type";
  }

  identity vxlan-tunnel {
    base "tunnel-type";
    description
      "VxLAN tunnel type";
  }

  identity nvgre-tunnel {
    base "tunnel-type";
    description
      "NVGRE tunnel type";
  }

  typedef tunnel-type-def {
```

```
    type identityref {
      base "tunnel-type";
    }
  }

  identity route-state {
    description
      "The route state.";
  }

  identity active {
    base "route-state";
  }

  identity inactive {
    base "route-state";
  }

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

  identity nexthop-state {
    description
      "The nexthop state.";
  }

  identity resolved {
    base "nexthop-state";
  }

  identity unresolved {
    base "nexthop-state";
  }

  typedef nexthop-state-def {
    type identityref {
      base "nexthop-state";
    }
  }

  identity route-installed-state {
    description
      "The route installed state. ";
  }
```

```
  identity uninstalled {
    base "route-installed-state";
  }

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

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

  identity route-reason {
    description
      "The reason of invalid route. ";
  }

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

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

  identity higher-metric {
    base "route-reason";
    description
      "Higher metric";
  }

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


  notification nexthop-resolution-status-change {
    description
        "Nexthop resolution status (resolved/unresolved)
         notification.";
```

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

  notification route-change {
    description
        "Route change notification.";
    leaf instance-name {
      description
        "A routing instance is identified by its name,
        INSTANCE_name.  This MUST be unique across all
        routing instances in a given network device.";
      type string ;
      mandatory true;
    }
    leaf rib-name {
      description
       "A reference to The name of a rib.";
      type string;
      mandatory true;
    }
    leaf rib-family {
      type rib-family-def;
      mandatory true;
    }
    uses route-prefix;
    leaf route-installed-state {
      description
       "Indicates whether the route got installed in the FIB.";
      type route-installed-state-def;
      mandatory true;
    }
    leaf route-state {
      description
       "Indicates whether a route is fully resolved and
        is a candidate for selection.";
      type route-state-def;
      mandatory true;
    }
    leaf route-reason {
```

```
      description
       "Need to be added.";
      type route-reason-def;
      mandatory true;
    }
  }
}
//   </code ends>
```

4. IANA Considerations

This draft includes no request to IANA.

5. Security Considerations

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

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

6.2. Informative References

[I-D.ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-09 (work in progress), March 2015.

[I-D.ietf-i2rs-rib-info-model]
Bahadur, N., Folkes, R., Kini, S., and J. Medved, "Routing Information Base Info Model", draft-ietf-i2rs-rib-info-model-06 (work in progress), March 2015.

[I-D.ietf-i2rs-usecase-reqs-summary]
Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", draft-ietf-i2rs-usecase-reqs-summary-00 (work in progress), November 2014.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

Authors' Addresses

Lixing Wang
Huawei

Email: wang_little_star@sina.com

Hariharan Ananthakrishnan
Packet Design

Email: hari@packetdesign.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Amit Dass
Ericsson
Torshamnsgatan 48.
Stockholm 16480
Sweden

Email: amit.dass@ericsson.com

Sriganesh Kini
Ericsson

Email: sriganesh.kini@ericsson.com

Nitin Bahadur
Bracket Computing

Email: nitin_bahadur@yahoo.com