Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 16, 2018

X. Liu
Jabil
P. Sarda
Ericsson
V. Choudhary
Individual

December 13, 2017

A YANG Data Model for Routing Information Protocol (RIP)
draft-ietf-rtgwg-yang-rip-07

Abstract

This document describes a data model for the Routing Information Protocol (RIP). Both RIP version 2 and RIPng are covered.

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 16, 2018.

Copyright Notice

Table of Contents

1. Introduction

This document introduces a YANG [RFC7950] data model for the Routing Information Protocol (RIP) [RFC2453][RFC2080]. RIP was designed to work as an Interior Gateway Protocol (IGP) in moderate-size Autonomous Systems (AS).

This YANG model supports both RIP version 2 and RIPng. RIP version 2 (defined in [RFC2453]) supports IPv4. RIPng (defined in [RFC2080]) supports IPv6.

1.1. Terminology

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 [RFC2119].

The following terms are defined in [RFC7950] and are not redefined here:

- augment

- data model

- data node

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.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, 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.

1.3. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
if	ietf-interfaces	[I-D.bjorklund-netmod-rfc7223bis]
ip	ietf-ip	[I-D.bjorklund-netmod-rfc7277bis]
rt	ietf-routing	[I-D.acee-netmod-rfc8022bis]
bfd-types	ietf-bfd-types	[I-D.ietf-bfd-yang]
isis	ietf-isis	[I-D.ietf-isis-yang-isis-cfg]
key-chain	ietf-key-chain	[RFC8177]
ospf	ietf-ospf	[I-D.ietf-ospf-yang]

Table 1: Prefixes and Corresponding YANG Modules

2. Design of the Data Model

2.1. Scope of the Model

The model covers RIP version 2 [RFC2453] and RIPng [RFC2080] protocols. The model is designed to be implemented on a device where RIP version 2 or RIPng is implemented, and can be used to:

- Configure the RIP version 2 or RIPng protocol.
- Manage the protocol operational behaviors.
- Retrieve the protocol operational status.

The capabilities describe in [RFC1724] are covered.

2.2. Relation with Core Routing Framework

This model augments the core routing data model "ietf-routing" specified in [I-D.acee-netmod-rfc8022bis].

```
   +--rw routing
      +--rw router-id?
      +--rw control-plane-protocols
      |  +--rw control-plane-protocol* [type name]
      |     +--rw type
      |     +--rw name
      |     +--rw rip     <= Augmented by this Model
                  ...
```

The "rip" container instantiates a RIP protocol entity that supports RIP version 2 or RIPng. Depending on the implementation of "ietf-routing", a RIP instance MAY belong to a logical router or network instance.

2.3. Protocol Configuration

The model structure for the protocol configuration is as shown below:

```
augment /rt:routing/rt:control-plane-protocols/
rt:control-plane-protocol:
   +--rw rip
      +--rw <per instance configuration>
      +--rw interface* [interface]
         +--rw interface                  if:interface-ref
         +--rw <per interface configuration>
         +--rw neighbors {explicit-neighbors}?
         |  +--rw neighbor* [address]
         |     +--rw address    inet:ip-address
         |     +--rw <per neighbor configuration>
```

The model allows to configure the following protocol entities:

- Protocol instance (RIP version 2 or RIPng)
- Interface
- Neighbor

2.4. Protocol States

The model structure for the protocol states is as shown below:

```
augment /rt:routing/rt:control-plane-protocols/
rt:control-plane-protocol:
   +--rw rip
      +--ro <per instance operational states>
      +--rw interface* [interface]
      |  +--rw interface                  if:interface-ref
      |  +--ro <per instance operational states>
      |  +--ro statistics {interface-statistics}?
      |     +--ro <per instance statistics>
      +--ro ipv4
      |  +--ro neighbors
      |  |  +--ro neighbor* [ipv4-address]
      |  |     +--ro <per neighbor IPv4 operational states>
      |  +--ro routes
      |     +--ro route* [ipv4-prefix]
      |        +--ro <IPv4 RIP route states>
      +--ro ipv6
      |  +--ro neighbors
      |  |  +--ro neighbor* [ipv6-address]
      |  |     +--ro <per neighbor IPv6 operational states>
      |  +--ro routes
      |     +--ro route* [ipv6-prefix]
      |        +--ro ipv6-prefix                    inet:ipv6-prefix
      |        +--ro <IPv4 RIP route states>
      +--ro statistics {global-statistics}?
         +--ro <per instance statistics>
```

This model conforms to the Network Management Datastore Architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. The operational state data is combined with the associated configuration data in the same hierarchy [I-D.ietf-netmod-rfc6087bis]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

The model allows to retrieve protocol states at the following levels:

- Protocol instance (RIP version 2 or RIPng)
- Interface
- Neighbor
- Route

2.5. RPC Operations

This model defines one RPC "clear-rip-route" that can be used to clear RIP routes from the routing table.

2.6. Notifications

This model does not define RIP specific notifications. To enable notifications, the mechanism defined in [I-D.ietf-netconf-yang-push] and [I-D.ietf-netconf-rfc5277bis] can be used. This mechanism currently allows the user to:

- Subscribe notifications on a per client basis.
- Specify subtree filters or xpath filters so that only interested contents will be sent.
- Specify either periodic or on-demand notifications.

2.7. Optional Features

This model defines several features are beyond the basic RIP configuration and it is the responsibility of each vendor to decide whether to support a given feature on a device.

3. Tree Structure

This document defines the YANG module "ietf-rip", which has the following tree structure:

```
module: ietf-rip
  augment /rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol:
    +--rw rip
       +--rw originate-default-route
       |  +--rw enabled?        boolean
       |  +--rw route-policy?   route-policy-ref
       +--rw default-metric?              uint8
       +--rw distance?                    uint8
       +--rw triggered-update-threshold?  uint8
       +--rw maximum-paths?               uint8
       +--rw output-delay?                uint8
       +--rw distribute-list* [prefix-set-name direction]
       |  +--rw prefix-set-name    prefix-set-ref
       |  +--rw direction          enumeration
       |  +--rw if-name?           if:interface-ref
       +--rw redistribute
       |  +--rw bgp* [asn]
```

```
          |  |  +--rw asn             inet:as-number
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw cg-nat!
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw connected!
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw ipsec!
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw isis* [instance]
          |  |  +--rw instance        -> ../../../../..
   /rt:control-plane-protocol/name
          |  |  +--rw level?          enumeration
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw nat!
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw ospfv2* [instance]
          |  |  +--rw instance        -> ../../../../..
   /rt:control-plane-protocol/name
          |  |  +--rw route-type?     ospf:route-type
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw ospfv3* [instance]
          |  |  +--rw instance        -> ../../../../..
   /rt:control-plane-protocol/name
          |  |  +--rw route-type?     ospf:route-type
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw ripv2* [instance]
          |  |  +--rw instance        -> ../../../../..
   /rt:control-plane-protocol/name
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw ripng* [instance]
          |  |  +--rw instance        -> ../../../../..
   /rt:control-plane-protocol/name
          |  |  +--rw metric?         uint8
          |  |  +--rw route-policy?   route-policy-ref
          |  +--rw static!
          |     +--rw metric?         uint8
          |     +--rw route-policy?   route-policy-ref
          +--rw timers
          |  +--rw update-interval?     uint16
```

```
          |  +--rw invalid-interval?    uint16
          |  +--rw holddown-interval?   uint16
          |  +--rw flush-interval?      uint16
          +--rw interfaces
          |  +--rw interface* [interface]
          |     +--rw interface                  if:interface-ref
          |     +--rw authentication
          |     |  +--rw (auth-type-selection)?
          |     |     +--:(auth-key-chain)
          |     |     |  +--rw key-chain?
    key-chain:key-chain-ref
          |     |     +--:(auth-key)
          |     |        +--rw key?                string
          |     |        +--rw crypto-algorithm?   identityref
          |     +--rw bfd {bfd}?
          |     |  +--rw enable?                     boolean
          |     |  +--rw local-multiplier?           multiplier
          |     |  +--rw (interval-config-type)?
          |     |     +--:(tx-rx-intervals)
          |     |     |  +--rw desired-min-tx-interval     uint32
          |     |     |  +--rw required-min-rx-interval    uint32
          |     |     +--:(single-interval)
          |     |        +--rw min-interval                uint32
          |     +--rw cost?                      uint8
          |     +--rw neighbors {explicit-neighbors}?
          |     |  +--rw neighbor* [address]
          |     |     +--rw address    inet:ip-address
          |     +--rw no-listen?                 empty
          |     +--rw originate-default-route
          |     |  +--rw enabled?        boolean
          |     |  +--rw route-policy?   route-policy-ref
          |     +--rw passive?                   empty
          |     +--rw split-horizon?             enumeration
          |     +--rw summary-address
          |     |  +--rw address?   inet:ip-prefix
          |     |  +--rw metric?    uint8
          |     +--rw timers
          |     |  +--rw update-interval?     uint16
          |     |  +--rw invalid-interval?    uint16
          |     |  +--rw holddown-interval?   uint16
          |     |  +--rw flush-interval?      uint16
          |     +--ro oper-status?               enumeration
          |     +--ro next-full-update?          uint32
          |     +--ro valid-address?             boolean
          |     +--ro statistics {interface-statistics}?
          |        +--ro discontinuity-time?   yang:date-and-time
          |        +--ro bad-packets-rcvd?     yang:counter32
          |        +--ro bad-routes-rcvd?      yang:counter32
```

```
          |       +--ro updates-sent?         yang:counter32
          +--ro next-triggered-update?        uint32
          +--ro num-of-routes?                uint32
          +--ro ipv4
          |  +--ro neighbors
          |  |  +--ro neighbor* [ipv4-address]
          |  |     +--ro ipv4-address        inet:ipv4-address
          |  |     +--ro last-update?        yang:date-and-time
          |  |     +--ro bad-packets-rcvd?   yang:counter32
          |  |     +--ro bad-routes-rcvd?    yang:counter32
          |  +--ro routes
          |     +--ro route* [ipv4-prefix]
          |        +--ro ipv4-prefix
    inet:ipv4-prefix
          |        +--ro next-hop?
    inet:ipv4-address
          |        +--ro interface?
    if:interface-ref
          |        +--ro redistributed?                 boolean
          |        +--ro route-type?                    enumeration
          |        +--ro metric?                        uint8
          |        +--ro expire-time?                   uint16
          |        +--ro deleted?                       boolean
          |        +--ro holddown?                      boolean
          |        +--ro need-triggered-update?         boolean
          |        +--ro inactive?                      boolean
          |        +--ro flush-expire-before-holddown?  boolean
          +--ro ipv6
          |  +--ro neighbors
          |  |  +--ro neighbor* [ipv6-address]
          |  |     +--ro ipv6-address        inet:ipv6-address
          |  |     +--ro last-update?        yang:date-and-time
          |  |     +--ro bad-packets-rcvd?   yang:counter32
          |  |     +--ro bad-routes-rcvd?    yang:counter32
          |  +--ro routes
          |     +--ro route* [ipv6-prefix]
          |        +--ro ipv6-prefix
    inet:ipv6-prefix
          |        +--ro next-hop?
    inet:ipv6-address
          |        +--ro interface?
    if:interface-ref
          |        +--ro redistributed?                 boolean
          |        +--ro route-type?                    enumeration
          |        +--ro metric?                        uint8
          |        +--ro expire-time?                   uint16
          |        +--ro deleted?                       boolean
          |        +--ro holddown?                      boolean
```

```
          |       +--ro need-triggered-update?          boolean
          |       +--ro inactive?                       boolean
          |       +--ro flush-expire-before-holddown?   boolean
          +--ro statistics {global-statistics}?
             +--ro discontinuity-time?   yang:date-and-time
             +--ro requests-rcvd?        yang:counter32
             +--ro requests-sent?        yang:counter32
             +--ro responses-rcvd?       yang:counter32
             +--ro responses-sent?       yang:counter32

    rpcs:
      +---x clear-rip-route
         +---w input
            +---w rip-instance?   -> /rt:routing
  /control-plane-protocols/control-plane-protocol/name
```

4. YANG Module

```
<CODE BEGINS> file "ietf-rip@2017-12-05.yang"
module ietf-rip {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-rip";

  prefix rip;

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-yang-types {
    prefix "yang";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix "ip";
  }

  import ietf-routing {
    prefix "rt";
  }

  import ietf-key-chain {
```

```
      prefix "key-chain";
    }

    import ietf-bfd-types {
      prefix "bfd-types";
    }

    import ietf-ospf {
      prefix "ospf";
    }

    import ietf-isis {
      prefix "isis";
    }

    organization "IETF Routing Area Working Group (rtgwg)";

    contact
      "WG Web:   <http://tools.ietf.org/wg/rgtwg/>
       WG List:  <mailto:rgtwg@ietf.org>

       WG Chair: Jeff Tantsura
                 <mailto:jefftant.ietf@gmail.com>

       WG Chair: Chris Bowers
                 <mailto:cbowers@juniper.net>

       Editor:   Xufeng Liu
                 <mailto:xufeng_liu@jabil.com>

       Editor:   Prateek Sarda
                 <mailto:prateek.sarda@ericsson.com>

       Editor:   Vikram Choudhary
                 <mailto:vikschw@gmail.com>";

    description
      "This YANG module defines a model for managing Routing
       Information Protocol (RIP), including RIP version 2 and RIPng.

       Copyright (c) 2016 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject to
       the license terms contained in, the Simplified BSD License set
       forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
```

```
      (http://trustee.ietf.org/license-info).";

    revision 2017-12-05 {
      description
        "Initial revision.";
      reference
        "RFC 2453: RIP Version 2.
         RFC 2080: RIPng for IPv6.
         RFC 1724: RIP Version 2 MIB Extension.";
    }

    /*
     * Features
     */
    feature bfd {
      description
        "This feature indicates that the RIP implementation on the
         system supports BFD (Bidirectional Forwarding Detection).";
    }

    feature explicit-neighbors {
      description
        "This feature indicates that the system supports explicit
         neighbor configuration on a RIP interface.";
    }

    feature global-statistics {
      description
        "This feature indicates that the system supports collecting
         global statistic data related to RIP.";
    }

    feature interface-statistics {
      description
        "This feature indicates that the system supports collecting
         per-interface statistic data related to RIP.";
    }

    /*
     * Typedefs
     */

    typedef prefix-set-ref {
      type string;
      description
        "A type for a reference to a prefix set.
         The string value is the name identifier for uniquely
         identifying the referenced prefix set, which contains a list
```

```
          of prefixes that a routing policy can applied. The definition
          of such a prefix set is outside the scope of this document.";
      }

      typedef route-policy-ref {
        type string;
        description
          "A type for a reference to a route policy.
           The string value is the name identifier for uniquely
           identifying the referenced routing policy, which contains one
           or more policy rules that can be used for a routing decision.
           The definition of such a routing policy is outside the scope
           of this document.";
      }

      /*
       * Identities
       */

      identity rip {
        base rt:routing-protocol;
        description "Identity for the RIP routing protocol.";
      }

      identity ripv2 {
        base rip:rip;
        description "Identity for RIPv2 (RIP version 2).";
      }

      identity ripng {
        base rip:rip;
        description "Identity for RIPng.";
      }

      /*
       * Groupings
       */

      grouping originate-default-route-container {
        description
          "Containing settings whether to originate the default route
           in RIP routing instance.";
        container originate-default-route {
          description
            "Injects the default route into the RIP (RIPv2 or RIPng)
             routing instance.";
          leaf enabled {
            type boolean;
```

```
          default false;
          description
            "'true' if originating default route is enabled.";
        }
        leaf route-policy {
          type route-policy-ref;
          description
            "The conditions of the route policy are applied to the
             default route.";
        }
      }
    }

    grouping redistribute-container {
      description
        "Container of redistribute attributes.";

      container redistribute {
        description
          "Redistributes routes learned from other routing protocols
           into the RIP routing instance.";
        list bgp {
          key "asn";
          description
            "Redistributes routes from the specified BGP (Border
             Gateway Protocol) autonomous system (AS) into the RIP
             routing instance.";
          leaf asn {
            type inet:as-number;
            description
              "BGP autonomous system (AS) number.";
          }
          uses redistribute-route-policy-attributes;
        }
        container cg-nat {
          presence
            "Present if Carrier Grade Network Address Translation
             (CGNAT) routes are redistributed.";
          description
            "Carrier Grade Network Address Translation (CGNAT)
             routes.";
          uses redistribute-route-policy-attributes;
        }
        container connected {
          presence
            "Present if directly attached network routes are
             redistributed.";
          description
```

```
            "Redistributes directly attached networks into the RIP
             routing instance.";
          uses redistribute-route-policy-attributes;
        }
        container ipsec {
          presence
            "Present if IP security routing instance routes
             are redistributed.";
          description
            "Redistributes routes from the IP security routing
             instance into the RIP routing instance.";
          uses redistribute-route-policy-attributes;
        }
        list isis {
          key "instance";
          description
            "Redistributes IS-IS routes.";
          leaf instance {
            type leafref {
              path "../../../../../rt:control-plane-protocol/rt:name";
            }
            must "derived-from-or-self("
              + "../../../../../rt:control-plane-protocol"
              + "[rt:name = current()]/rt:type, 'isis:isis')" {
              description
                "The type of the routing protocol must be 'isis'";
            }
            description
              "Redistributes routes from the specified IS-IS routing
               instance into the RIP routing instance.";
          }
          leaf level {
            type enumeration {
              enum 1 {
                description "IS-IS level 1 routes.";
              }
              enum 2 {
                description "IS-IS level 2 routes.";
              }
              enum 1-2 {
                description "IS-IS level 1-2 routes.";
              }
            }
            description
              "IS-IS level.";
          }
          uses redistribute-route-policy-attributes;
        }
```

```
        container nat {
          presence
            "Present if Network Address Translation (NAT) routes
             are redistributed.";
          description
            "Redistributes Network Address Translation (NAT)
             routes into the RIP routing instance.";
          uses redistribute-route-policy-attributes;
        }
        list ospfv2 {
          when "derived-from-or-self(../../../rt:type, 'rip:ripv2')" {
            description
              "Applicable to RIPv2.";
          }
          key "instance";
          description
            "Redistributes routes from the specified OSPFv2 routing
             instance into the RIPv2 routing instance.";
          leaf instance {
            type leafref {
              path "../../../../../rt:control-plane-protocol/rt:name";
            }
            must "derived-from-or-self("
              + "../../../../../rt:control-plane-protocol"
              + "[rt:name = current()]/rt:type, 'ospf:ospfv2')" {
              description
                "The type of the routing protocol must be 'ospfv2'";
            }
            description
              "OSPFv2 instance ID. Redistributes routes from the
               specified OSPFv2 routing instance into the RIPv2 routing
               instance. ";
          }
          leaf route-type {
            type ospf:route-type;
            description
              "Redistributes only those OSPFv2 routes matching the
               specified route type into the RIPv2 routing instance.";
          }
          uses redistribute-route-policy-attributes;
        }
        list ospfv3 {
          when "derived-from-or-self(../../../rt:type, 'rip:ripng')" {
            description
              "Applicable to RIPng.";
          }
          key "instance";
          description
```

```
            "Redistributes routes from the specified OSPFv3 routing
             instance into the RIPng routing instance.";
          leaf instance {
            type leafref {
              path "../../../../../rt:control-plane-protocol/rt:name";
            }
            must "derived-from-or-self("
              + "../../../../../rt:control-plane-protocol"
              + "[rt:name = current()]/rt:type, 'ospf:ospfv3')" {
              description
                "The type of the routing protocol must be 'ospfv3'";
            }
            description
              "OSPFv3 instance ID. Redistributes routes from the
               specified OSPFv3 routing instance into the RIPng routing
               instance. ";
          }
          leaf route-type {
            type ospf:route-type;
            description
              "Redistributes only those OSPFv3 routes matching the
               specified route type into the RIPng routing instance.";
          }
          uses redistribute-route-policy-attributes;
        }
        list ripv2 {
          when "derived-from-or-self(../../../rt:type, 'rip:ripv2')" {
            description
              "Applicable to RIPv2.";
          }
          key "instance";
          description
            "Redistributes routes from another RIPv2 routing instance
             into the current RIPv2 routing instance.";
          leaf instance {
            type leafref {
              path "../../../../../rt:control-plane-protocol/rt:name";
            }
            must "derived-from-or-self("
              + "../../../../../rt:control-plane-protocol"
              + "[rt:name = current()]/rt:type, 'rip:ripv2')" {
              description
                "The type of the routing protocol must be 'ripv2'";
            }
            description
              "Redistributes routes from the specified RIPv2 routing
               instance into the RIPv2 routing instance.";
          }
```

```
          uses redistribute-route-policy-attributes;
        }
        list ripng {
          when "derived-from-or-self(../../../rt:type, 'rip:ripng')" {
            description
              "Applicable to RIPng.";
          }
          key "instance";
          description
            "Redistributes routes from another RIPng routing instance
             into the current RIPng routing instance.";
          leaf instance {
            type leafref {
              path "../../../../../rt:control-plane-protocol/rt:name";
            }
            must "derived-from-or-self("
              + "../../../../../rt:control-plane-protocol"
              + "[rt:name = current()]/rt:type, 'rip:ripng')" {
              description
                "The type of the routing protocol must be 'ripng'";
            }
            description
              "Redistributes routes from the specified RIPng routing
               instance into the RIPng routing instance.";
          }
          uses redistribute-route-policy-attributes;
        }
        container static {
          presence "Present if redistributing static routes.";
          description
            "Redistributes static routes into the RIP routing
             instance.";
          uses redistribute-route-policy-attributes;
        }
      } // redistribute
    } // redistribute-container

    grouping redistribute-route-policy-attributes {
      description
        "Attributes for redistributing a route policy.";
      leaf metric {
        type uint8 {
          range 0..16;
        }
        description
          "Metric used for the redistributed route. If a metric is
           not specified, the metric configured with the
           default-metric attribute in RIP router configuration is
```

```
            used. If the default-metric attribute has not been
            configured, the default metric for redistributed routes
            is 0.";
        }
        leaf route-policy {
          type route-policy-ref;
          description
            "Applies the conditions of the specified route policy to
             routes that are redistributed into the RIP routing
             instance.";
        }
      } // redistribute-route-policy-attributes

      grouping timers-container {
        description
          "Container for settings of basic timers";
        container timers {
          must "invalid-interval >= (update-interval * 3)" {
            description
              "invalid-interval must be at least three times the value
               for the update-interval argument.";
          }
          must "flush-interval > invalid-interval" {
            description
              "flush-interval must be larger than the value for the
               invalid-interval argument";
          }
          description
            "Timers for the specified RIPv2 or RIPng instance or
             interface.";
          leaf update-interval {
            type uint16 {
              range  1..32767;
            }
            units seconds;
            default 30;
            description
              "Interval at which RIPv2 or RIPng updates are sent.";
          }
          leaf invalid-interval {
            type uint16 {
              range 1..32767;
            }
            units seconds;
            default 180;
            description
              "Interval before a route is declared invalid after no
               updates are received. This value is at least three times
```

```
             the value for the update-interval argument.";
        }
        leaf holddown-interval {
          type uint16 {
            range 1..32767;
          }
          units seconds;
          default 180;
          description
            "Interval before better routes are released.";
        }
        leaf flush-interval {
          type uint16 {
            range 1..32767;
          }
          units seconds;
          default 240;
          description
            "Interval before a route is flushed from the routing
             table. This value must be larger than the value for the
             invalid-interval argument.";
        }
      } // timers
    }

    grouping global-attributes {
      description
        "Global configuration and state attributes.";
      uses originate-default-route-container;

      leaf default-metric {
        type uint8 {
          range 0..16;
        }
        default 0;
        description
          "Set the default metric.";
      }

      leaf distance {
        type uint8 {
          range 1..255;
        }
        default 120;
        description
          "The administrative distance of the RIPv2 or RIPng for the
           current RIPv2 or RIPng instance.";
      }
```

```
      leaf triggered-update-threshold {
        type uint8 {
          range 1..30;
        }
        units seconds;
        default 5;
        description
          "This attribute is used to suppress triggered updates.
           When the arrival of a regularly scheduled update matches the
           number of seconds or is less than the number seconds
           configured with this attribute, the triggered update is
           suppressed.";
      }

      leaf maximum-paths {
        type uint8 {
          range 1..16;
        }
        default 8;
        description
          "The number of multiple equal-cost RIPv2 or RIPng routes
           that can be used as the best paths for balancing the load
           of outgoing traffic packets.";
      }

      leaf output-delay {
        type uint8 {
          range 1..50;
        }
        units milliseconds;
        description
          "A delay time between packets sent in multipacket
           RIPv2 or RIPng updates.";
      }
    } // global-attributes

    grouping distribute-lists {
      description
        "Grouping for distribute lists.";
      list distribute-list {
        key "prefix-set-name direction";
        description
          "List of distribute-lists, which are used to filter in-coming
           or out-going routing updates.";

        leaf prefix-set-name {
          type prefix-set-ref;
          description
```

```
            "Reference to a prefix list to be applied to RIPv2 or
             RIPng packets.";
        }

        leaf direction {
          type enumeration {
            enum "in" {
              description
                "Apply the distribute-list to in-coming routes.";
            }
            enum "out" {
              description
                "Apply the distribute-list to out-going routes.";
            }
          }
          description
            "Direction of the routing updates.";
        }

        leaf if-name {
          type if:interface-ref;
          description
            "Reference to an interface to which the prefix list is
             applied.";
        }
      }
    } // distribute-lists

    grouping route-attributes {
      description
        "Grouping for route attributes.";
      leaf redistributed {
        type boolean;
        description
          "Redistributed routes";
      }

      leaf route-type {
        type enumeration {
          enum connected {
            description "Connected route.";
          }
          enum external {
            description "External route.";
          }
          enum external-backup {
            description "External backup route.";
          }
```

```
          enum rip {
            description "RIP route.";
          }
        }
        description
          "Route type.";
      }
      leaf metric {
        type uint8 {
          range 0..16;
        }
        description "Route metric.";
      }
      leaf expire-time {
        type uint16;
        description "Expiration time.";
      }
      leaf deleted {
        type boolean;
        description "Deleted route.";
      }
      leaf holddown {
        type boolean;
        description "Holddown route.";
      }
      leaf need-triggered-update {
        type boolean;
        description "The route needs triggered update.";
      }
      leaf inactive {
        type boolean;
        description "The route is inactive.";
      }
      leaf flush-expire-before-holddown {
        type boolean;
        description
          "The flush timer expired before holddown time.";
      }
    } // route-attribute

    /*
     * Configuration data and operational state data nodes
     */

    augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol" {
      when "derived-from(rt:type, 'rip:rip')" {
        description
```

```
          "This augment is only valid for a routing protocol instance
           of RIP (type 'ripv2' or 'ripng').";
      }
      description "RIP augmentation.";

      container rip {
        description
          "RIP data.";

        uses global-attributes;
        uses distribute-lists;
        uses redistribute-container;
        uses timers-container;

        container interfaces {
          description
            "Containing a list of RIP interfaces.";
          list interface {
            key "interface";
            description
              "List of RIP interfaces.";
            leaf interface {
              type if:interface-ref;
              must "(derived-from-or-self("
                + "../../../../rt:type, 'rip:ripv2') and "
                + "/if:interfaces/if:interface[if:name=current()]/"
                + "ip:ipv4) or "
                + "(derived-from-or-self("
                + "../../../../rt:type, 'rip:ripng') and "
                + "/if:interfaces/if:interface[if:name=current()]/"
                + "ip:ipv6)" {
                error-message "Invalid interface type.";
                description
                  "RIPv2 can be enabled on IPv4 interfae, and
                   RIPng can be enabled on IPv6 interface.";
              }
              description
                "Enable RIP on this interface.";
            }

            container authentication {
              when "derived-from-or-self("
                + "../../../../rt:type, 'rip:ripv2')" {
                description "Only applicable to RIPv2.";
              }
              description
                "Enables authentication and specifies the
                 authentication scheme for the RIP interface";
```

```
        choice auth-type-selection {
          description
            "Specify the authentication scheme.";
          reference
            "RFC8177: YANG Data Model for Key Chains.";
          case auth-key-chain {
            leaf key-chain {
              type key-chain:key-chain-ref;
              description
                "key-chain name.";
            }
          }
          case auth-key {
            leaf key {
              type string;
              description
                "Key string in ASCII format.";
            }
            leaf crypto-algorithm {
              type identityref {
                base key-chain:crypto-algorithm;
              }
              description
                "Cryptographic algorithm associated with key.";
            }
          }
        }
      }

      container bfd {
        if-feature bfd;
        description "BFD configuration.";
        uses bfd-types:client-cfg-parms;
      }

      leaf cost {
        type uint8 {
          range  1..16;
        }
        default 1;
        description
          "Interface cost.";
      }

      container neighbors {
        if-feature explicit-neighbors;
        description
          "Specifies the RIP neighbors. Useful for a
```

```
           non-broadcast multiple access (NBMA) network.";
        list neighbor {
          key "address";
          description
            "Specify a RIP neighbor on a non-broadcast network.";
          leaf address {
            type inet:ip-address;
            description "Neighbor IP address.";
          }
        }
      }

      leaf no-listen {
        type empty;
        description
          "Disables listening to and processing of RIPv2 or RIPng
           packets on the specified interface.";
      }

      uses originate-default-route-container;

      leaf passive {
        type empty;
        description
          "Disables sending of RIPv2 or RIPng packets on the
           specified interface.";
      }

      leaf split-horizon {
        type enumeration {
          enum disabled {
            description
              "Disables split-horizon processing.";
          }
          enum simple {
            description
              "Enables simple split-horizon processing.";
          }
          enum poison-reverse {
            description
              "Enables split-horizon processing with poison
               reverse.";
          }
        }
        default simple;
        description
          "Controls RIPv2 or RIPng split-horizon processing on
           the specified interface.";
```

```
            }

            container summary-address {
              description
                "Summarizes information about RIPv2 or RIPng routes
                 sent over the specified interface in RIPv2 or RIPng
                 update packets.";
              leaf address {
                type inet:ip-prefix;
                description
                  "IPv4 address, in the form A.B.C.D, and the prefix
                   length, separated by the slash (/) character;
                   or IPv6 address, in the form A:B:C:D:E:F:G:H, and
                   the prefix length, separated by the slash (/)
                   character.";
              }
              leaf metric {
                type uint8 {
                  range 0..16;
                }
                description
                  "Metric used for the route. If this attribute is not
                   used, the value set through the default-metric
                   attribute in RIPv2 or RIPng router configuration is
                   used for the route. ";
              }
            }

            uses timers-container;

            /* Operational state */
            leaf oper-status {
              type enumeration {
                enum up {
                  description
                    "RIPv2 or RIPng is operational on this interface.";
                }
                enum down {
                  description
                    "RIPv2 or RIPng is not operational on this
                     interface.";
                }
              }
              config false;
              description
                "Operational state.";
            }
            leaf next-full-update {
```

```
            type uint32;
            config false;
            description
              "Next full update time.";
          }
          leaf valid-address {
            type boolean;
            config false;
            description
              "The interface has a valid address.";
          }

          container statistics {
            if-feature interface-statistics;
            config false;
            description
              "Interface statistic counters.";
            leaf discontinuity-time {
              type yang:date-and-time;
              description
                "The time on the most recent occasion at which any
                 one or more of the statistic counters suffered a
                 discontinuity. If no such discontinuities have
                 occurred since the last re-initialization of the
                 local management subsystem, then this node contains
                 the time the local management subsystem
                 re-initialized itself.";
            }
            leaf bad-packets-rcvd {
              type yang:counter32;
              description
                "The number of RIP invalid packets received by
                 the RIP process which were subsequently discarded
                 for any reason (e.g. a version 0 packet, or an
                 unknown command type).";
            }
            leaf bad-routes-rcvd {
              type yang:counter32;
              description
                "The number of routes, in valid RIP packets,
                 which were ignored for any reason (e.g. unknown
                 address family, or invalid metric).";
            }
            leaf updates-sent {
              type yang:counter32;
              description
                "The number of triggered RIP updates actually
                 sent on this interface.  This explicitly does
```

```
                NOT include full updates sent containing new
                information.";
            }
          }
        } // interface
      } // interfaces

      /* Operational state */
      leaf next-triggered-update {
        type uint32;
        config false;
        description
          "Next triggered update.";
      }
      leaf num-of-routes {
        type uint32;
        config false;
        description
          "The number of routes.";
      }

      container ipv4 {
        when "derived-from-or-self(../../rt:type, 'rip:ripv2')" {
          description
            "IPv4 address family is supported by RIPv2.";
        }
        config false;
        description
          "IPv4 address family information.";
        container neighbors {
          description
            "IPv4 neighbor information.";
          list neighbor {
            key "ipv4-address";
            description
              "A RIPv2 neighbor.";

            leaf ipv4-address {
              type inet:ipv4-address;
              description
                "IP address that a RIP neighbor is using as its
                 source address.";
            }
            leaf last-update {
              type yang:date-and-time;
              description
                "The time when the most recent RIP update was
                 received from this neighbor.";
```

```
            }
            leaf bad-packets-rcvd {
              type yang:counter32;
              description
                "The number of RIP invalid packets received from
                 this neighbor which were subsequently discarded
                 for any reason (e.g. a version 0 packet, or an
                 unknown command type).";
            }
            leaf bad-routes-rcvd {
              type yang:counter32;
              description
                "The number of routes received from this neighbor,
                 in valid RIP packets, which were ignored for any
                 reason (e.g. unknown address family, or invalid
                 metric).";
            }
          } // neighbor
        } // neighbors
        container routes {
          description
            "IPv4 route information.";
          list route {
            key "ipv4-prefix";
            description
              "A RIPv2 IPv4 route.";

            leaf ipv4-prefix {
              type inet:ipv4-prefix;
              description
                "IP address (in the form A.B.C.D) and prefix length,
                 separated by the slash (/) character. The range of
                 values for the prefix-length is 0 to 32.";
            }
            leaf next-hop {
              type inet:ipv4-address;
              description
                "Next hop IPv4 address.";
            }
            leaf interface {
              type if:interface-ref;
              description
                "The interface that the route uses.";
            }
            uses route-attributes;
          } // route
        } // routes
      } // ipv4
```

```
        container ipv6 {
          when "derived-from-or-self(../../rt:type, 'rip:ripng')" {
            description
              "IPv6 address family is supported by RIPng.";
          }
          config false;
          description
            "IPv6 address family information.";
          container neighbors {
            description
              "IPv6 neighbor information.";
            list neighbor {
              key "ipv6-address";
              description
                "A RIPng neighbor.";

              leaf ipv6-address {
                type inet:ipv6-address;
                description
                  "IP address that a RIP neighbor is using as its
                   source address.";
              }
              leaf last-update {
                type yang:date-and-time;
                description
                  "The time when the most recent RIP update was
                   received from this neighbor.";
              }
              leaf bad-packets-rcvd {
                type yang:counter32;
                description
                  "The number of RIP invalid packets received from
                   this neighbor which were subsequently discarded
                   for any reason (e.g. a version 0 packet, or an
                   unknown command type).";
              }
              leaf bad-routes-rcvd {
                type yang:counter32;
                description
                  "The number of routes received from this neighbor,
                   in valid RIP packets, which were ignored for any
                   reason (e.g. unknown address family, or invalid
                   metric).";
              }
            } // neighbor
          } // neighbors
          container routes {
            description
```

```
            "IPv6 route information.";
          list route {
            key "ipv6-prefix";
            description
              "A RIPng IPv6 route.";

            leaf ipv6-prefix {
              type inet:ipv6-prefix;
              description
                "IP address (in the canonical format defined in
                 RFC5952) and prefix length, separated by the slash
                 (/) character. The range of values for the
                 prefix-length is 0 to 128.";
            }
            leaf next-hop {
              type inet:ipv6-address;
              description
                "Next hop IPv6 address.";
            }
            leaf interface {
              type if:interface-ref;
              description
                "The interface that the route uses.";
            }
            uses route-attributes;
          } // route
        } // routes
      } // ipv6

      container statistics {
        if-feature global-statistics;
        config false;
        description
          "Global statistic counters.";
        leaf discontinuity-time {
          type yang:date-and-time;
          description
            "The time on the most recent occasion at which any one
             or more of the statistic counters suffered a
             discontinuity. If no such discontinuities have occurred
             since the last re-initialization of the local
             management subsystem, then this node contains the time
             the local management subsystem re-initialized itself.";
        }
        leaf requests-rcvd {
          type yang:counter32;
          description
            "The number of requests received by RIP.";
```

```
          }
          leaf requests-sent {
            type yang:counter32;
            description
              "The number of requests sent by RIP.";
          }
          leaf responses-rcvd {
            type yang:counter32;
            description
              "The number of responses received by RIP.";
          }
          leaf responses-sent {
            type yang:counter32;
            description
              "The number of responses sent by RIP.";
          }
        } // statistics
      } // container rip
    }

    /*
     * RPCs
     */

    rpc clear-rip-route {
      description
        "Clears RIP routes from the IP routing table and routes
         redistributed into the RIP protocol for the specified RIP
         instance or for all RIP instances in the current context.";

      input {
        leaf rip-instance {
          type leafref {
            path "/rt:routing/rt:control-plane-protocols/"
              + "rt:control-plane-protocol/rt:name";
          }
          description
            "Instance name identifying a specific RIP instance.
             This leaf is optional for the rpc.
             If it is specified, the rpc will clear all routes in the
             specified RIP instance;
             if it is not specified, the rpc will clear all routes in
             all RIP instances.";
        }
      }
    } // rcp clear-rip-route
  }
  <CODE ENDS>
```

5. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

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

This document registers the following YANG modules in the YANG Module Names registry [RFC7950]:

```
---------------------------------------------------------------------
name:         ietf-rip
namespace:    urn:ietf:params:xml:ns:yang:ietf-rip
prefix:       rip
reference:    RFC XXXX
---------------------------------------------------------------------
```

6. Security Considerations

The configuration, state, and action data defined in this document are designed to be accessed via a management protocol with a secure transport layer, such as NETCONF [RFC6241]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and contents.

A number of configuration data nodes defined in this document are writable/creatable/deletable (i.e., "config true" in YANG terms, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations to these data nodes, such as "edit-config" in NETCONF, can have negative effects on the network if the protocol operations are not properly protected. The vulnerable "config true" parameters and subtrees are the following:

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/rip:rip

Unauthorized access to any node of these can adversely affect the routing subsystem of both the local device and the network. This may

lead to network malfunctions, delivery of packets to inappropriate destinations, and other problems.

This data model also defines a RPC "clear-rip-route", which may affect the routing subsystem in the same way as described above.

7. References

7.1. Normative References

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

[RFC2453] Malkin, G., "RIP Version 2", STD 56, RFC 2453, DOI 10.17487/RFC2453, November 1998, <https://www.rfc-editor.org/info/rfc2453>.

[RFC2080] Malkin, G. and R. Minnear, "RIPng for IPv6", RFC 2080, DOI 10.17487/RFC2080, January 1997, <https://www.rfc-editor.org/info/rfc2080>.

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

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

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

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

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[I-D.bjorklund-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-bjorklund-netmod-rfc7223bis-00 (work in progress), August 2017.

[I-D.bjorklund-netmod-rfc7277bis]
Bjorklund, M., "A YANG Data Model for IP Management", draft-bjorklund-netmod-rfc7277bis-00 (work in progress), August 2017.

[I-D.acee-netmod-rfc8022bis]
Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NDMA Version)", draft-acee-netmod-rfc8022bis-06 (work in progress), October 2017.

[I-D.ietf-bfd-yang]
Rahman, R., Zheng, L., Jethanandani, M., Networks, J., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-07 (work in progress), October 2017.

[I-D.ietf-isis-yang-isis-cfg]
Litkowski, S., Yeung, D., Lindem, A., Zhang, Z., and L. Lhotka, "YANG Data Model for IS-IS protocol", draft-ietf-isis-yang-isis-cfg-19 (work in progress), November 2017.

[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-09 (work in progress), October 2017.

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-07 (work in progress), November 2017.

7.2. Informative References

[RFC1724] Malkin, G. and F. Baker, "RIP Version 2 MIB Extension", RFC 1724, DOI 10.17487/RFC1724, November 1994, <https://www.rfc-editor.org/info/rfc1724>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

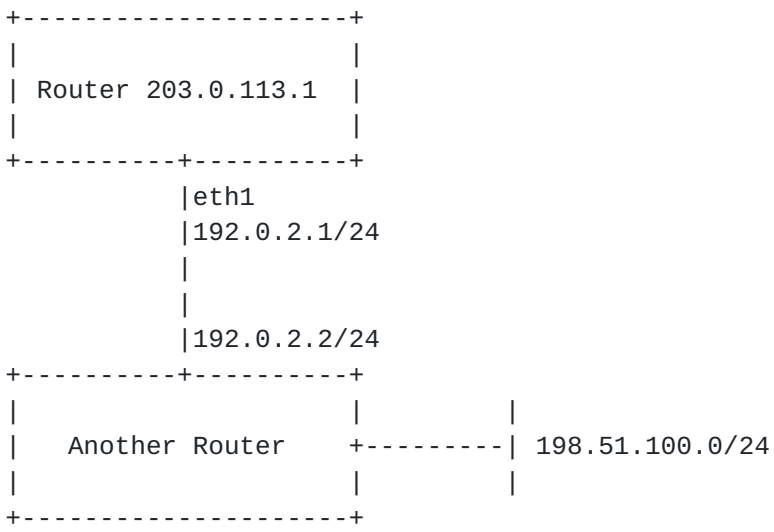
[I-D.ietf-netconf-rfc5277bis]
Clemm, A., Prieto, A., Voit, E., Nilsen-Nygaard, E., Tripathy, A., Chisholm, S., and H. Trevino, "Subscribing to Event Notifications", draft-ietf-netconf-rfc5277bis-01 (work in progress), October 2016.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-11 (work in progress), October 2017.

[I-D.ietf-netmod-rfc6087bis]
Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-14 (work in progress), September 2017.

Appendix A. Data Tree Example

This section contains an example of an instance data tree in the JSON encoding [RFC7951], containing both configuration and state data.



The configuration instance data tree for Router 203.0.113.1 in the above figure could be as follows:

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with RIPv2 enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "192.0.2.1",
              "prefix-length": 24
            }
          ],
          "forwarding": true
        }
      }
    ]
  },
  "ietf-routing:routing": {
    "router-id": "203.0.113.1",
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "ietf-rip:ripv2",
          "name": "ripv2-1",
          "description": "RIPv2 instance ripv2-1.",
          "ietf-rip:rip": {
            "redistribute": {
              "connected": {
              }
            }
            "interfaces": {
              "interface": [
                {
                  "interface": "eth1",
                  "split-horizon": "poison-reverse"
                }
              ]
            }
          }
        }
      ]
    }
  }
}
```

The cooresponding operational state data for Router 203.0.113.1 could be as follows:

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth1",
        "description": "An interface with RIPv2 enabled.",
        "type": "iana-if-type:ethernetCsmacd",
        "phys-address": "00:0C:42:E5:B1:E9",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2016-10-24T17:11:27+02:00"
        },
        "ietf-ip:ipv4": {
          "forwarding": true,
          "mtu": 1500,
          "address": [
            {
              "ip": "192.0.2.1",
              "prefix-length": 24
            }
          ]
        }
      }
    ]
  },
  "ietf-routing:routing": {
    "router-id": "203.0.113.1",
    "interfaces": {
      "interface": [
        "eth1"
      ]
    },
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "ietf-rip:rip",
          "name": "ripv2-1"
          "ietf-rip:rip": {
            "default-metric": 1,
            "next-triggered-update": 5
            "interfaces": {
              "interface": [
                {
                  "interface": "eth1",
                  "oper-status": "up",
```

```
                    "cost": 1,
                    "split-horizon": "poison-reverse",
                    "valid-address": true
                  }
                ]
              },
              "ipv4" {
                "neighbors": {
                  "neighbor": [
                    {
                      "address": "192.0.2.2"
                    }
                  ]
                }
                "routes": {
                  "route": [
                    {
                      "ipv4-prefix": "192.0.2.1/24",
                      "interface": "eth1",
                      "redistributed": true,
                      "route-type": "connected",
                      "metric": 0,
                      "expire-time": 22
                    },
                    {
                      "ipv4-prefix": "198.51.100.0/24",
                      "next-hop": "192.0.2.2",
                      "interface": "eth1",
                      "redistributed": false,
                      "route-type": "rip",
                      "metric": 1,
                      "expire-time": 82
                    }
                  ]
                }
              },
              "statistics": {
                "discontinuity-time": "2016-10-24T17:11:27+02:00",
                "requests-rcvd": 523,
                "requests-sent": 262,
                "responses-rcvd": 261,
                "responses-sent": 523
              }
            }
          }
        ]
      }
    }
```

```
}
```

Authors' Addresses

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

EMail: Xufeng_Liu@jabil.com

Prateek Sarda
Ericsson
Fern Icon, Survey No 28 and 36/5, Doddanakundi Village
Bangalore Karnataka 560037
India

EMail: prateek.sarda@ericsson.com

Vikram Choudhary
Individual
Bangalore 560066
India

EMail: vikschw@gmail.com