

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 3, 2016

X. Liu
P. Sarda
Ericsson
V. Choudhary
Huawei Technologies
August 3, 2015

A YANG Data Model for Routing Information Protocol (RIP)
draft-ietf-rtgwg-yang-rip-00.txt

Abstract

This document describes a data model for 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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on January 1, 2015.

Copyright Notice

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

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

Internet-Draft [draft-ietf-rtgwg-yang-rip-00.txt](#)

August 2015

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

Table of Contents

1.	Introduction.....	2
1.1.	Terminology.....	2
2.	RIP YANG model overview.....	3
3.	RIP YANG module.....	7
4.	IANA Considerations.....	28
5.	Security Considerations.....	28
6.	References.....	29
6.1.	Normative References.....	29
6.2.	Informative References.....	29

[1.](#) Introduction

This document introduces a YANG [[RFC6020](#)] data model for 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 keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#), [[RFC2119](#)].

The following terms are defined in [[RFC6020](#)] and are not redefined here:

- o augment

- o data model
- o data node

[2. RIP YANG model overview](#)

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

```

module: ietf-rip
augment /rt:routing/rt:routing-instance/rt:routing-protocols
/rt:routing-protocol:
  +--rw rip
    +--rw originate-default-route!
      | +--rw route-map?  string
    +--rw default-metric?          uint8
    +--rw distance?                uint8
    +--rw flash-update-threshold?  uint8
    +--rw maximum-paths?          uint8
    +--rw output-delay?           uint8
    +--rw distribute-list* [prefix-list-name direction]
      | +--rw prefix-list-name  string
      | +--rw direction        enumeration
      | +--rw if-name?         if:interface-ref
    +--rw timers
      | +--rw update-interval?   uint16
      | +--rw invalid-interval?  uint16
      | +--rw holddown-interval? uint16
      | +--rw flush-interval?    uint16
    +--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
        |     +--rw (algorithm)?
        |     | +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
        |     | | +--rw hmac-sha1-12?  empty
        |     | +--:(md5)

```

```

|           | +--rw md5?           empty
|           +---:(sha-1)
|           | +--rw sha-1?       empty
|           +---:(hmac-sha-1)

```

Internet-Draft [draft-ietf-rtgwg-yang-rip-00.txt](#)

August 2015

```

|           | +--rw hmac-sha-1?       empty
|           +---:(hmac-sha-256)
|           | +--rw hmac-sha-256?   empty
|           +---:(hmac-sha-384)
|           | +--rw hmac-sha-384?   empty
|           +---:(hmac-sha-512)
|           +---rw hmac-sha-512?   empty
+--rw bfd?                          boolean {bfd}?
+--rw cost?                          uint8
+--rw neighbors {neighbor-configuration}?
| +--rw neighbor* [address]
|   +--rw address      inet:ip-address
+--rw no-listen?                    empty
+--rw no-supply?                    empty
+--rw originate-default-route!
| +--rw route-map?   string
+--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
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols
/rt:routing-protocol:
  +--ro rip
    +--ro originate-default-route!
    | +--ro route-map?   string
    +--ro default-metric?      uint8
    +--ro distance?            uint8
    +--ro flash-update-threshold?  uint8
    +--ro maximum-paths?        uint8
    +--ro output-delay?         uint8
    +--ro distribute-list* [prefix-list-name direction]
    | +--ro prefix-list-name  string
    | +--ro direction        enumeration

```

```
| +--ro if-name?          if:interface-ref
+--ro next-flash-update?  uint32
+--ro num-of-routes?     uint32
```

```
+--ro routing-table-version?  uint32
+--ro timers
| +--ro update-interval?      uint16
| +--ro invalid-interval?     uint16
| +--ro holddown-interval?    uint16
| +--ro flush-interval?       uint16
+--ro interface* [interface]
| +--ro interface             if:interface-ref
| +--ro oper-status?          enumeration
| +--ro cost?                 uint8
| +--ro listen?               boolean
| +--ro next-full-update?     uint32
| +--ro originate-default-route? boolean
| +--ro poison-reverse?       boolean
| +--ro split-horizon?        boolean
| +--ro supply?               boolean
| +--ro valid-address?        boolean
| +--ro timers
| | +--ro update-interval?    uint16
| | +--ro invalid-interval?   uint16
| | +--ro holddown-interval?  uint16
| | +--ro flush-interval?     uint16
| +--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 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-flash?           boolean
|      +---ro need-download-to-rib? boolean
|      +---ro inactive?             boolean
|      +---ro next-hop-flags?       bits
+---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-flash?         boolean
|  |  |  +---ro need-download-to-rib? boolean
|  |  |  +---ro inactive?           boolean
|  |  |  +---ro next-hop-flags?     bits
+---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
|  +---ro input
|  |  +---ro instance-name?         leafref
```

Internet-Draft [draft-ietf-rtgwg-yang-rip-00.txt](#)

August 2015

3. RIP YANG module

```
<CODE BEGINS> file "ietf-rip@2015-06-22.yang"
module ietf-rip {
  namespace "urn:ietf:params:xml:ns:yang:ietf-rip";
  // replace with IANA namespace when assigned
  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";
  }

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

  revision 2015-06-22 {
    description
      "Initial revision.";
```

Internet-Draft [draft-ietf-rtgwg-yang-rip-00.txt](#)

August 2015

```
reference
  "RFC 2453: RIP Version 2.
  "RFC 2080: RIPng for IPv6.
  "RFC 1724: RIP Version 2 MIB Extension.";
}

/*
 * Features
 */

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

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

feature bfd {
  description
    "This feature indicates that the system supports BFD.";
}

feature neighbor-configuration {
  description
    "This feature indicates that the system supports
    neighbor configuration.";
}

/*
 * Identities
 */

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


Internet-Draft

[draft-ietf-rtgwg-yang-rip-00.txt](#)

August 2015

```
identity ripv2 {
  base "rip:rip";
  description "RIPv2";
}

identity ripng {
  base "rip:rip";
  description "RIPng";
}

/*
 * Groupings
 */

grouping originate-default-route-container {
  description
    "Container for setting of originating default route.";
  container originate-default-route {
    presence "Present if originating default route is enabled.";
    description
      "Injects the default route into the RIP or RIPng
      instance.";
    leaf route-map {
      type string;
      description
        "The conditions of the route map are applied to the default
        route.";
    }
  }
}

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 RIP or RIPng instance or
  interface.";
leaf update-interval {
  type uint16 {
    range 1..32767;
  }
  units seconds;
  default 30;
  description
    "Interval at which RIP 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 RIP or RIPng for the
            current RIP or RIPng instance.";
    }

    leaf flash-update-threshold {
        type uint8 {
            range 1..30;
        }
        units seconds;
        default 5;
        description
```

```
        "The minimum interval between consecutive RIP or RIPng
        flash updates.";
    }

    leaf maximum-paths {
        type uint8 {
            range 1..16;
        }
        default 8;
        description
            "The number of multiple equal-cost RIP 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
            RIP or RIPng updates.";
    }
} // global-attributes

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

        leaf prefix-list-name {
            type string;
            description
                "Reference to a prefix list to be applied to RIP 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-flash {
    type boolean;
    description "The route needs flash.";
}
leaf need-download-to-rib {
    type boolean;
    description "The route needs to be downloaded to rib.";
}
leaf inactive {
    type boolean;
    description "The route is inactive.";
}
```

```
leaf next-hop-flags {
  type bits {
    bit withdrawn-from-rib {
      description "The route is withdrawn from rib.";
    }
    bit holddown {
      description "The route is holddown.";
    }
    bit flush-expire-before-holddown {
      description
        "The flush timer expired before holddown time.";
    }
  }
  description "The next hop properties of the route.";
}
} // route-attribute

/*
 * Configuration data nodes
 */

augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
+ "rt:routing-protocol" {
  when "rt:type = 'rip:ripv2' or rt:type = 'rip:ripng'" {
    description
      "This augment is only valid for a routing protocol instance
      of RIP (type 'ripv2' or 'ripng').";
  }
  description "RIP augmentation.";

  container rip {
    description
      "RIP configuration data.";

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

    list interface {
      key "interface";
      description
```



```
"List of RIP interfaces.";
leaf interface {
  type if:interface-ref;
  must "(../.././rt:type = 'rip:ripv2' and "
    + "/if:interfaces/if:interface[name=current()]/"
    + "ip:ipv4) or "
    + "(../.././rt:type = 'rip:ripng' and "
    + "/if:interfaces/if:interface[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 "../.././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.
      The use of the key-chain reference here is:
      1) Designed to align with other proposed protocol
      models.
      2) Not finalized, pending resolution of alignment with
      the RFC published KeyTables information model.";
    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.";
    }
    container crypto-algorithm {
      uses key-chain:crypto-algorithm-types;
      description
        "Cryptographic algorithm associated with key.";
    }
  }
}

leaf bfd {
  if-feature bfd;
  type boolean;
  description
    "Enable/disable bfd.";
}

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

container neighbors {
  if-feature neighbor-configuration;
  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
        "Disable the specified interface to receive (listen to)
        and process RIP or RIPng packets.";
}

leaf no-supply {
    type empty;
    description
        "Disables sending of RIP or RIPng packets on the
        specified interface.";
}

uses originate-default-route-container;

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

container summary-address {
```

```
description
  "Summarizes information about RIP or RIPng routes sent
  over the specified interface in RIP 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 RIP or RIPng router configuration is
    used for the route. ";
}
}

uses timers-container;
} // interface
} // container rip
}

/*
 * Operational state data nodes
 */

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/rt:routing-protocol" {
  when "rt:type = 'rip:ripv2' or rt:type = 'rip:ripng'" {
    description
      "This augment is only valid for a routing protocol instance
      of type 'ripv2' or 'ripng'.";
  }
}
description
```

```
"RIP state.";
container rip {
  description "RIP operational state.";

  uses global-attributes;
  uses distribute-lists;

  leaf next-flash-update {
    type uint32;
    description
      "Next flash update.";
  }
  leaf num-of-routes {
    type uint32;
    description
      "The number of routes.";
  }
  leaf routing-table-version {
    type uint32;
    description
      "Routing table version.";
  }
}

uses timers-container;

list interface {
  key "interface";
  description
    "List of RIP interfaces.";
  leaf interface {
    type if:interface-ref;
    description
      "Enable RIP on this interface.";
  }
  leaf oper-status {
    type enumeration {
      enum up {
        description
          "RIPv2 or RIPv6 is operational on this interface.";
      }
      enum down {
```

```
        description
            "RIPv2 or RIPv6 is not operational on this
            interface.";
    }
}
description
    "Operational state.";
}
leaf cost {
    type uint8 {
        range 1..16;
    }
    default 1;
    description
        "Interface cost.";
}
leaf listen {
    type boolean;
    description
        "The interface is enabled to receive (listen to)
        and process RIP or RIPv6 packets.";
}
leaf next-full-update {
    type uint32;
    description
        "Next full update time.";
}
leaf originate-default-route {
    type boolean;
    description
        "'true' if originating default route is enabled.";
}
leaf poison-reverse {
    type boolean;
    description
        "'true' if Split Horizon with Poisoned Reverse is
        enabled.";
}
leaf split-horizon {
    type boolean;
    description
```

```
        "'true' if Split Horizon processing is enabled.";
    }
    leaf supply {
        type boolean;
        description
            "The interface is enabled to supply (send) RIP or RIPng
            packets.";
    }
    leaf valid-address {
        type boolean;
        description
            "The interface has a valid address.";
    }
    uses timers-container;

    container statistics {
        if-feature interface-statistics;
        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

container ipv4 {
    when "../..//rt:type = 'rip:ripv2'" {
        description
            "IPv4 address family is supported by RIPv2.";
    }
    description
        "IPv4 address family information.";
    container neighbors {
        description
            "IPv4 neighbor information.";
        list neighbor {
            key "ipv4-address";
            description
                "A RIPv2 RIP 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 "../..//rt:type = 'rip:ripng'" {
        description
            "IPv6 address family is supported by RIPng.";
    }
    description
        "IPv6 address family information.";
    container neighbors {
        description
            "IPv6 neighbor information.";
        list neighbor {
            key "ipv6-address";
            description
                "A RIPv2 RIP 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;
  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.";
  }
}
} // rip
} // augment

/*
 * 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 instance-name {
      type leafref {
        path "/rt:routing/rt:routing-instance/"
          + "rt:routing-protocols/rt:routing-protocol/rt:name";
      }
      description
        "Instance name identifying the specific RIP Instance.";
    }
  }
} // rcp clear-rip-route
}
<CODE ENDS>
```

[4.](#) 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](#)]:

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

[5.](#) Security Considerations

The configuration, state, and action data defined in this document are designed to be accessed via the NETCONF protocol [[RFC6241](#)]. The data-model by itself does not create any security implications. The security considerations for the NETCONF protocol are applicable. The NETCONF protocol used for sending the data supports authentication and encryption.

[6.](#) References

[6.1.](#) Normative References

- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), October 2010.
- [RFC6991] Schoenwaelder, J., "Common YANG Data Types", [RFC 6991](#), July 2013.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", [RFC 6241](#), June 2011.
- [RFC2453] G. Malkin, "RIP Version 2", [RFC2453](#), November 1998.
- [RFC2080] G. Malkin and R. Minnear, "RIPng for IPv6", [RFC2080](#), January 1997.
- [RFC1724] G. Malkin and F. Baker, "RIP Version 2 MIB Extension", [RFC1724](#), November 1994.

[6.2.](#) Informative References

- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", [RFC 6087](#), January 2011.

Internet-Draft

[draft-ietf-rtgwg-yang-rip-00.txt](#)

August 2015

Authors' Addresses

Xufeng Liu
Ericsson
1595 Spring Hill Road, Suite 500
Vienna, VA 22182
USA

Email: xufeng.liu@ericsson.com

Prateek Sarda
Ericsson India Global Services Pvt. Ltd.
Fern Icon, Survey No 28 and 36/5, Doddanakundi Village
Bangalore, Karnataka 560037
India

Email: prateek.sarda@ericsson.com

Vikram Choudhary
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560037
India

Email: vikram.choudhary@huawei.com

Liu

Expires February 3, 2016

[Page 30]