Data model for RIB I2RS protocol
draft-wang-i2rs-rib-data-model-00

Abstract

Routing and routing functions in enterprise and carrier networks are typically performed by network devices (routers and switches) using a routing information base (RIB). Protocols and configuration collectively push data into RIB and the RIB manager installs state information into the hardware; for packet forwarding. This draft specifies a data model for the RIB in order to define and enable a standardized data model. Such a data model can be used to define an interface to the RIB by using an entity that may even be external to the network device. This interface can be used to support new use-cases being defined by the IETF I2RS WG.

This document introduces a yang data for I2RS RIB that aligns with the I2RS RIB use cases and 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 June 11, 2015.
1. Introduction

The Interface to the Routing System (I2RS) provides read and write access to the information and state within the routing process that exists inside the routing elements via protocol message exchange between an I2RS Client and an I2RS Agent associated with the routing system. The [I-D.ietf-i2rs-architecture] describes the basic interaction procedures of this exchange activity. One of the important functions of this message exchange is to get the I2RS client interact with one or more I2RS agents in order to collect information from the network routing systems.

Protocols and configuration collectively push data into RIB and the RIB manager installs state information into the hardware; for packet
forwarding. This draft specifies a standardized data model for the RIB. Such a data model can be used to define an interface to the RIB from an entity that may even be external to the network device. This interface can be used to support new use-cases being defined by the IETF I2RS WG.

2. Definitions and Acronyms

RIB: routing information base

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

NETCONF: The Network Configuration Protocol as defined in [RFC6536]

RESTCONF: The REST-like protocol that provides a programmatic interface over HTTP for accessing the data defined in YANG, using datastores defined in NETCONF Protocol [I-D.ietf-netconf-restconf] as defined in RBNF: Routing Backus-Naur Form [RFC5511].

3. Yang Top-level description

3.1. Capabilities

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 SHOULD provide a way to expose the nexthop chaining capability supported by a given network device.

The high-level yang for the next-hop-capacity and the nexthop-tunnel-encap-capacity:
3.2. routing-instance-list

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.

A routing instance MUST contain the following mandatory fields.

o INSTANCE_NAME: A routing instance is identified by its defined name

o rib-list: This is the list of RIBs associated with this routing instance. Each routing instance can have multiple RIBs to represent routes of different types. A route is essentially a match condition and an action following that match. The match condition specifies the kind of route (IPv4, MPLS, etc.) and the set of fields to match on.

A routing instance MAY contain the following optional fields.

o interface-list: This represents the list of interfaces associated with a particular routing instance.

o router-id: The router-id field identifies the network device in various control plane interactions with other network devices.

Top level yang:
3.3. Route

A route is essentially a match condition and an action following that match. The match condition specifies the kind of route (IPv4, MPLS, MAC.Interface) and the set of fields to match on. Each route MUST have associated with an identified ROUTE_PREFERENCE attributes and preferably it can have one or more optional route attributes, such as the route-vendor-attributes.

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

A nexthop represents an object value resulting from a route lookup. Nexthops can be Unicast, Tunnel nexthops, Replication lists, Weighted lists, Protection lists, Nexthop chains, Indirect nexthops, Special nexthops.

Top level yang:
3.4. 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.

Top level yang:

```
notifications:
  +++ n nexthop-resolution-status-change
    +++ ro nexthop-chain-identifier
      +++ ro (nexthop-identifier-type)?
      |   +++ ro nexthop-name    string
      |   +++ ro nexthop-id      uint32
      |   +++ ro nexthop* [nexthop-index]
      |     +++ ro nexthop-index  uint32
      |     +++ ro (next-hop-options)?
      |     |   +++ ro outgoing-interface string
      |     |   +++ ro next-hop-ipv4-address inet:ipv4-address
      |     |   +++ ro ipv4-rib-name? string
      |     |   +++ ro next-hop-ipv6-address inet:ipv6-address
      |     |   +++ ro ipv6-rib-name? string
      |     |   +++ ro next-hop-egress-interface-ipv4-address
      |     |     +++ ro outgoing-interface string
      |     |     +++ ro next-hop-egress-ipv4-address inet:ipv4-address
      |     |     +++ ro (egress-interface-ipv4-next-hop)
      |     |     +++ ro next-hop-egress-interface-ipv4-address
      |     |     |   +++ ro outgoing-interface string
      |     |     |   +++ ro next-hop-egress-ipv4-address inet:ipv4-address
      |     |     |     +++ ro (egress-interface-ipv6-next-hop)
      |     |     |     +++ ro next-hop-egress-interface-ipv6-address
      |     |     |     |   +++ ro outgoing-interface string
      |     |     |     |   +++ ro next-hop-egress-ipv6-address inet:ipv4-address
      |     |     |     |     +++ ro (egress-interface-mac-next-hop)
      |     |     |     |     +++ ro next-hop-egress-interface-mac-address
      |     |     |     |     |   +++ ro outgoing-interface string
      |     |     |     |     |   +++ ro ieee-mac-address uint32
      |     |     |     |     |     +++ ro (logical-tunnel-next-hop)
```
3.5. NextHops

A nexthop represents an object resulting from a route lookup.

A nexthop can be Special nexthop or a normal nexthop.

1) special-next-hops -for performing some specific well-defined functions, for example, discard, discard with error, or receive.

Top level yang :

```yang
+-ro source-ipv4-prefix inet:ipv4-prefix
+-ro ipv6
  +-ro ipv6-route-type ip-route-type-def
  +-ro (ip-route-type)?
    +-ro destination-ipv6-prefix inet:ipv6-prefix
    +-ro source-ipv6-prefix inet:ipv6-prefix
  +-ro (ipv6)
    +-ro (destination-ipv6-address)
      +-ro destination-ipv6-prefix inet:ipv6-prefix
      +-ro source-ipv6-prefix inet:ipv6-prefix
    +-ro (source-ipv6-address)
      +-ro destination-ipv6-prefix inet:ipv6-prefix
      +-ro source-ipv6-prefix inet:ipv6-prefix
    +-ro (destination-source-ipv6-address)
      +-ro destination-ipv6-prefix inet:ipv6-prefix
      +-ro source-ipv6-prefix inet:ipv6-prefix
  +-ro mpls-route
    +-ro mpls-label-in uint32
    +-ro mpls-action mpls-action-def
    +-ro mpls-label-out? uint32
  +-ro mac-route
    +-ro mac-address uint32
  +-ro interface-route
    +-ro interface-identifier uint32
    +-ro route-installed-state route-installed-state-def
    +-ro route-state route-state-def
    +-ro route-reason route-reason-def
```

2) normal next-hops

Normal nexthops can be a nexthop list member which include only one nexthop or a list of list. One nexthop can be:

- IP address: A route lookup on this IP address is done to determine the egress interface.

- egress-interface - pointing to an interface.
o logical-tunnel - pointing to a tunnel.

o tunnel-encap is used to specify multiple headers over a packet, before a packet is forwarded. Using Nexthop chains can implement chained headers, e.g. MPLS label over a GRE header.

o Indirect nexthops - pointing to a nexthop identifier.

Top level yang:

```yang
 +-:(normal-nexthop)
  +-:(one-nexthop-list-member)
    +-rw (nexthop-member-or-list-of-list)?
    +-:(nexthop-chain)
       +-rw nexthop-chain-identifier
       +-:(nexthop-identifier)
         +-:(nexthop-name)
         +-:(nexthop-id)
       +-rw nexthop* [nexthop-index]
       +-rw nexthop-index uint32
  +-rw (next-hop-options)?
    +-:(egress-interface-next-hop)
    +-:(ipv4-address-next-hop)
    +-:(ipv6-address-next-hop)
    +-:(egress-interface-ipv4-next-hop)
    +-:(egress-interface-ipv6-next-hop)
    +-:(egress-interface-mac-next-hop)
  +-:(logical-tunnel-next-hop)
```
4. Full Yang Top-level description

Below is the full RIB Yang top-level description for the configuration portion of I2RS configuration model. Additional information on the structure of the information model is described in [I-D.ietf-i2rs-rib-info-model].
module: i2rs-rib
  +-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
  +--rw routing-instance-list* [instance-name]
     |   +-rw instance-name         string
     |   |   +-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 {rib-route-type}?
           |   +-:(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-ipv4-prefix inet:ipv4-prefix
           |   |   |       |   |   +-rw source-ipv4-prefix inet:ipv4-prefix
           |   +--:(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

| +--:(egress-interface-ipv4-next-hop) |
| +--rw next-hop-egress-interface-ipv4-address  string |
| +--rw outgoing-interface                  string |
| +--rw next-hop-egress-ipv4-address        inet:ipv4-address |

| +--:(egress-interface-ipv6-next-hop) |
| +--rw next-hop-egress-interface-ipv6-address  string |
| +--rw outgoing-interface                  string |
| +--rw next-hop-egress-ipv6-address        inet:ipv4-address |

| +--:(egress-interface-mac-next-hop) |
| +--rw next-hop-egress-interface-mac-address  string |
| +--rw outgoing-interface                  string |
| +--rw ieee-mac-address                   uint32 |

| +--:(logical-tunnel-next-hop) |
| +--rw logical-tunnel |
| +--rw tunnel-type        tunnel-type-def |
| +--rw tunnel-name         string |

| +--:(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) |
++-:(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 (nextrhop-third-encap-or-not)?
      ++-:(nextrhop-third-encap)
        ++-rw nexthop-third-encap
          ++-:(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
          ++-:(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 (route-type)?
 | +--:(ip-route-attributes)
 | +--:(mpls-route-attributes)
 | +--:(eThernet-route-attributes)

notifications:
+---n nexthop-resolution-status-change
  +--ro nexthop-chain-identifier
    | +--ro (nexthop-identifier-type)?
    | | +--:(nexthop-name)
    | | | +--ro nexthop-name    string
    | +--:(nexthop-id)
    | +--ro nexthop-id      uint32
  +--ro nexthop* [nexthop-index]
    +--ro nexthop-index                             uint32
    +--ro (next-hop-options)?
      +--:(nexthop-identifier-next-hop)
      | +--ro (nexthop-identifier-type)?
      | | +--:(nexthop-name)
      | | | +--ro nexthop-name    string
      | +--:(nexthop-id)
      | +--ro nexthop-id      uint32
      +--:(egress-interface-next-hop)
      | +--ro outgoing-interface string
      +--:(ipv4-address-next-hop)
      | +--ro next-hop-ipv4-address                     inet:ipv4-address
      | +--ro ipv4-rib-name?                            string
      +--:(ipv6-address-next-hop)
      | +--ro next-hop-ipv6-address                     inet:ipv6-address
      | +--ro ipv6-rib-name?                            string
      +--:(egress-interface-ipv4-next-hop)
      | +--ro next-hop-egress-interface-ipv4-address   inet:ipv4-address
      | +--ro outgoing-interface                      string
      +--:(egress-interface-ipv6-next-hop)
      | +--ro next-hop-egress-interface-ipv6-address   inet:ipv4-address
      | +--ro outgoing-interface                      string
      +--:(egress-interface-mac-next-hop)
      | +--ro next-hop-egress-interface-mac-address    inet:ipv4-address
      | +--ro ieee-mac-address                        uint32
      +--:(logical-tunnel-next-hop)
      | +--ro logical-tunnel
      | | +--ro tunnel-type    tunnel-type-def
      | | +--ro tunnel-name    string
      +--:(tunnel-encap-next-hop)
      | +--ro tunnel-encap
      | | +--ro (tunnel-type)?
++--ro {tunnel-type}?
  +--ro source-ipv4-address    inet:ipv4-address
  +--ro destination-ipv4-address    inet:ipv4-address
  +--ro protocol    uint8
  +--ro ttl7?    uint8
  +--ro dscp?    uint8

++--ro source-ipv6-address    inet:ipv6-address
  +--ro destination-ipv6-address    inet:ipv6-address
  +--ro next-header    uint8
  +--ro traffic-class?    uint8
  +--ro flow-label?    uint16
  +--ro hop-limit?    uint8

++--ro {mpls-action-type}?
  +--ro mpls-push    boolean
  +--ro mpls-label    uint32
  +--ro s-bit?    boolean
  +--ro tos-value?    uint8
  +--ro ttl-value?    uint8

++--ro gre-ip-destination    inet:ipv4-address
++--ro gre-protocol-type    inet:ipv4-address
++--ro gre-key?    uint64

++--ro {nvGRE-type}?
  +--ro source-ipv4-address    inet:ipv4-address
  +--ro destination-ipv4-address    inet:ipv4-address
  +--ro protocol    uint8
  +--ro ttl7?    uint8
  +--ro dscp?    uint8

++--ro source-ipv6-address    inet:ipv6-address
  +--ro destination-ipv6-address    inet:ipv6-address
  +--ro next-header    uint8
  +--ro traffic-class?    uint8
  +--ro flow-label?    uint16
  +--ro hop-limit?    uint8
  +--ro virtual-subnet-id    uint32
  +--ro flow-id?    uint16

++--ro {nexthop-third-encap-or-not}?
  +--ro {nexthop-third-encap}
++--ro nexthop-third-encap
  ++--ro (tunnel-type)?
    ++--:(ipv4)
      ++--ro source-ipv4-address inet:ipv4-address
      ++--ro destination-ipv4-address inet:ipv4-address
      ++--ro protocol uint8
      ++--ro ttl? uint8
      ++--ro dscp? uint8
    ++--:(ipv6)
      ++--ro source-ipv6-address inet:ipv6-address
      ++--ro destination-ipv6-address inet:ipv6-address
      ++--ro next-header uint8
      ++--ro traffic-class? uint8
      ++--ro flow-label? uint16
      ++--ro hop-limit? uint8
    ++--(mpls)
      ++--ro (mpls-action-type)?
        ++--:(mpls-push)
          ++--ro mpls-push boolean
          ++--ro mpls-label uint32
          ++--ro s-bit? boolean
          ++--ro tos-value? uint8
          ++--ro ttl-value? uint8
        ++--:(mpls-pop)
          ++--ro mpls-pop boolean
          ++--ro ttl-action? uint8
      ++--:(gre)
        ++--ro gre-ip-destination inet:ipv4-address
        ++--ro gre-protocol-type inet:ipv4-address
        ++--ro gre-key? uint64
    ++--(nvgre)
      ++--ro (nvgre-type)?
        ++--:(ipv4)
          ++--ro source-ipv4-address inet:ipv4-address
          ++--ro destination-ipv4-address inet:ipv4-address
          ++--ro protocol uint8
          ++--ro ttl? uint8
          ++--ro dscp? uint8
        ++--:(ipv6)
          ++--ro source-ipv6-address inet:ipv6-address
          ++--ro destination-ipv6-address inet:ipv6-address
          ++--ro next-header uint8
          ++--ro traffic-class? uint8
          ++--ro flow-label? uint16
          ++--ro hop-limit? uint8
          ++--ro virtual-subnet-id uint32
          ++--ro flow-id? uint16
        ++--ro (nexthop-forth-encap-or-not)?
| +--ro flow-id?             uint16 |
| +--ro outgoing-interface?         string |
| +--ro nexthop-state          nexthop-state-def |

--- 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 (rib-route-type)?
    +--ro ipv4
        +--ro ipv4-route-type          ip-route-type-def
        +--ro (ip-route-type)?
            +--:(destination-ipv4-address)
            | +--ro destination-ipv4-prefix      inet:ipv4-prefix
            +--:(source-ipv4-address)
            | +--ro source-ipv4-prefix           inet:ipv4-prefix
            +--:(destination-source-ipv4-address)
                +--ro destination-ipv4-prefix    inet:ipv4-prefix
                +--ro source-ipv4-prefix         inet:ipv4-prefix
    +--ro ipv6
        +--ro ipv6-route-type          ip-route-type-def
        +--ro (ip-route-type)?
            +--:(destination-ipv6-address)
            | +--ro destination-ipv6-prefix      inet:ipv6-prefix
            +--:(source-ipv6-address)
            | +--ro source-ipv6-prefix           inet:ipv6-prefix
            +--:(destination-source-ipv6-address)
                +--ro destination-ipv6-prefix    inet:ipv6-prefix
                +--ro source-ipv6-prefix         inet:ipv6-prefix
    +--ro (mpls-route)
        | +--ro mpls-label-in             uint32
        | +--ro mpls-action               mpls-action-def
        | +--ro mpls-label-out?           uint32
        +--ro (mac-route)
            | +--ro mac-address              uint32
            +--ro (interface-route)
                | +--ro interface-identifier      uint32
                +--ro route-installed-state    route-installed-state-def
                +--ro route-state              route-state-def
                +--ro route-reason             route-reason-def
5. RIB Yang description

//<code begins> file "i2rs isis@2014-08-31.yang"

module i2rs-rib {
    namespace "urn:huawei: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
        "Huawei technologies co., ltd. ";

    contact
        "email: wanglixing@huawei.com
            email: shares@ndzh.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 unit 
        ";
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;
}
}

list routing-instance-list{
  description
  "configuration of a 'i2rs' pseudo-protocol instance
  consists of a list of routes.";
  key "instance-name";
  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;
    }
    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;
                    }
                }
            }
        }
    }
}
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;
            }
        }
    }
case mpls-route {
    description
    "Match on a MPLS label at the top of the MPLS label stack";
    leaf mpls-label-in {
        type uint32 ;
        mandatory true;
    }
    leaf mpls-action {
        type mpls-action-def ;
        mandatory true;
    }
    leaf mpls-label-out {
        type uint32 ;
    }
}

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 uint32 ;
        mandatory true;
    }
}
}

grouping route {
    description
    "The common attributes used for all routes";
    uses route-prefix;
    list nexthop-list{
description
"One can create a replication list for replication traffic to multiple
destinations. The destinations, in turn, could be complex nexthops
in themselves - at a level supported by the network device. Point to
multipoint and broadcast are examples that involve replication";
key "nexthop-list-index";
uses nexthop-list;
}
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;
}
uses route-attributes;
uses route-vendor-attributes;
}

grouping nexthop-list{
  leaf nexthop-list-index{
    type uint32;
  }
  choice nexthop-list-type{
    case special-nexthop {
      leaf special-nexthop{
        type special-nexthop-def;
      }
    }
    case normal-nexthop {
      choice nexthop-member-or-list-of-list{
        case one-nexthop-list-member {
          uses nexthop-list-member;
        }
        case nexthop-list-of-list {
          list nexthop-list-member{
            key "nexthop-list-member-index";
            leaf nexthop-list-index{
              description
              "Lists of lists is a complex construct. One example of usage of such
              a construct is to replicate traffic to multiple destinations, with
              high availability. In other words, for each destination you have a
              primary and backup nexthop (replication list) to ensure there is no
            }
traffic drop in case of a failure. So the outer list is a protection
list and the inner lists are replication lists of primary/backup";

```plaintext
type uint32;

uses nexthop-list-member;
```
uses rt:next-hop-classifiers;

grouping nexthop-identifier{
  choice nexthop-identifier-type{
    case nexthop-name {
      leaf nexthop-name{
        type string;
        mandatory true;
      }
    }
    case nexthop-id {
      leaf nexthop-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 "vxlan-header.";
  case ipv4 {
    uses ipv4-header;
  case...
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 nexthop {
    description
        "One Nexthop content for routes."
    leaf nexthop-index {
        type uint32;
        mandatory true;
    }
    choice next-hop-options {
        case nexthop-identifier-next-hop{
            uses nexthop-identifier;
        }
        case egress-interface-next-hop {
            description
                "simple next-hop is specified as an outgoing interface, 
                next-hop address or both. 
                address-family-specific modules are expected to provide 
                'next-hop-address' leaf via augmentation."
            leaf outgoing-interface {
                type string;
                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."
            }
            leaf ipv4-rib-name {
                type string;
                description

        case gre {
            uses gre-header;
        }
        case nvgre {
            uses nvgre-header;
        }
    }
    }
    }
}

    description
        "One Nexthop content for routes."
    leaf nexthop-index {
        type uint32;
        mandatory true;
    }
    choice next-hop-options {
        case nexthop-identifier-next-hop{
            uses nexthop-identifier;
        }
        case egress-interface-next-hop {
            description
                "simple next-hop is specified as an outgoing interface, 
                next-hop address or both. 
                address-family-specific modules are expected to provide 
                'next-hop-address' leaf via augmentation."
            leaf outgoing-interface {
                type string;
                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."
            }
            leaf ipv4-rib-name {
                type string;
                description

        case gre {
            uses gre-header;
        }
        case nvgre {
            uses nvgre-header;
        }
    }
    }
    }
}

    description
        "One Nexthop content for routes."
    leaf nexthop-index {
        type uint32;
        mandatory true;
    }
    choice next-hop-options {
        case nexthop-identifier-next-hop{
            uses nexthop-identifier;
        }
        case egress-interface-next-hop {
            description
                "simple next-hop is specified as an outgoing interface, 
                next-hop address or both. 
                address-family-specific modules are expected to provide 
                'next-hop-address' leaf via augmentation."
            leaf outgoing-interface {
                type string;
                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."
            }
            leaf ipv4-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 ipv6-address-next-hop {
  leaf next-hop-ipv6-address {
    type inet:ipv6-address;
    mandatory true;
    description "Ipv6 address of the next-hop."
  }
  leaf ipv6-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 egress-interface-ipv4-next-hop {
  container next-hop-egress-interface-ipv4-address{
    leaf outgoing-interface {
      type string;
      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 used 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 string;
      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 used 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 string;
      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 used in cases e.g. where the ip address is a link-local address.";
  }
}

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 tunnel-encap-next-hop {
  container tunnel-encap {
    uses tunnel-encap;
    choice nexthop-second-encap-or-not{
      case nexthop-second-encap{
        container nexthop-second-encap{
          description "the two encapsulating nexthop. One example is a Pseudowire - which is MPLS over some transport (MPLS or GRE for instance). Another example is VxLAN over IP.";
          uses tunnel-encap;
          choice nexthop-third-encap-or-not{
case nexthop-third-encap {
  container nexthop-third-encap{
    description
    "the three encapsulating nexthop. One example is Option A - L3VPN OVER MPLS tunnel and MPLS over TE tunnel."
    uses tunnel-encap;
  }
}

case nexthop-fourth-encap-or-not{
  case nexthop-fourth-encap {
    container nexthop-fourth-encap{
      description
      "the four encapsulating nexthop. One example is Option C - which L3VPN OVER BGP-LSP and over MPLS tunnel, and MPLS over TE tunnel."
      uses tunnel-encap;
    }
    choose nexthop-fifth-encap-or-not{
      case nexthop-fifth-encap {
        container nexthop-fifth-encap{
          description
          "the five encapsulating nexthop. One example is Option C - which L3VPN OVER BGP-LSP and over MPLS tunnel, and MPLS over TE tunnel, the inner TE tunnel is FRR."
          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 useful when the network device contains ethernet interfaces and one needs to perform address resolution for the ip packet."
}

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 is 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 mpls-action-def {
  type enumeration {
    enum "push";
    enum "pop";
    enum "swap";
  }
}

typedef special-nextho-def {
  type enumeration {
    enum "discard";
    enum "discard-with-error";
    enum "receive";
    enum "cos-value";
  }
}

typedef ip-route-type-def {
  type enumeration {
    enum "src";
  }
enum "dest";
enum "dest-src";
}
}
typedef rib-family-def {
  type enumeration {
    enum "ipv4-rib-family";
    enum "ipv6-rib-family";
    enum "mpls-rib-family";
    enum "ieee-mac-rib-family";
  }
}
}
typedef route-type-def {
  type enumeration {
    enum "ipv4";
    enum "ipv6";
    enum "mpls";
    enum "ieee-mac";
    enum "interface";
  }
}
}
typedef tunnel-type-def {
  type enumeration {
    enum "ipv4";
    enum "ipv6";
    enum "mpls";
    enum "gre";
    enum "vxlan";
    enum "nvgre";
  }
}
}
typedef special-nexthop-def {
  type enumeration {
    enum "discard";
    enum "discard-with-error";
    enum "receive";
    enum "cos-value";
  }
}
}
typedef route-state-def {
  type enumeration {
    enum "active";
    enum "inactive";
  }
}
typedef nexthop-state-def {
    type enumeration {
        enum "resolved";
        enum "unresolved";
    }
}

typedef route-installed-state-def {
    type enumeration {
        enum "Installed";
        enum "uninstalled";
    }
}

typedef route-reason-def {
    type enumeration {
        enum "low preference";
        enum "unresolved nexthop";
        enum "higher metric";
    }
}

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

6. IANA Considerations

This draft includes no request to IANA.

7. Security Considerations

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

8.1. Informative References


8.2. Normative References


Authors' Addresses

Lixing Wang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: wanglixing@huawei.com

Amit Dass
Ericsson
Torshamngatan 48.
Stockholm 16480
Sweden

Email: amit.dass@ericsson.com