NETMOD
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2014

L. Lhotka
CZ.NIC

October 18, 2013

A YANG Data Model for Routing Management
draft-ietf-netmod-routing-cfg-11

Abstract

This document contains a specification of three YANG modules. Together they form the core routing data model which serves as a framework for configuring and managing a routing subsystem. It is expected that these modules will be augmented by additional YANG modules defining data models for individual routing protocols and other related functions. The core routing data model provides common building blocks for such extensions - routing instances, routes, routing information bases (RIB), routing protocols and route filters.

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 April 21, 2014.

Table of Contents

1. Introduction

This document contains a specification of the following YANG modules:

- Module "ietf-routing" provides generic components of a routing data model.

- Module "ietf-ipv4-unicast-routing" augments the "ietf-routing" module with additional data specific to IPv4 unicast.

- Module "ietf-ipv6-unicast-routing" augments the "ietf-routing" module with additional data specific to IPv6 unicast, including the router configuration variables required by [RFC4861].

These modules together define the so-called core routing data model, which is proposed as a basis for the development of data models for configuration and management of more sophisticated routing systems. While these three modules can be directly used for simple IP devices with static routing, their main purpose is to provide essential building blocks for more complicated setups involving multiple routing protocols, multicast routing, additional address families, and advanced functions such as route filtering or policy routing. To this end, it is expected that the core routing data model will be augmented by numerous modules developed by other IETF working groups.

2. Terminology and Notation

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 [RFC6241]:

- client
- message
- protocol operation
- server

The following terms are defined in [RFC6020]:

- augment
- configuration data
- data model
- data node
- feature
- mandatory node
- module
- state data
- RPC operation

2.1. Glossary of New Terms

active route: a route that is actually used for sending packets. If there are multiple candidate routes with a matching destination prefix, then it is up to the routing algorithm to select the active route (or several active routes in the case of multi-path routing).

core routing data model: YANG data model resulting from the combination of "ietf-routing", "ietf-ipv4-unicast-routing" and "ietf-ipv6-unicast-routing" modules.

direct route: a route to a directly connected network.

routing information base (RIB): An object containing routes together with other information. See Section 5.3 for details.

system-controlled entry: An entry of a list in operational state data ("config false") that is created by the system independently of what has been explicitly configured. See Section 4.1 for details.

user-controlled entry: An entry of a list in operational state data ("config false") that is created and deleted as a direct consequence of certain configuration changes. See Section 4.1 for details.

2.2. Tree Diagrams

A simplified graphical representation of the complete data tree is presented in Appendix A, and similar diagrams of its various subtrees appear in the main text. 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" state data (read-only).
- Symbols after data node names: "?" means an optional node and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2.3. Prefixes in Data Node Names

In this document, names of data nodes, RPC methods 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
if	ietf-interfaces	[YANG-IF]
ip	ietf-ip	[YANG-IP]
rt	ietf-routing	Section 7
v4ur	ietf-ipv4-unicast-routing	Section 8
v6ur	ietf-ipv6-unicast-routing	Section 9
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

3. Objectives

The initial design of the core routing data model was driven by the following objectives:

- The data model should be suitable for the common address families, in particular IPv4 and IPv6, and for unicast and multicast routing, as well as Multiprotocol Label Switching (MPLS).

- Simple routing setups, such as static routing, should be configurable in a simple way, ideally without any need to develop additional YANG modules.

- On the other hand, the core routing framework must allow for complicated setups involving multiple routing information bases (RIB) and multiple routing protocols, as well as controlled redistributions of routing information.

- Device vendors will want to map the data models built on this generic framework to their proprietary data models and configuration interfaces. Therefore, the framework should be flexible enough to facilitate such a mapping and accommodate data models with different logic.

4. The Design of the Core Routing Data Model

The core routing data model consists of three YANG modules. The first module, "ietf-routing", defines the generic components of a routing system. The other two modules, "ietf-ipv4-unicast-routing" and "ietf-ipv6-unicast-routing", augment the "ietf-routing" module with additional data nodes that are needed for IPv4 and IPv6 unicast routing, respectively. Figures 1 and 2 show abridged views of the configuration and operational state data hierarchies. See Appendix A for the complete data trees.

```
   +--rw routing
      +--rw routing-instance* [name]
      |  +--rw name
      |  +--rw routing-instance-id?
      |  +--rw type?
      |  +--rw enabled?
      |  +--rw router-id?
      |  +--rw description?
      |  +--rw default-ribs
      |  |  +--rw default-rib* [address-family]
      |  |     +--rw address-family
      |  |     +--rw name
      |  +--rw interfaces
      |  |  +--rw interface* [name]
      |  |     +--rw name
      |  |     +--rw v6ur:ipv6-router-advertisements
      |  |           ...
      |  +--rw routing-protocols
      |     +--rw routing-protocol* [name]
      |        +--rw name
      |        +--rw description?
      |        +--rw enabled?
      |        +--rw type
      |        +--rw connected-ribs
      |        |     ...
      |        +--rw static-routes
      |              ...
      +--rw ribs
      |  +--rw rib* [name]
      |     +--rw name
      |     +--rw id?
      |     +--rw address-family
      |     +--rw description?
      |     +--rw recipient-ribs
      |        +--rw recipient-rib* [rib-name]
      |              ...
      +--rw route-filters
         +--rw route-filter* [name]
            +--rw name
            +--rw description?
            +--rw type
```

Figure 1: Configuration data hierarchy.

```
+--ro routing-state
   +--ro routing-instance* [id]
   |  +--ro id
   |  +--ro name?
   |  +--ro type?
   |  +--ro router-id?
   |  +--ro default-ribs
   |  |  +--ro default-rib* [address-family]
   |  |     +--ro address-family
   |  |     +--ro rib-id
   |  +--ro interfaces
   |  |  +--ro interface* [name]
   |  |     +--ro name
   |  |     +--ro v6ur:ipv6-router-advertisements
   |  |           ...
   |  +--ro routing-protocols
   |     +--ro routing-protocol* [name]
   |        +--ro name
   |        +--ro type
   |        +--ro connected-ribs
   |              ...
   +--ro ribs
   |  +--ro rib* [id]
   |     +--ro id
   |     +--ro name?
   |     +--ro address-family
   |     +--ro routes
   |     |  +--ro route* [id]
   |     |        ...
   |     +--ro recipient-ribs
   |        +--ro recipient-rib* [rib-id]
   |              ...
   +--ro route-filters
      +--ro route-filter* [name]
         +--ro name
         +--ro type
```

Figure 2: Operational state data hierarchy.

As can be seen from Figures 1 and 2, the core routing data model introduces several generic components of a routing framework: routing instances, RIBs containing lists of routes, routing protocols and route filters. The following subsections describe these components in more detail.

By combining the components in various ways, and possibly augmenting them with appropriate contents defined in other modules, various routing systems can be realized.

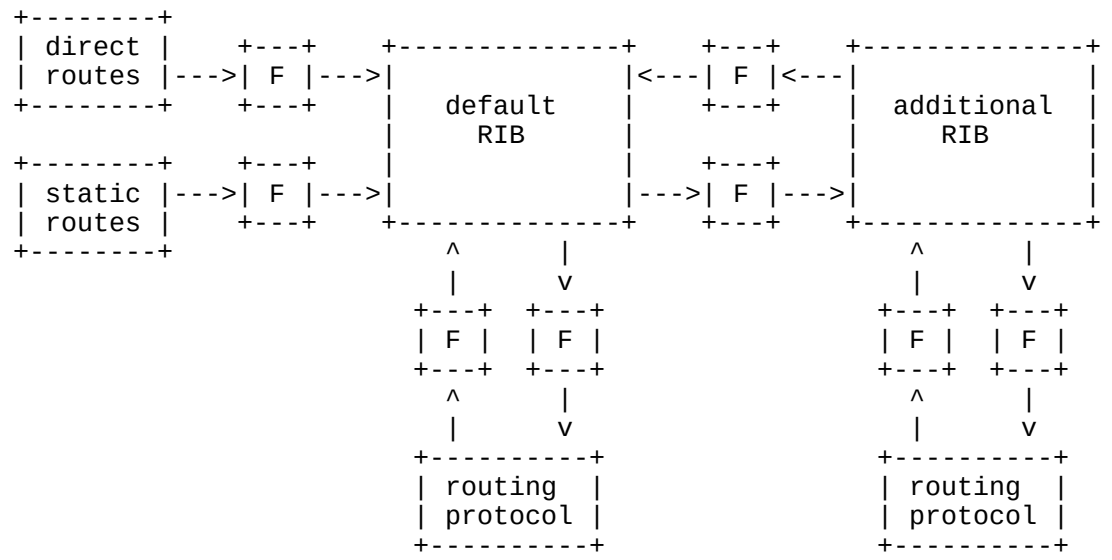

Figure 3: Example setup of a routing system

The example in Figure 3 shows a typical (though certainly not the only possible) organization of a more complex routing subsystem for a single address family. Several of its features are worth mentioning:

- Along with the default RIB, which is always present, an additional RIB is configured.
- Each routing protocol instance, including the "static" and "direct" pseudo-protocols, is connected to exactly one RIB with which it can exchange routes (in both directions, except for the "static" and "direct" pseudo-protocols).
- RIBs may also be connected to each other and exchange routes in either direction (or both).
- Route exchanges along all connections may be controlled by means of route filters, denoted by "F" in Figure 3.

4.1. System-Controlled and User-Controlled List Entries

The core routing data model defines several lists, for example "rt: routing-instance" or "rt:rib", that have to be populated with at least one entry in any properly functioning device, and additional entries may be configured by the user.

In such a list, the server creates the required item as a so-called

system-controlled entry in operational state data, i.e., inside the "rt:routing-state" container.

Additional entries may be created in the configuration by the user via the NETCONF protocol. These are the so-called user-controlled entries. If the server accepts a configured user-controlled entry, then this entry also appears in the operational state version of the list.

Each version of the list (in operational state data and configuration) uses its own set of list keys. In operational state, the keys are unique numeric identifiers assigned by the server. In configuration, the list keys are selected by the user.

The user may also provide supplemental configuration of system-controlled entries. To do so, the user creates a new entry in the configuration with an arbitrary key and desired configuration contents. In order to bind this entry with the corresponding entry in the operational state list, the user writes the operational state key as a value of a special leaf that is defined in the data model for this purpose.

An example can be seen in Appendix C: the "/routing-state/routing-instance" list has a single system-controlled entry whose "id" key has the value "1415926535". This entry is configured by the "/routing/routing-instance" entry whose "name" key is "rtr0". The binding with the operational state entry is established through the value of the leaf "routing-instance-id".

Deleting a user-controlled entry from the configuration list results in the removal of the corresponding entry in the operational state list. In contrast, if a system-controlled entry is deleted from the configuration list, only the extra configuration specified in that entry is removed but the corresponding operational state entry remains in the list.

4.2. Simple versus Advanced Routers

The core routing data model attempts to address devices with elementary routing functions as well as advanced routers. For simple devices, some parts and options of the data model are not needed and represent unnecessary complications for the implementation. Therefore, the core routing data model makes the advanced functionality optional by means of a feature "advanced-router".

Specifically, the following objects and options are supported only in devices that advertise the "advanced-router" feature:

- multiple RIBs per address family, and user-controlled RIB entries in particular,
- routing protocols connected to non-default RIBs,
- RIBs configured as receivers of routes from other RIBs,
- routes with multiple nexthops.

See the "ietf-routing" module for details.

5. Basic Building Blocks

This section describes the essential components of the core routing data model.

5.1. Routing Instance

Each routing instance in the core routing data model represents a logical router. The exact semantics of this term are left to implementations. For example, routing instances may be completely isolated virtual routers or, alternatively, they may internally share certain information.

A routing instance together with its operational status is represented as an entry of the list "/routing-state/routing-instance", and identified by a unique numeric identifier. Configuration of that router instance appears as entry of the list "/routing/routing-instance" whose key is a routing instance name selected by the client.

An implementation MAY support multiple types of logical routers simultaneously. Instances of all routing instance types are organized as entries of the same flat "routing-instance" list. In order to discriminate routing instances belonging to different types, the "type" leaf is defined as a child of the "routing-instance" node.

An implementation MAY create one or more system-controlled routing instances, and MAY also pose restrictions on allowed routing instance types and on the number of supported instances for each type. For example, a simple router implementation may support only one system-controlled routing instance of the default type "rt:standard-routing-instance" and may not allow creation of any user-controlled instances.

Each network layer interface has to be assigned to one or more routing instances in order to be able to participate in packet forwarding, routing protocols and other operations of those routing instances. The assignment is accomplished by placing a corresponding (system- or user-controlled) entry in the list of routing instance interfaces ("rt:interface"). The key of the list entry is the name of a configured network layer interface, see the "ietf-interfaces" module [YANG-IF].

In YANG terms, the list of routing instance interfaces is modeled as the "list" node rather than "leaf-list" in order to allow for adding, via augmentation, other configuration or state data related to the corresponding interface.

Implementations MAY specify additional rules for the assignment of interfaces to routing instances. For example, it may be required that the sets of interfaces assigned to different routing instances be disjoint.

5.1.1. Parameters of IPv6 Routing Instance Interfaces

The module "ietf-ipv6-unicast-routing" augments the definition of the data node "rt:interface", in both configuration and operational state data, with definitions of the following variables as required by [RFC4861], sec. 6.2.1:

- send-advertisements,
- max-rtr-adv-interval,
- min-rtr-adv-interval,
- managed-flag,
- other-config-flag,
- link-mtu,
- reachable-time,
- retrans-timer,
- cur-hop-limit,
- default-lifetime,
- prefix-list: a list of prefixes to be advertised.

  The following parameters are associated with each prefix in the list:

  * valid-lifetime,
  * on-link-flag,
  * preferred-lifetime,
  * autonomous-flag.

The definitions and descriptions of the above parameters can be found in the text of the module "ietf-ipv6-unicast-routing" (Section 9).

NOTES:

1. The "IsRouter" flag, which is also required by [RFC4861], is implemented in the "ietf-ip" module [YANG-IP] (leaf "ip: forwarding").

2. The original specification [RFC4861] allows the implementations to decide whether the "valid-lifetime" and "preferred-lifetime" parameters remain the same in consecutive advertisements, or decrement in real time. However, the latter behavior seems problematic because the values might be reset again to the (higher) configured values after a configuration is reloaded. Moreover, no implementation is known to use the decrementing behavior. The "ietf-ipv6-unicast-routing" module therefore assumes the former behavior with constant values.

5.2. Route

Routes are basic elements of information in a routing system. The core routing data model defines only the following minimal set of route attributes:

- destination prefix: IP prefix specifying the set of destination addresses for which the route may be used. This attribute is mandatory.

- next hop or action: outgoing interface, IP address of one or more adjacent routers to which a packet should be forwarded, or other special action.

The above list of route attributes suffices for a simple static routing configuration. It is expected that future modules defining routing protocols will add other route attributes such as metrics or preferences.

Routes and their attributes are used both in configuration data, for example as manually configured static routes, and in operational state data, for example as entries in RIBs.

5.3. Routing Information Base (RIB)

A routing information base (RIB) is a list of routes complemented with administrative data, namely:

- "source-protocol": type of the routing protocol from which the route was originally obtained.

- o "last-updated": the date and time when the route was last updated, or inserted into the RIB.

Each RIB MUST contain only routes of the same address family. In the data model, address family is represented with an identity derived from the "rt:address-family" base identity.

In the core routing data model, RIBs are operational state data represented as entries of the list "/routing-state/ribs/rib". The contents of RIBs are controlled and manipulated by routing protocol operations which may result in route additions, removals and modifications. This also includes manipulations via the "static" and/or "direct" pseudo-protocols, see Section 5.4.1.

RIBs are global, which means that a RIB may be used by any or all routing instances. However, an implementation MAY specify rules and restrictions for sharing RIBs among routing instances.

Each routing instance must have, for every supported address family, one RIB selected as the so-called default RIB. This selection is recorded in the list "default-rib". The role of default RIBs is explained in Section 5.4.

Simple router implementations that do not advertise the feature "advanced-router" will typically create one system-controlled RIB per supported address family, and declare it as a default RIB (via a system-controlled entry of the "default-rib" list).

5.3.1. Multiple RIBs per Address Family

More complex router implementations advertising the "advanced-router" feature support multiple RIBs per address family that can be used for policy routing and other purposes. Every RIB can then serve as a source of routes for other RIBs of the same address family. To achieve this, one or more recipient RIBs may be specified in the configuration of the source RIB. Optionally, a route filter may be configured for any or all recipient RIBs. Such a route filter then selects and/or manipulates the routes that are passed between the source and recipient RIB.

A RIB MUST NOT appear among its own recipient RIBs.

5.4. Routing Protocol

The core routing data model provides an open-ended framework for defining multiple routing protocol instances within a routing instance. Each routing protocol instance MUST be assigned a type, which is an identity derived from the "rt:routing-protocol" base

identity. The core routing data model defines two identities for the direct and static pseudo-protocols (Section 5.4.1).

Each routing protocol instance is connected to exactly one RIB for each address family that the routing protocol instance supports. Routes learned from the network by a routing protocol are normally installed into the connected RIB(s) and, conversely, routes from the connected RIB(s) are normally injected into the routing protocol. However, routing protocol implementations MAY specify rules that restrict this exchange of routes in either direction (or both directions).

On devices supporting the "advanced-router" feature, any RIB (system-controlled or user-controlled) may be connected to a routing protocol instance by configuring a corresponding entry in the "connected-rib" list. If such an entry is not configured for an address family, then the default RIB MUST be used as the connected RIB for this address family.

In addition, two independent route filters (see Section 5.5) may be configured for each connected RIB to apply user-defined policies controlling the exchange of routes in both directions between the routing protocol instance and the connected RIB:

- import filter controls which routes are passed from the routing protocol instance to the connected RIB,

- export filter controls which routes the routing protocol instance receives from the connected RIB.

Note that the terms import and export are used from the viewpoint of a RIB.

5.4.1. Routing Pseudo-Protocols

The core routing data model defines two special routing protocol types - "direct" and "static". Both are in fact pseudo-protocols, which means that they are confined to the local device and do not exchange any routing information with neighboring routers. Routes from both "direct" and "static" protocol instances are passed to the connected RIB (subject to route filters, if any), but an exchange in the opposite direction is not allowed.

Every routing instance MUST implement exactly one instance of the "direct" pseudo-protocol type. The name of this instance MUST also be "direct". It is the source of direct routes for all configured address families. Direct routes are normally supplied by the operating system kernel, based on the configuration of network

interface addresses, see Section 6.2. The "direct" pseudo-protocol MUST always be connected to the default RIBs of all supported address families. Unlike other routing protocol types, this connection cannot be changed in the configuration. Direct routes MAY be filtered before they appear in the default RIB.

A pseudo-protocol of the type "static" allows for specifying routes manually. It MAY be configured in zero or multiple instances, although a typical configuration will have exactly one instance per routing instance.

Static routes are configured within the "static-routes" container, see Figure 4.

```
   +--rw static-routes
      +--rw v4ur:ipv4
      |  +--rw v4ur:route* [id]
      |     +--rw v4ur:id
      |     +--rw v4ur:description?
      |     +--rw v4ur:destination-prefix
      |     +--rw (nexthop-options)
      |        +--:(special-nexthop)
      |        |  +--rw v4ur:special-nexthop?
      |        +--:(simple-nexthop)
      |        |  +--rw v4ur:gateway?
      |        |  +--rw v4ur:outgoing-interface?
      |        +--:(nexthop-list) {rt:advanced-router}?
      |           +--rw v4ur:nexthop* [id]
      |              +--rw v4ur:id
      |              +--rw v4ur:address?
      |              +--rw v4ur:outgoing-interface?
      |              +--rw v4ur:priority?
      |              +--rw v4ur:weight?
      +--rw v6ur:ipv6
         +--rw v6ur:route* [id]
            +--rw v6ur:id
            +--rw v6ur:description?
            +--rw v6ur:destination-prefix
            +--rw (nexthop-options)
               +--:(special-nexthop)
               |  +--rw v6ur:special-nexthop?
               +--:(simple-nexthop)
               |  +--rw v6ur:gateway?
               |  +--rw v6ur:outgoing-interface?
               +--:(nexthop-list) {rt:advanced-router}?
                  +--rw v6ur:nexthop* [id]
                     +--rw v6ur:id
                     +--rw v6ur:address?
                     +--rw v6ur:outgoing-interface?
                     +--rw v6ur:priority?
                     +--rw v6ur:weight?
```

Figure 4: Structure of "static-routes" subtree.

5.4.2. Defining New Routing Protocols

It is expected that future YANG modules will create data models for additional routing protocol types. Such a new module has to define the protocol-specific configuration and state data, and it has to fit it into the core routing framework in the following way:

- o A new identity MUST be defined for the routing protocol and its base identity MUST be set to "rt:routing-protocol", or to an identity derived from "rt:routing-protocol".

- o Additional route attributes MAY be defined, preferably in one place by means of defining a YANG grouping. The new attributes have to be inserted as state data by augmenting the definitions of the nodes

      /rt:ribs/rt:rib/rt:route

  and

      /rt:active-route/rt:output/rt:route,

  and possibly other places in the configuration, state data and RPC input or output.

- o Configuration parameters and/or state data for the new protocol can be defined by augmenting the "routing-protocol" data node under both "/routing" and "/routing-state".

- o Per-interface configuration, including activation of the routing protocol on individual interfaces, can use references to entries in the list of routing instance interfaces (rt:interface).

By using the "when" statement, the augmented configuration parameters and state data specific to the new protocol SHOULD be made conditional and valid only if the value of "rt:type" or "rt:source-protocol" is equal to the new protocol's identity. It is also RECOMMENDED that the protocol-specific data be encapsulated in appropriately named containers.

The above steps are implemented by the example YANG module for the RIP routing protocol in Appendix B.

5.5. Route Filter

The core routing data model provides a skeleton for defining route filters that can be used to restrict the set of routes being exchanged between a routing protocol instance and a connected RIB, or between a source and a recipient RIB. Route filters may also manipulate routes, i.e., add, delete, or modify their attributes.

Route filters are global, which means that a configured route filter may be used by any or all routing instances. However, an implementation MAY specify rules and restrictions for sharing route filters among routing instances.

By itself, the route filtering framework defined in this document allows for applying only two extreme routing policies which are represented by the following pre-defined route filter types:

- "deny-all-route-filter": all routes are blocked,

- "allow-all-route-filter": all routes are permitted.

The latter type is equivalent to no route filter.

It is expected that more comprehensive route filtering frameworks will be developed separately.

Each route filter is identified by a unique name. Its type MUST be specified by the "type" identity reference - this opens the space for multiple route filtering framework implementations.

5.6. RPC Operations

The "ietf-routing" module defines two RPC operations:

- active-route: query the routing system for the active route(s) that are currently used for sending datagrams to a destination host whose address is passed as an input parameter.

- route-count: retrieve the total number of entries in a RIB.

6. Interactions with Other YANG Modules

The semantics of the core routing data model also depend on several configuration parameters that are defined in other YANG modules.

6.1. Module "ietf-interfaces"

The following boolean switch is defined in the "ietf-interfaces" YANG module [YANG-IF]:

/if:interfaces/if:interface/if:enabled

If this switch is set to "false" for a network layer interface, the device MUST behave exactly as if that interface was not assigned to any routing instance at all.

6.2. Module "ietf-ip"

The following boolean switches are defined in the "ietf-ip" YANG module [YANG-IP]:

/if:interfaces/if:interface/ip:ipv4/ip:enabled

If this switch is set to "false" for a network layer interface, then all IPv4 routing functions related to that interface MUST be disabled.

/if:interfaces/if:interface/ip:ipv4/ip:forwarding

If this switch is set to "false" for a network layer interface, then the forwarding of IPv4 datagrams to and from this interface MUST be disabled. However, the interface may participate in other IPv4 routing functions, such as routing protocols.

/if:interfaces/if:interface/ip:ipv6/ip:enabled

If this switch is set to "false" for a network layer interface, then all IPv6 routing functions related to that interface MUST be disabled.

/if:interfaces/if:interface/ip:ipv6/ip:forwarding

If this switch is set to "false" for a network layer interface, then the forwarding of IPv6 datagrams to and from this interface MUST be disabled. However, the interface may participate in other IPv6 routing functions, such as routing protocols.

In addition, the "ietf-ip" module allows for configuring IPv4 and

IPv6 addresses and network prefixes or masks on network layer interfaces. Configuration of these parameters on an enabled interface MUST result in an immediate creation of the corresponding direct route. The destination prefix of this route is set according to the configured IP address and network prefix/mask, and the interface is set as the outgoing interface for that route.

7. Routing YANG Module

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

<CODE BEGINS> file "ietf-routing@2013-10-18.yang"

```
module ietf-routing {

  namespace "urn:ietf:params:xml:ns:yang:ietf-routing";

  prefix "rt";

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

  import ietf-interfaces {
    prefix "if";
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

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

     WG Chair: David Kessens
     <mailto:david.kessens@nsn.com>

     WG Chair: Juergen Schoenwaelder
     <mailto:j.schoenwaelder@jacobs-university.de>

     Editor: Ladislav Lhotka
     <mailto:lhotka@nic.cz>
    ";

  description
    "This YANG module defines essential components for the management
     of a routing subsystem.

     Copyright (c) 2013 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).

       This version of this YANG module is part of RFC XXXX; see the
       RFC itself for full legal notices.
     ";

   revision 2013-10-18 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: A YANG Data Model for Routing Management";
   }

   /* Features */

   feature advanced-router {
     description
       "This feature indicates that the device supports advanced
        routing functions, namely:

        - user-defined RIBs,

        - multi-path routes.

        Devices that do not support this feature MUST provide exactly
        one system-controlled RIB per supported address family. These
        RIBs then appear as entries of the list
        /routing-state/ribs/rib.
       ";
   }

   /* Identities */

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

   identity ipv4 {
     base address-family;
     description
       "This identity represents IPv4 address family.";
   }
```

```
     identity ipv6 {
       base address-family;
       description
         "This identity represents IPv6 address family.";
     }

     identity routing-instance-type {
       description
         "Base identity from which identities describing routing
          instance types are derived.

          It is primarily intended for discriminating among different
          types of logical routers or router virtualization.
         ";
     }

     identity standard-routing-instance {
       base routing-instance-type;
       description
         "This identity represents a default routing instance.";
     }

     identity routing-protocol {
       description
         "Base identity from which routing protocol identities are
          derived.";
     }

     identity direct {
       base routing-protocol;
       description
         "Routing pseudo-protocol which provides routes to directly
          connected networks.";
     }

     identity static {
       base routing-protocol;
       description
         "Static routing pseudo-protocol.";
     }

     identity route-filter {
       description
         "Base identity from which all route filters are derived.";
     }

     identity deny-all-route-filter {
       base route-filter;
```

```
      description
        "Route filter that blocks all routes.";
    }

    identity allow-all-route-filter {
      base route-filter;
      description
        "Route filter that permits all routes.";
    }

    /* Type Definitions */

    typedef routing-instance-ref {
      type leafref {
        path "/rt:routing/rt:routing-instance/rt:name";
      }
      description
        "This type is used for leafs that reference a routing instance
         configuration.";
    }

    typedef routing-instance-state-ref {
      type leafref {
        path "/rt:routing-state/rt:routing-instance/rt:id";
      }
      description
        "This type is used for leafs that reference state data of a
         routing instance.";
    }

    typedef rib-ref {
      type leafref {
        path "/rt:routing/rt:ribs/rt:rib/rt:name";
      }
      description
        "This type is used for leafs that reference a RIB
         configuration.";
    }

    typedef rib-state-ref {
      type leafref {
        path "/rt:routing-state/rt:ribs/rt:rib/rt:id";
      }
      description
        "This type is used for leafs that reference a RIB in state
         data.";
    }
```

```
  typedef route-filter-ref {
    type leafref {
      path "/rt:routing/rt:route-filters/rt:route-filter/rt:name";
    }
    description
      "This type is used for leafs that reference a route filter
       configuration.";
  }

  typedef route-filter-state-ref {
    type leafref {
      path "/rt:routing-state/rt:route-filters/rt:route-filter/"
         + "rt:name";
    }
    description
      "This type is used for leafs that reference a route filter in
       state data.";
  }

  /* Groupings */

  grouping address-family {
    description
      "This grouping provides a leaf identifying an address
       family.";
    leaf address-family {
      type identityref {
        base address-family;
      }
      mandatory "true";
      description
        "Address family.";
    }
  }

  grouping router-id {
    description
      "This grouping provides the definition of router ID.";
    leaf router-id {
      type yang:dotted-quad;
      description
        "Router ID - 32-bit number in the form of a dotted quad.";
    }
  }

  grouping outgoing-interface {
    description
      "This grouping defines the outgoing interface for use in
```

```
          nexthops.";
        leaf outgoing-interface {
          type leafref {
            path "/routing-state/routing-instance/interfaces/interface/"
               + "name";
          }
          description
            "Name of the outgoing interface.";
        }
      }

      grouping special-nexthop {
        description
          "This grouping provides the leaf for specifying special nexthop
           options.";
        leaf special-nexthop {
          type enumeration {
            enum blackhole {
              description
                "Silently discard the packet.";
            }
            enum unreachable {
              description
                "Discard the packet and notify the sender with an error
                 message indicating that the destination host is
                 unreachable.";
            }
            enum prohibit {
              description
                "Discard the packet and notify the sender with an error
                 message indicating that the communication is
                 administratively prohibited.";
            }
            enum receive {
              description
                "The packet will be received by the local network
                 device.";
            }
          }
          description
            "Special nexthop options.";
        }
      }

      grouping nexthop-classifiers {
        description
          "This grouping provides two nexthop classifiers.";
        leaf priority {
```

```
        type enumeration {
          enum primary {
            value "1";
            description
              "Primary nexthop.";
          }
          enum backup {
            value "2";
            description
              "Backup nexthop.";
          }
        }
        default "primary";
        description
          "Simple priority for distinguishing between primary and
           backup nexthops.

           Backup nexthops are used if and only if no primary nexthops
           are reachable.
          ";
      }
      leaf weight {
        type uint8;
        must ". = 0 or not(../../nexthop/weight = 0)" {
          error-message "Illegal combination of zero and non-zero "
                      + "nexthop weights.";
          description
            "Nexthop weights must be either all zero (equal
             load-balancing) or all non-zero.";
        }
        default "0";
        description
          "This parameter specifies the weight of the nexthop for load
           balancing. The number specifies the relative fraction of the
           traffic that will use the corresponding nexthop.

           The default value of 0 represents equal load-balancing.

           If both primary and backup nexthops are present, then the
           weights for each priority level are used separately.
          ";
      }
    }

    grouping nexthop-content {
      description
        "Generic parameters of nexthops in routes.";
      choice nexthop-options {
```

```
        mandatory "true";
        description
          "Options for expressing the nexthop in routes.";
        case special-nexthop {
          uses special-nexthop;
        }
        case simple-nexthop {
          uses outgoing-interface;
        }
        case nexthop-list {
          if-feature advanced-router;
          list nexthop {
            key "id";
            description
              "An entry of a nexthop list.";
            leaf id {
              type uint64;
              description
                "A numeric identifier of the entry, assigned by the
                 server.";
            }
            uses outgoing-interface;
            uses nexthop-classifiers;
          }
        }
      }
    }

    grouping route-metadata {
      description
        "Route metadata.";
      leaf source-protocol {
        type identityref {
          base routing-protocol;
        }
        mandatory "true";
        description
          "Type of the routing protocol from which the route
           originated.";
      }
      leaf last-updated {
        type yang:date-and-time;
        description
          "Time stamp of the last modification of the route. If the
           route was never modified, it is the time when the route was
           inserted into the RIB.";
      }
    }
```

```
     /* Operational state data */

     container routing-state {
       config "false";
       description
         "Operational state of the routing subsystem.";
       list routing-instance {
         key "id";
         description
           "Each list entry is a container for operational state data of
            a routing instance.

            An implementation MAY create one or more system-controlled
            instances, other user-controlled instances MAY be created by
            configuration.
           ";
         leaf id {
           type uint64;
           description
             "Unique numeric identifier of the routing instance.";
         }
         leaf name {
           type leafref {
             path "/routing/routing-instance/name";
           }
           description
             "The name of the routing instance assigned in the
              corresponding configuration entry (if any).
             ";
         }
         leaf type {
           type identityref {
             base routing-instance-type;
           }
           default "rt:standard-routing-instance";
           description
             "The routing instance type, primarily intended for
              discriminating among different types of logical routers,
              route virtualization, master-slave arrangements etc.,
              while keeping all routing instances in the same flat list.
             ";
         }
         uses router-id {
           description
             "Global router ID.

              An implementation may choose a value if none is
              configured.
```

```
            Routing protocols MAY override this global parameter.
          ";
        }
        container default-ribs {
          description
            "Default RIBs used by the routing instance.";
          list default-rib {
            key "address-family";
            description
              "Each list entry specifies the default RIB for one
               address family.

               The default RIB is operationally connected to all
               routing protocols for which a connected RIB has not been
               explicitly configured.

               The 'direct' pseudo-protocol is always connected to the
               default RIBs.
              ";
            uses address-family;
            leaf rib-id {
              type rib-state-ref;
              mandatory "true";
              description
                "Name of an existing RIB to be used as the default RIB
                 for the given routing instance and address family.";
            }
          }
        }
        container interfaces {
          description
            "Network layer interfaces belonging to the routing
             instance.";
          list interface {
            key "name";
            description
              "List of network layer interfaces assigned to the routing
               instance.";
            leaf name {
              type if:interface-state-ref;
              description
                "A reference to the name of a configured network layer
                 interface.";
            }
          }
        }
        container routing-protocols {
          description
```

```
        "Container for the list of routing protocol instances.";
      list routing-protocol {
        key "name";
        description
          "Operational state of a routing protocol instance.
          ";
        leaf name {
          type string;
          description
            "The name of the routing protocol instance.";
        }
        leaf type {
          type identityref {
            base routing-protocol;
          }
          mandatory "true";
          description
            "Type of the routing protocol.";
        }
        container connected-ribs {
          if-feature advanced-router;
          description
            "Container for connected RIBs.
            ";
          list connected-rib {
            key "rib-id";
            description
              "List of RIBs to which the routing protocol instance
               is connected (at most one RIB per address family).
              ";
            leaf rib-id {
              type rib-state-ref;
              description
                "Name of an existing RIB.";
            }
            leaf import-filter {
              type route-filter-state-ref;
              description
                "Reference to a route filter that is used for
                 filtering routes passed from this routing protocol
                 instance to the RIB specified by the 'name'
                 sibling node.

                 If this leaf is not present, the behavior is
                 protocol-specific, but typically it means that all
                 routes are accepted.
                ";
            }
```

```
                leaf export-filter {
                  type route-filter-state-ref;
                  description
                    "Reference to a route filter that is used for
                     filtering routes passed from the RIB specified by
                     the 'name' sibling node to this routing protocol
                     instance.

                     If this leaf is not present, the behavior is
                     protocol-specific - typically it means that all
                     routes are accepted.

                     The 'direct' and 'static' pseudo-protocols accept
                     no routes from any RIB.
                    ";
                }
              }
            }
          }
        }
      }
      container ribs {
        description
          "Container for RIBs.";
        list rib {
          key "id";
          description
            "Each entry represents a RIB identified by the 'name' key.
             All routes in a RIB MUST belong to the same address
             family.

             The server MUST create the default RIB for each address
             family, and MAY create other RIBs. Additional RIBs MAY be
             created in the configuration.
            ";
          leaf id {
            type uint64;
            description
              "Unique numeric identifier of the RIB instance.";
          }
          leaf name {
            type leafref {
              path "/routing/ribs/rib/name";
            }
            description
              "The name of the RIB assigned in the corresponding
               configuration entry (if any).";
          }
```

```
          uses address-family;
          container routes {
            description
              "Current contents of the RIB.";
            list route {
              key "id";
              description
                "A RIB route entry. This data node MUST be augmented
                 with information specific for routes of each address
                 family.";
              leaf id {
                type uint64 {
                  range "1..max";
                }
                description
                  "Unique numeric identifier of the route.";
              }
              uses nexthop-content;
              uses route-metadata;
            }
          }
          container recipient-ribs {
            if-feature advanced-router;
            description
              "Container for recipient RIBs.";
            list recipient-rib {
              key "rib-id";
              description
                "List of RIBs that receive routes from this RIB.";
              leaf rib-id {
                type rib-state-ref;
                description
                  "The name of the recipient RIB.";
              }
              leaf filter {
                type route-filter-state-ref;
                description
                  "A route filter which is applied to the routes passed
                   to the recipient RIB.";
              }
            }
          }
        }
      }
      container route-filters {
        description
          "Container for route filters.";
        list route-filter {
```

```
          key "name";
          description
            "Route filters are used for filtering and/or manipulating
             routes that are passed between a routing protocol and a
             RIB and vice versa, or between two RIBs.

             It is expected that other modules augment this list with
             contents specific for a particular route filter type.
            ";
          leaf name {
            type string;
            description
              "The name of the route filter.";
          }
          leaf type {
            type identityref {
              base route-filter;
            }
            mandatory "true";
            description
              "Type of the route-filter - an identity derived from the
               'route-filter' base identity.";
          }
        }
      }
    }

    /* Configuration Data */

    container routing {
      description
        "Configuration parameters for the routing subsystem.";
      list routing-instance {
        key "name";
        unique "routing-instance-id";
        description
          "Configuration of a routing instance.
          ";
        leaf name {
          type string;
          description
            "The name of the configured routing instance.";
        }
        leaf routing-instance-id {
          type uint64;
          description
            "Reference to a system-assigned numeric identifier of the
             routing instance.
```

```
            This leaf is essential for creating new configuration
            entries that refer to existing system-controlled routing
            instances.
          ";
        }
        leaf type {
          type identityref {
            base routing-instance-type;
          }
          default "rt:standard-routing-instance";
          description
            "The type of the routing instance.";
        }
        leaf enabled {
          type boolean;
          default "true";
          description
            "Enable/disable the routing instance.

             If this parameter is false, the parent routing instance is
             disabled and does not appear in operational state data,
             despite any other configuration that might be present.
            ";
        }
        uses router-id {
          description
            "Configuration of the global router ID.";
        }
        leaf description {
          type string;
          description
            "Textual description of the routing instance.";
        }
        container default-ribs {
          if-feature advanced-router;
          description
            "Configuration of the default RIBs used by the routing
             instance.

             The default RIB for an addressed family if by default
             connected to all routing protocol instances supporting
             that address family, and always receives direct routes.
            ";
          list default-rib {
            must "address-family=/routing/ribs/rib[name=current()/"
               + "name]/address-family" {
              error-message "Address family mismatch.";
              description
```

```
                "The entry's address family MUST match that of the
                 referenced RIB.";
            }
            key "address-family";
            description
              "Each list entry configures the default RIB for one
               address family.";
            uses address-family;
            leaf name {
              type string;
              mandatory "true";
              description
                "Name of an existing RIB to be used as the default RIB
                 for the given routing instance and address family.";
            }
          }
        }
        container interfaces {
          description
            "Configuration of the routing instance's interfaces.";
          list interface {
            key "name";
            description
              "List of network layer interfaces assigned to the routing
               instance.";
            leaf name {
              type if:interface-ref;
              description
                "A reference to the name of a configured network layer
                 interface.";
            }
          }
        }
        container routing-protocols {
          description
            "Configuration of routing protocol instances.";
          list routing-protocol {
            key "name";
            description
              "Each entry contains configuration of a routing protocol
               instance.";
            leaf name {
              type string;
              description
                "An arbitrary name of the routing protocol instance.";
            }
            leaf description {
              type string;
```

```
        description
          "Textual description of the routing protocol
           instance.";
      }
      leaf enabled {
        type boolean;
        default "true";
        description
          "Enable/disable the routing protocol instance.

           If this parameter is false, the parent routing
           protocol instance is disabled and does not appear in
           operational state data, despite any other
           configuration that might be present.
          ";
      }
      leaf type {
        type identityref {
          base routing-protocol;
        }
        mandatory "true";
        description
          "Type of the routing protocol - an identity derived
           from the 'routing-protocol' base identity.";
      }
      container connected-ribs {
        if-feature advanced-router;
        description
          "Configuration of connected RIBs.
          ";
        list connected-rib {
          must "not(/routing/ribs/rib[name=current()/"
             + "preceding-sibling::connected-rib/"
             + "name and address-family=/routing/ribs/"
             + "rib[name=current()/name]/address-family])" {
            error-message
              "Duplicate address family for connected RIBs.";
            description
              "For each address family, there MUST NOT be more
               than one connected RIB.";
          }
          key "rib-name";
          description
            "List of RIBs to which the routing protocol instance
             is connected (at most one RIB per address family).

             If no connected RIB is configured for an address
             family, the routing protocol is connected to the
```

```
                    default RIB for that address family.
                  ";
                leaf rib-name {
                  type rib-ref;
                  must "../../../type != 'rt:direct' or "
                     + "../../../../../default-ribs/ "
                     + "default-rib/name=." {
                    error-message "The 'direct' protocol can be "
                                + "connected only to a default RIB.";
                    description
                      "For the 'direct' pseudo-protocol, the connected
                       RIB must always be a default RIB.";
                  }
                  description
                    "Name of an existing RIB.";
                }
                leaf import-filter {
                  type route-filter-ref;
                  description
                    "Configuration of import filter.";
                }
                leaf export-filter {
                  type route-filter-ref;
                  description
                    "Configuration of export filter.";
                }
              }
            }
            container static-routes {
              when "../type='rt:static'" {
                description
                  "This container is only valid for the 'static'
                   routing protocol.";
              }
              description
                "Configuration of the 'static' pseudo-protocol.

                 Address family specific modules augment this node with
                 their lists of routes.
                ";
            }
          }
        }
      }
      container ribs {
        description
          "Configured RIBs.";
        list rib {
```

```
          key "name";
          unique "id";
          description
            "Each entry represents a configured RIB identified by the
             'name' key.

             Entries having the same key as a system-controlled entry
             of the list /routing-state/ribs/rib are used for
             configuring parameters of that entry. Other entries define
             additional user-controlled RIBs.
            ";
          leaf name {
            type string;
            description
              "The name of the RIB.";
          }
          leaf id {
            type uint64;
            description
              "System-assigned numeric identifier of the RIB instance.

               This leaf is essential for creating new configuration
               entries that refer to existing system-controlled RIBs.
              ";
          }
          uses address-family;
          leaf description {
            type string;
            description
              "Textual description of the RIB.";
          }
          container recipient-ribs {
            if-feature advanced-router;
            description
              "Configuration of recipient RIBs.";
            list recipient-rib {
              must "name != ../../name" {
                error-message
                  "Source and recipient RIBs are identical.";
                description
                  "A RIB MUST NOT appear among its recipient RIBs.";
              }
              must "/routing/ribs/rib[name=current()/name]/"
                 + "address-family=../../address-family" {
                error-message "Address family mismatch.";
                description
                  "Address family of the recipient RIB MUST match that
                   of the source RIB.";
```

```
            }
            key "rib-name";
            description
              "Each entry configures a recipient RIB.";
            leaf rib-name {
              type rib-ref;
              description
                "The name of the recipient RIB.";
            }
            leaf filter {
              type route-filter-ref;
              description
                "A route filter which is applied to the routes passed
                 to the recipient RIB.";
            }
          }
        }
      }
    }
    container route-filters {
      description
        "Configuration of route filters.";
      list route-filter {
        key "name";
        description
          "Each entry configures a named route filter.";
        leaf name {
          type string;
          description
            "The name of the route filter.";
        }
        leaf description {
          type string;
          description
            "Textual description of the route filter.";
        }
        leaf type {
          type identityref {
            base route-filter;
          }
          mandatory "true";
          description
            "Type of the route filter..";
        }
      }
    }
  }
```

```
     /* RPC methods */

     rpc active-route {
       description
         "Return the active route that a routing-instance uses for
          sending packets to a destination address.
         ";
       input {
         leaf routing-instance-id {
           type routing-instance-state-ref;
           mandatory "true";
           description
             "Identifier of the routing instance whose forwarding
              information base is being queried.

              If the routing instance with 'id' equal to
              'routing-instance-id' doesn't exist, then this operation
              SHALL fail with error-tag 'data-missing' and error-app-tag
              'routing-instance-not-found'.
             ";
         }
         container destination-address {
           description
             "Network layer destination address.

              Address family specific modules MUST augment this
              container with a leaf named 'address'.
             ";
           uses address-family;
         }
       }
       output {
         container route {
           description
             "The active route for the specified destination.

              If the routing instance has no active route for the
              destination address, no output is returned - the server
              SHALL send an <rpc-reply> containing a single element
              <ok>.

              Address family specific modules MUST augment this list
              with appropriate route contents.
             ";
           uses address-family;
           uses nexthop-content;
           uses route-metadata;
         }
```

```
      }
    }

    rpc route-count {
      description
        "Return the current number of routes in a RIB.

         If the RIB with 'id' equal to 'rib-id' doesn't exist, then
         this operation SHALL fail with error-tag 'data-missing' and
         error-app-tag 'rib-not-found'.
        ";
      input {
        leaf rib-id {
          type rib-state-ref;
          mandatory "true";
          description
            "Identifier of the RIB.";
        }
      }
      output {
        leaf number-of-routes {
          type uint64;
          mandatory "true";
          description
            "Number of routes in the RIB.";
        }
      }
    }
  }

  <CODE ENDS>
```

8. IPv4 Unicast Routing YANG Module

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

<CODE BEGINS> file "ietf-ipv4-unicast-routing@2013-10-18.yang"

```
module ietf-ipv4-unicast-routing {

  namespace "urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing";

  prefix "v4ur";

  import ietf-routing {
    prefix "rt";
  }

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

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

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

     WG Chair: David Kessens
     <mailto:david.kessens@nsn.com>

     WG Chair: Juergen Schoenwaelder
     <mailto:j.schoenwaelder@jacobs-university.de>

     Editor: Ladislav Lhotka
     <mailto:lhotka@nic.cz>
    ";

  description
    "This YANG module augments the 'ietf-routing' module with basic
     configuration and operational state data for IPv4 unicast
     routing.

     Copyright (c) 2013 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).

       This version of this YANG module is part of RFC XXXX; see the
       RFC itself for full legal notices.
     ";

   revision 2013-10-18 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: A YANG Data Model for Routing Management";
   }

   /* Identities */

   identity ipv4-unicast {
     base rt:ipv4;
     description
       "This identity represents the IPv4 unicast address family.";
   }

   /* Operational state data */

   augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
     when "../../rt:address-family = 'v4ur:ipv4-unicast'" {
       description
         "This augment is valid only for IPv4 unicast.";
     }
     description
       "This leaf augments an IPv4 unicast route.";
     leaf destination-prefix {
       type inet:ipv4-prefix;
       description
         "IPv4 destination prefix.";
     }
   }

   augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
         + "rt:nexthop-options/rt:simple-nexthop" {
     when "../../rt:address-family = 'v4ur:ipv4-unicast'" {
       description
         "This augment is valid only for IPv4 unicast.";
     }
     description
```

```
        "This leaf augments the 'simple-nexthop' case of IPv4 unicast
         routes.";
      leaf gateway {
        type inet:ipv4-address;
        description
          "IPv4 address of the gateway.";
      }
    }

    augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
          + "rt:nexthop-options/rt:nexthop-list/rt:nexthop" {
      when "../../../rt:address-family = 'v4ur:ipv4-unicast'" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      description
        "This leaf augments the 'nexthop-list' case of IPv4 unicast
         routes.";
      leaf address {
        type inet:ipv4-address;
        description
          "IPv4 address of the nexthop.";
      }
    }

    /* Configuration data */

    augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
          + "rt:routing-protocol/rt:static-routes" {
      description
        "This augment defines the configuration of the 'static'
         pseudo-protocol with data specific to IPv4 unicast.";
      container ipv4 {
        description
          "Configuration of a 'static' pseudo-protocol instance
           consists of a list of routes.";
        list route {
          key "id";
          ordered-by "user";
          description
            "A user-ordered list of static routes.";
          leaf id {
            type uint32 {
              range "1..max";
            }
            description
              "Unique numeric identifier of the route.
```

```
             This value is unrelated to system-assigned keys of
             routes in RIBs.
           ";
         }
         leaf description {
           type string;
           description
             "Textual description of the route.";
         }
         leaf destination-prefix {
           type inet:ipv4-prefix;
           mandatory "true";
           description
             "IPv4 destination prefix.";
         }
         choice nexthop-options {
           mandatory "true";
           description
             "Options for expressing the nexthop in static routes.";
           case special-nexthop {
             uses rt:special-nexthop;
           }
           case simple-nexthop {
             leaf gateway {
               type inet:ipv4-address;
               description
                 "IPv4 address of the gateway.";
             }
             leaf outgoing-interface {
               type leafref {
                 path "../../../../../../rt:interfaces/rt:interface/"
                    + "rt:name";
               }
               description
                 "Name of the outgoing interface.

                  Only interfaces configured for the parent routing
                  instance can be given.
                 ";
             }
           }
           case nexthop-list {
             if-feature rt:advanced-router;
             list nexthop {
               key "id";
               description
                 "An entry of a nexthop list.";
               leaf id {
```

```
                  type uint32;
                  description
                    "Unique numeric identifier of the entry.

                     This value is unrelated to system-assigned keys of
                     nexthops in RIBs.
                    ";
                }
                leaf address {
                  type inet:ipv4-address;
                  description
                    "IPv4 address of the nexthop.";
                }
                leaf outgoing-interface {
                  type leafref {
                    path "../../../../../../../rt:interfaces/"
                       + "rt:interface/rt:name";
                  }
                  description
                    "Name of the outgoing interface.

                     Only interfaces configured for the parent routing
                     instance can be given.
                    ";
                }
                uses rt:nexthop-classifiers;
              }
            }
          }
        }
      }

      /* RPC methods */

      augment "/rt:active-route/rt:input/rt:destination-address" {
        when "rt:address-family='v4ur:ipv4-unicast'" {
          description
            "This augment is valid only for IPv4 unicast.";
        }
        description
          "This leaf augments the 'rt:destination-address' parameter of
           the 'rt:active-route' operation.";
        leaf address {
          type inet:ipv4-address;
          description
            "IPv4 destination address.";
        }
```

```
  }

  augment "/rt:active-route/rt:output/rt:route" {
    when "rt:address-family='v4ur:ipv4-unicast'" {
      description
        "This augment is valid only for IPv4 unicast.";
    }
    description
      "This leaf augments the reply to the 'rt:active-route'
       operation.";
    leaf destination-prefix {
      type inet:ipv4-prefix;
      description
        "IPv4 destination prefix.";
    }
  }

  augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
        + "rt:simple-nexthop" {
    when "rt:address-family='v4ur:ipv4-unicast'" {
      description
        "This augment is valid only for IPv4 unicast.";
    }
    description
      "This leaf augments the 'simple-nexthop' case in the reply to
       the 'rt:active-route' operation.";
    leaf gateway {
      type inet:ipv4-address;
      description
        "IPv4 address of the gateway.";
    }
  }

  augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
        + "rt:nexthop-list/rt:nexthop" {
    when "../rt:address-family='v4ur:ipv4-unicast'" {
      description
        "This augment is valid only for IPv4 unicast.";
    }
    if-feature rt:advanced-router;
    description
      "This leaf augments the 'nexthop-list' case in the reply to the
       'rt:active-route' operation.";
    leaf address {
      type inet:ipv4-address;
      description
        "IPv4 address of the nexthop.";
    }
```

```
      }
    }
```

<CODE ENDS>

9. IPv6 Unicast Routing YANG Module

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

<CODE BEGINS> file "ietf-ipv6-unicast-routing@2013-10-18.yang"

```
module ietf-ipv6-unicast-routing {

  namespace "urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing";

  prefix "v6ur";

  import ietf-routing {
    prefix "rt";
  }

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix "ip";
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

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

     WG Chair: David Kessens
     <mailto:david.kessens@nsn.com>

     WG Chair: Juergen Schoenwaelder
     <mailto:j.schoenwaelder@jacobs-university.de>

     Editor: Ladislav Lhotka
     <mailto:lhotka@nic.cz>
    ";

  description
```

```
       "This YANG module augments the 'ietf-routing' module with basic
        configuration and operational state data for IPv6 unicast
        routing.

        Copyright (c) 2013 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).

        This version of this YANG module is part of RFC XXXX; see the
        RFC itself for full legal notices.
       ";

     revision 2013-10-18 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for Routing Management";
     }

     /* Identities */

     identity ipv6-unicast {
       base rt:ipv6;
       description
         "This identity represents the IPv6 unicast address family.";
     }

     /* Operational state data */

     augment "/rt:routing-state/rt:routing-instance/rt:interfaces/"
           + "rt:interface" {
       when "/if:interfaces/if:interface[if:name=current()/rt:name]/"
          + "ip:ipv6/ip:enabled='true'" {
         description
           "This augment is only valid for router interfaces with
            enabled IPv6.";
       }
       description
         "IPv6-specific parameters of router interfaces.";
       container ipv6-router-advertisements {
         description
           "Parameters of IPv6 Router Advertisements.";
```

```
          leaf send-advertisements {
            type boolean;
            default "false";
            description
              "A flag indicating whether or not the router sends periodic
               Router Advertisements and responds to Router
               Solicitations.";
            reference
              "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
               AdvSendAdvertisements.";
          }
          leaf max-rtr-adv-interval {
            type uint16 {
              range "4..1800";
            }
            units "seconds";
            default "600";
            description
              "The maximum time allowed between sending unsolicited
               multicast Router Advertisements from the interface.";
            reference
              "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
               MaxRtrAdvInterval.";
          }
          leaf min-rtr-adv-interval {
            type uint16 {
              range "3..1350";
            }
            units "seconds";
            description
              "The minimum time allowed between sending unsolicited
               multicast Router Advertisements from the interface.

               The default value to be used operationally if this leaf is
               not configured is determined as follows:

               - if max-rtr-adv-interval >= 9 seconds, the default value
                 is 0.33 * max-rtr-adv-interval;

               - otherwise it is 0.75 * max-rtr-adv-interval.
              ";
            reference
              "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
               MinRtrAdvInterval.";
          }
          leaf managed-flag {
            type boolean;
            default "false";
```

```
          description
            "The boolean value to be placed in the 'Managed address
             configuration' flag field in the Router Advertisement.";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvManagedFlag.";
        }
        leaf other-config-flag {
          type boolean;
          default "false";
          description
            "The boolean value to be placed in the 'Other
             configuration' flag field in the Router Advertisement.";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvOtherConfigFlag.";
        }
        leaf link-mtu {
          type uint32;
          default "0";
          description
            "The value to be placed in MTU options sent by the router.
             A value of zero indicates that no MTU options are sent.";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvLinkMTU.";
        }
        leaf reachable-time {
          type uint32 {
            range "0..3600000";
          }
          units "milliseconds";
          default "0";
          description
            "The value to be placed in the Reachable Time field in the
             Router Advertisement messages sent by the router. The
             value zero means unspecified (by this router).";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvReachableTime.";
        }
        leaf retrans-timer {
          type uint32;
          units "milliseconds";
          default "0";
          description
            "The value to be placed in the Retrans Timer field in the
             Router Advertisement messages sent by the router. The
```

```
            value zero means unspecified (by this router).";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvRetransTimer.";
        }
        leaf cur-hop-limit {
          type uint8;
          default "64";
          description
            "The default value to be placed in the Cur Hop Limit field
             in the Router Advertisement messages sent by the router.
             The value should be set to the current diameter of the
             Internet. The value zero means unspecified (by this
             router).

             The default SHOULD be set to the value specified in IANA
             Assigned Numbers that was in effect at the time of
             implementation.
            ";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvCurHopLimit.

             IANA: IP Parameters,
             http://www.iana.org/assignments/ip-parameters
            ";
        }
        leaf default-lifetime {
          type uint16 {
            range "0..9000";
          }
          units "seconds";
          description
            "The value to be placed in the Router Lifetime field of
             Router Advertisements sent from the interface, in seconds.
             MUST be either zero or between max-rtr-adv-interval and
             9000 seconds. A value of zero indicates that the router is
             not to be used as a default router. These limits may be
             overridden by specific documents that describe how IPv6
             operates over different link layers.

             If this parameter is not configured, a value of 3 *
             max-rtr-adv-interval SHOULD be used.
            ";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvDefaultLifeTime.";
        }
```

```
        container prefix-list {
          description
            "A list of prefixes that are placed in Prefix Information
             options in Router Advertisement messages sent from the
             interface.

             By default, these are all prefixes that the router
             advertises via routing protocols as being on-link for the
             interface from which the advertisement is sent.

             The link-local prefix SHOULD NOT be included in the list
             of advertised prefixes.
            ";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvPrefixList.";
          list prefix {
            key "prefix-spec";
            description
              "Advertised prefix entry with parameters.";
            leaf prefix-spec {
              type inet:ipv6-prefix;
              description
                "IPv6 address prefix.";
            }
            leaf valid-lifetime {
              type uint32;
              units "seconds";
              default "2592000";
              description
                "The value to be placed in the Valid Lifetime in the
                 Prefix Information option. The designated value of all
                 1's (0xffffffff) represents infinity.
                ";
              reference
                "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
                 AdvValidLifetime.";
            }
            leaf on-link-flag {
              type boolean;
              default "true";
              description
                "The value to be placed in the on-link flag ('L-bit')
                 field in the Prefix Information option.";
              reference
                "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
                 AdvOnLinkFlag.";
            }
```

```
            leaf preferred-lifetime {
              type uint32;
              units "seconds";
              default "604800";
              description
                "The value to be placed in the Preferred Lifetime in
                 the Prefix Information option, in seconds. The
                 designated value of all 1's (0xffffffff) represents
                 infinity.";
              reference
                "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
                 AdvPreferredLifetime.";
            }
            leaf autonomous-flag {
              type boolean;
              default "true";
              description
                "The value to be placed in the Autonomous Flag field in
                 the Prefix Information option.";
              reference
                "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
                 AdvAutonomousFlag.";
            }
          }
        }
      }
    }

    augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
      when "../../rt:address-family = 'v6ur:ipv6-unicast'" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This leaf augments an IPv6 unicast route.";
      leaf destination-prefix {
        type inet:ipv6-prefix;
        description
          "IPv6 destination prefix.";
      }
    }

    augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
          + "rt:nexthop-options/rt:simple-nexthop" {
      when "../../rt:address-family = 'v6ur:ipv6-unicast'" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
```

```
      description
        "This leaf augments the 'simple-nexthop' case of IPv6 unicast
         routes.";
      leaf gateway {
        type inet:ipv6-address;
        description
          "IPv6 address of the gateway.";
      }
    }

    augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
          + "rt:nexthop-options/rt:nexthop-list/rt:nexthop" {
      when "../../../rt:address-family = 'v6ur:ipv6-unicast'" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This leaf augments the 'nexthop-list' case of IPv6 unicast
         routes.";
      leaf address {
        type inet:ipv6-address;
        description
          "IPv6 address of the nexthop.";
      }
    }

    /* Configuration data */

    augment
      "/rt:routing/rt:routing-instance/rt:interfaces/rt:interface" {
      when "/if:interfaces/if:interface[if:name=current()/rt:name]/"
         + "ip:ipv6/ip:enabled='true'" {
        description
          "This augment is only valid for router interfaces with
           enabled IPv6.";
      }
      description
        "Configuration of IPv6-specific parameters of router
         interfaces.";
      container ipv6-router-advertisements {
        description
          "Configuration of IPv6 Router Advertisements.

           See the corresponding parameters under /rt:routing-state for
           detailed descriptions and references.
          ";
        leaf send-advertisements {
          type boolean;
```

```
            default "false";
            description
              "A flag indicating whether or not the router sends periodic
               Router Advertisements and responds to Router
               Solicitations.";
          }
          leaf max-rtr-adv-interval {
            type uint16 {
              range "4..1800";
            }
            units "seconds";
            default "600";
            description
              "The maximum time allowed between sending unsolicited
               multicast Router Advertisements from the interface.";
          }
          leaf min-rtr-adv-interval {
            type uint16 {
              range "3..1350";
            }
            units "seconds";
            must ". <= 0.75 * ../max-rtr-adv-interval" {
              description
                "The value MUST NOT be greater than 75 % of
                 'max-rtr-adv-interval'.";
            }
            description
              "The minimum time allowed between sending unsolicited
               multicast Router Advertisements from the interface.
              ";
          }
          leaf managed-flag {
            type boolean;
            default "false";
            description
              "The boolean value to be placed in the 'Managed address
               configuration' flag field in the Router Advertisement.";
          }
          leaf other-config-flag {
            type boolean;
            default "false";
            description
              "The boolean value to be placed in the 'Other
               configuration' flag field in the Router Advertisement.";
          }
          leaf link-mtu {
            type uint32;
            default "0";
```

```
          description
            "The value to be placed in MTU options sent by the router.
             A value of zero indicates that no MTU options are sent.";
        }
        leaf reachable-time {
          type uint32 {
            range "0..3600000";
          }
          units "milliseconds";
          default "0";
          description
            "The value to be placed in the Reachable Time field in the
             Router Advertisement messages sent by the router. The
             value zero means unspecified (by this router).";
        }
        leaf retrans-timer {
          type uint32;
          units "milliseconds";
          default "0";
          description
            "The value to be placed in the Retrans Timer field in the
             Router Advertisement messages sent by the router. The
             value zero means unspecified (by this router).";
        }
        leaf cur-hop-limit {
          type uint8;
          default "64";
          description
            "The default value to be placed in the Cur Hop Limit field
             in the Router Advertisement messages sent by the router.
            ";
        }
        leaf default-lifetime {
          type uint16 {
            range "0..9000";
          }
          units "seconds";
          description
            "The value to be placed in the Router Lifetime field of
             Router Advertisements sent from the interface, in seconds.
            ";
        }
        container prefix-list {
          description
            "Configuration of prefixes to be placed in Prefix
             Information options in Router Advertisement messages sent
             from the interface.
```

```
           Prefixes that are advertised by default but do not have
           their entries in the child 'prefix' list are advertised
           with the default values of all parameters.
          ";
        list prefix {
          key "prefix-spec";
          description
            "Advertised prefix entry.";
          leaf prefix-spec {
            type inet:ipv6-prefix;
            description
              "IPv6 address prefix.";
          }
          choice control-adv-prefixes {
            default "advertise";
            description
              "The prefix either may be explicitly removed from the
               set of advertised prefixes, or parameters with which
               it is advertised may be specified (default case).";
            leaf no-advertise {
              type empty;
              description
                "The prefix will not be advertised.

                 This can be used for removing the prefix from the
                 default set of advertised prefixes.
                ";
            }
            case advertise {
              leaf valid-lifetime {
                type uint32;
                units "seconds";
                default "2592000";
                description
                  "The value to be placed in the Valid Lifetime in
                   the Prefix Information option.";
              }
              leaf on-link-flag {
                type boolean;
                default "true";
                description
                  "The value to be placed in the on-link flag
                   ('L-bit') field in the Prefix Information
                   option.";
              }
              leaf preferred-lifetime {
                type uint32;
                units "seconds";
```

```
                must ". <= ../valid-lifetime" {
                  description
                    "This value MUST NOT be greater than
                     valid-lifetime.";
                }
                default "604800";
                description
                  "The value to be placed in the Preferred Lifetime
                   in the Prefix Information option.";
              }
              leaf autonomous-flag {
                type boolean;
                default "true";
                description
                  "The value to be placed in the Autonomous Flag
                   field in the Prefix Information option.";
              }
            }
          }
        }
      }
    }

    augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
          + "rt:routing-protocol/rt:static-routes" {
      description
        "This augment defines the configuration of the 'static'
         pseudo-protocol with data specific to IPv6 unicast.";
      container ipv6 {
        description
          "Configuration of a 'static' pseudo-protocol instance
           consists of a list of routes.";
        list route {
          key "id";
          ordered-by "user";
          description
            "A user-ordered list of static routes.";
          leaf id {
            type uint32 {
              range "1..max";
            }
            description
              "Unique numeric identifier of the route.

               This value is unrelated to system-assigned keys of
               routes in RIBs.
              ";
```

```
          }
          leaf description {
            type string;
            description
              "Textual description of the route.";
          }
          leaf destination-prefix {
            type inet:ipv6-prefix;
            mandatory "true";
            description
              "IPv6 destination prefix.";
          }
          choice nexthop-options {
            mandatory "true";
            description
              "Options for expressing the nexthop in static routes.";
            case special-nexthop {
              uses rt:special-nexthop;
            }
            case simple-nexthop {
              leaf gateway {
                type inet:ipv6-address;
                description
                  "IPv6 address of the gateway.";
              }
              leaf outgoing-interface {
                type leafref {
                  path "../../../../../../rt:interfaces/rt:interface/"
                     + "rt:name";
                }
                description
                  "Name of the outgoing interface.

                   Only interfaces configured for the parent routing
                   instance can be given.
                  ";
              }
            }
            case nexthop-list {
              if-feature rt:advanced-router;
              list nexthop {
                key "id";
                description
                  "An entry of a nexthop list.";
                leaf id {
                  type uint32;
                  description
                    "Unique numeric identifier of the entry.
```

```
                    This value is unrelated to system-assigned keys of
                    nexthops in RIBs.
                  ";
              }
              leaf address {
                type inet:ipv6-address;
                description
                  "IPv6 address of the nexthop.";
              }
              leaf outgoing-interface {
                type leafref {
                  path "../../../../../../../rt:interfaces/"
                     + "rt:interface/rt:name";
                }
                description
                  "Name of the outgoing interface.

                   Only interfaces configured for the parent routing
                   instance can be given.
                  ";
              }
              uses rt:nexthop-classifiers;
            }
          }
        }
      }
    }

    /* RPC methods */

    augment "/rt:active-route/rt:input/rt:destination-address" {
      when "rt:address-family='v6ur:ipv6-unicast'" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This leaf augments the 'rt:destination-address' parameter of
         the 'rt:active-route' operation.";
      leaf address {
        type inet:ipv6-address;
        description
          "IPv6 destination address.";
      }
    }

    augment "/rt:active-route/rt:output/rt:route" {
      when "rt:address-family='v6ur:ipv6-unicast'" {
```

```
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This leaf augments the reply to the 'rt:active-route'
         operation.";
      leaf destination-prefix {
        type inet:ipv6-prefix;
        description
          "IPv6 destination prefix.";
      }
    }

    augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
          + "rt:simple-nexthop" {
      when "rt:address-family='v6ur:ipv6-unicast'" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This leaf augments the 'simple-nexthop' case in the reply to
         the 'rt:active-route' operation.";
      leaf gateway {
        type inet:ipv6-address;
        description
          "IPv6 address of the gateway.";
      }
    }

    augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
          + "rt:nexthop-list/rt:nexthop" {
      when "../rt:address-family='v6ur:ipv6-unicast'" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      if-feature rt:advanced-router;
      description
        "This leaf augments the 'nexthop-list' case in the reply to the
         'rt:active-route' operation.";
      leaf address {
        type inet:ipv6-address;
        description
          "IPv6 address of the nexthop.";
      }
    }
  }

  <CODE ENDS>
```

10. 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-routing

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing

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 [RFC6020]:

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

   ------------------------------------------------------------------
   name:         ietf-ipv4-unicast-routing
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing
   prefix:       v4ur
   reference:    RFC XXXX
   ------------------------------------------------------------------

   ------------------------------------------------------------------
   name:         ietf-ipv6-unicast-routing
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing
   prefix:       v6ur
   reference:    RFC XXXX
   ------------------------------------------------------------------
```

11. Security Considerations

Configuration and state data conforming to the core routing data model (defined in this document) are designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

A number of data nodes defined in the YANG modules belonging to the configuration part of the core routing data model 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", can have negative effects on the network if the protocol operations are not properly protected.

The vulnerable "config true" subtrees and data nodes are the following:

/routing/routing-instance/interfaces/interface This list assigns a network layer interface to a routing instance and may also specify interface parameters related to routing.

/routing/routing-instance/routing-protocols/routing-protocol This list specifies the routing protocols configured on a device.

/routing/route-filters/route-filter This list specifies the configured route filters which represent administrative policies for redistributing and modifying routing information.

/routing/ribs/rib This list specifies the RIBs configured for the device.

Unauthorized access to any of these lists 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.

12. Acknowledgments

The author wishes to thank Nitin Bahadur, Martin Bjorklund, Joel Halpern, Wes Hardaker, Sriganesh Kini, Andrew McGregor, Jan Medved, Xiang Li, Thomas Morin, Tom Petch, Bruno Rijsman, Juergen Schoenwaelder, Phil Shafer, Dave Thaler and Yi Yang for their helpful comments and suggestions.

13. References

13.1. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

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

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "NETCONF Configuration Protocol", RFC 6241, June 2011.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013.

[YANG-IF] Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-interfaces-cfg-12 (work in progress), July 2013.

[YANG-IP] Bjorklund, M., "A YANG Data Model for IP Management", draft-ietf-netmod-ip-cfg-10 (work in progress), August 2013.

13.2. Informative References

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

Appendix A. The Complete Data Trees

This appendix presents the complete configuration and operational state data trees of the core routing data model.

See Section 2.2 for an explanation of the symbols used. Data type of every leaf node is shown near the right end of the corresponding line.

A.1. Configuration Data

```
+--rw routing
   +--rw routing-instance* [name]
   |  +--rw name                    string
   |  +--rw routing-instance-id?    uint64
   |  +--rw type?                   identityref
   |  +--rw enabled?                boolean
   |  +--rw router-id?              yang:dotted-quad
   |  +--rw description?            string
   |  +--rw default-ribs {advanced-router}?
   |  |  +--rw default-rib* [address-family]
   |  |     +--rw address-family    identityref
   |  |     +--rw name              string
   |  +--rw interfaces
   |  |  +--rw interface* [name]
   |  |     +--rw name                              if:interface-ref
   |  |     +--rw v6ur:ipv6-router-advertisements
   |  |        +--rw v6ur:send-advertisements?    boolean
   |  |        +--rw v6ur:max-rtr-adv-interval?   uint16
   |  |        +--rw v6ur:min-rtr-adv-interval?   uint16
   |  |        +--rw v6ur:managed-flag?           boolean
   |  |        +--rw v6ur:other-config-flag?      boolean
   |  |        +--rw v6ur:link-mtu?               uint32
   |  |        +--rw v6ur:reachable-time?         uint32
   |  |        +--rw v6ur:retrans-timer?          uint32
   |  |        +--rw v6ur:cur-hop-limit?          uint8
   |  |        +--rw v6ur:default-lifetime?       uint16
   |  |        +--rw v6ur:prefix-list
   |  |           +--rw v6ur:prefix* [prefix-spec]
   |  |              +--rw v6ur:prefix-spec       inet:ipv6-prefix
   |  |              +--rw (control-adv-prefixes)?
   |  |                 +--:(no-advertise)
   |  |                 |  +--rw v6ur:no-advertise?         empty
   |  |                 +--:(advertise)
   |  |                    +--rw v6ur:valid-lifetime?       uint32
   |  |                    +--rw v6ur:on-link-flag?         boolean
   |  |                    +--rw v6ur:preferred-lifetime?   uint32
   |  |                    +--rw v6ur:autonomous-flag?      boolean
```

```
|  +--rw routing-protocols
|     +--rw routing-protocol* [name]
|        +--rw name              string
|        +--rw description?      string
|        +--rw enabled?          boolean
|        +--rw type              identityref
|        +--rw connected-ribs {advanced-router}?
|        |  +--rw connected-rib* [rib-name]
|        |     +--rw rib-name         rib-ref
|        |     +--rw import-filter?   route-filter-ref
|        |     +--rw export-filter?   route-filter-ref
|        +--rw static-routes
|           +--rw v4ur:ipv4
|           |  +--rw v4ur:route* [id]
|           |     +--rw v4ur:id                 uint32
|           |     +--rw v4ur:description?       string
|           |     +--rw v4ur:destination-prefix  inet:ipv4-prefix
|           |     +--rw (nexthop-options)
|           |        +--:(special-nexthop)
|           |        |  +--rw v4ur:special-nexthop?  enumeration
|           |        +--:(simple-nexthop)
|           |        |  +--rw v4ur:gateway?    inet:ipv4-address
|           |        |  +--rw v4ur:outgoing-interface?   leafref
|           |        +--:(nexthop-list) {rt:advanced-router}?
|           |           +--rw v4ur:nexthop* [id]
|           |              +--rw v4ur:id       uint32
|           |              +--rw v4ur:address?  inet:ipv4-address
|           |              +--rw v4ur:outgoing-interface? leafref
|           |              +--rw v4ur:priority?       enumeration
|           |              +--rw v4ur:weight?               uint8
|           +--rw v6ur:ipv6
|              +--rw v6ur:route* [id]
|                 +--rw v6ur:id                    uint32
|                 +--rw v6ur:description?          string
|                 +--rw v6ur:destination-prefix  inet:ipv6-prefix
|                 +--rw (nexthop-options)
|                    +--:(special-nexthop)
|                    |  +--rw v6ur:special-nexthop?   enumeration
|                    +--:(simple-nexthop)
|                    |  +--rw v6ur:gateway?     inet:ipv6-address
|                    |  +--rw v6ur:outgoing-interface?   leafref
|                    +--:(nexthop-list) {rt:advanced-router}?
|                       +--rw v6ur:nexthop* [id]
|                          +--rw v6ur:id                   uint32
|                          +--rw v6ur:address?  inet:ipv6-address
|                          +--rw v6ur:outgoing-interface? leafref
|                          +--rw v6ur:priority?       enumeration
|                          +--rw v6ur:weight?               uint8
```

```
      +--rw ribs
      |  +--rw rib* [name]
      |     +--rw name              string
      |     +--rw id?               uint64
      |     +--rw address-family    identityref
      |     +--rw description?      string
      |     +--rw recipient-ribs {advanced-router}?
      |        +--rw recipient-rib* [rib-name]
      |           +--rw rib-name    rib-ref
      |           +--rw filter?     route-filter-ref
      +--rw route-filters
         +--rw route-filter* [name]
            +--rw name           string
            +--rw description?   string
            +--rw type           identityref
```

A.2. Operational State Data

```
   +--ro routing-state
      +--ro routing-instance* [id]
      |  +--ro id                    uint64
      |  +--ro name?                 leafref
      |  +--ro type?                 identityref
      |  +--ro router-id?            yang:dotted-quad
      |  +--ro default-ribs
      |  |  +--ro default-rib* [address-family]
      |  |     +--ro address-family    identityref
      |  |     +--ro rib-id            rib-state-ref
      |  +--ro interfaces
      |  |  +--ro interface* [name]
      |  |     +--ro name                        if:interface-state-ref
      |  |     +--ro v6ur:ipv6-router-advertisements
      |  |        +--ro v6ur:send-advertisements?    boolean
      |  |        +--ro v6ur:max-rtr-adv-interval?   uint16
      |  |        +--ro v6ur:min-rtr-adv-interval?   uint16
      |  |        +--ro v6ur:managed-flag?           boolean
      |  |        +--ro v6ur:other-config-flag?      boolean
      |  |        +--ro v6ur:link-mtu?               uint32
      |  |        +--ro v6ur:reachable-time?         uint32
      |  |        +--ro v6ur:retrans-timer?          uint32
      |  |        +--ro v6ur:cur-hop-limit?          uint8
      |  |        +--ro v6ur:default-lifetime?       uint16
      |  |        +--ro v6ur:prefix-list
      |  |           +--ro v6ur:prefix* [prefix-spec]
      |  |              +--ro v6ur:prefix-spec       inet:ipv6-prefix
      |  |              +--ro v6ur:valid-lifetime?       uint32
      |  |              +--ro v6ur:on-link-flag?         boolean
      |  |              +--ro v6ur:preferred-lifetime?   uint32
```

```
|  |              +--ro v6ur:autonomous-flag?      boolean
|  +--ro routing-protocols
|     +--ro routing-protocol* [name]
|        +--ro name              string
|        +--ro type              identityref
|        +--ro connected-ribs {advanced-router}?
|           +--ro connected-rib* [rib-id]
|              +--ro rib-id           rib-state-ref
|              +--ro import-filter?   route-filter-state-ref
|              +--ro export-filter?   route-filter-state-ref
+--ro ribs
|  +--ro rib* [id]
|     +--ro id               uint64
|     +--ro name?            leafref
|     +--ro address-family   identityref
|     +--ro routes
|     |  +--ro route* [id]
|     |     +--ro id                          uint64
|     |     +--ro (nexthop-options)
|     |     |  +--:(special-nexthop)
|     |     |  |  +--ro special-nexthop?           enumeration
|     |     |  +--:(simple-nexthop)
|     |     |  |  +--ro outgoing-interface?    leafref
|     |     |  |  +--ro v4ur:gateway?          inet:ipv4-address
|     |     |  |  +--ro v6ur:gateway?          inet:ipv6-address
|     |     |  +--:(nexthop-list) {advanced-router}?
|     |     |     +--ro nexthop* [id]
|     |     |        +--ro id                     uint64
|     |     |        +--ro outgoing-interface?    leafref
|     |     |        +--ro priority?              enumeration
|     |     |        +--ro weight?                uint8
|     |     |        +--ro v4ur:address?        inet:ipv4-address
|     |     |        +--ro v6ur:address?        inet:ipv6-address
|     |     +--ro source-protocol             identityref
|     |     +--ro last-updated?               yang:date-and-time
|     |     +--ro v4ur:destination-prefix?    inet:ipv4-prefix
|     |     +--ro v6ur:destination-prefix?    inet:ipv6-prefix
|     +--ro recipient-ribs {advanced-router}?
|        +--ro recipient-rib* [rib-id]
|           +--ro rib-id    rib-state-ref
|           +--ro filter?   route-filter-state-ref
+--ro route-filters
   +--ro route-filter* [name]
      +--ro name    string
      +--ro type    identityref
```

Appendix B. Example: Adding a New Routing Protocol

This appendix demonstrates how the core routing data model can be extended to support a new routing protocol. The YANG module "example-rip" shown below is intended only as an illustration rather than a real definition of a data model for the RIP routing protocol. For the sake of brevity, we do not follow all the guidelines specified in [RFC6087]. See also Section 5.4.2.

```
module example-rip {

  namespace "http://example.com/rip";

  prefix "rip";

  import ietf-routing {
    prefix "rt";
  }

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

  typedef rip-metric {
    type uint8 {
      range "0..16";
    }
  }

  grouping route-content {
    description
      "This grouping defines RIP-specific route attributes.";
    leaf metric {
      type rip-metric;
    }
    leaf tag {
      type uint16;
      default "0";
      description
        "This leaf may be used to carry additional info, e.g. AS
         number.";
    }
  }

  augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
    when "rt:source-protocol = 'rip:rip'" {
```

```
        description
          "This augment is only valid for a routes whose source
           protocol is RIP.";
      }
      description
        "RIP-specific route attributes.";
      uses route-content;
    }

    augment "/rt:active-route/rt:output/rt:route" {
      description
        "RIP-specific route attributes in the output of 'active-route'
         RPC.";
      uses route-content;
    }

    augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
          + "rt:routing-protocol" {
      when "rt:type = 'rip:rip'" {
        description
          "This augment is only valid for a routing protocol instance
           of type 'rip'.";
      }
      container rip {
        description
          "RIP instance configuration.";
        container interfaces {
          description
            "Per-interface RIP configuration.";
          list interface {
            key "name";
            description
              "RIP is enabled on interfaces that have an entry in this
               list, unless 'enabled' is set to 'false' for that
               entry.";
            leaf name {
              type leafref {
                path "../../../../../../rt:interfaces/rt:interface/"
                   + "rt:name";
              }
            }
            leaf enabled {
              type boolean;
              default "true";
            }
            leaf metric {
              type rip-metric;
              default "1";
```

```
            }
          }
        }
        leaf update-interval {
          type uint8 {
            range "10..60";
          }
          units "seconds";
          default "30";
          description
            "Time interval between periodic updates.";
        }
      }
    }
  }
```

Appendix C. Example: NETCONF <get> Reply

This section contains a sample reply to the NETCONF <get> message, which could be sent by a server supporting (i.e., advertising them in the NETCONF <hello> message) the following YANG modules:

- ietf-interfaces [YANG-IF],
- ietf-ip [YANG-IP],
- ietf-routing (Section 7),
- ietf-ipv4-unicast-routing (Section 8),
- ietf-ipv6-unicast-routing (Section 9).

We assume a simple network setup as shown in Figure 5: router "A" uses static default routes with the "ISP" router as the next hop. IPv6 router advertisements are configured only on the "eth1" interface and disabled on the upstream "eth0" interface.

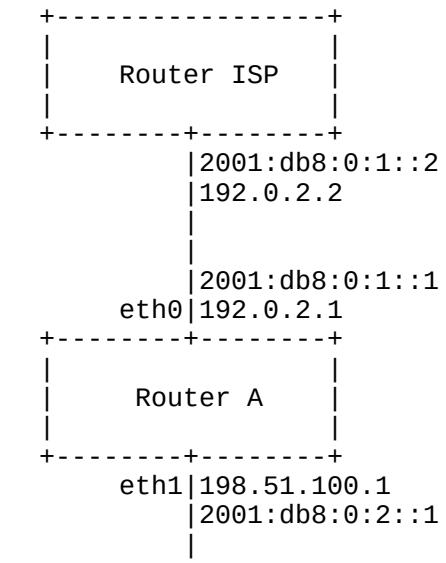

Figure 5: Example network configuration

A reply to the NETCONF <get> message sent by router "A" would then be as follows:

```
<?xml version="1.0"?>
<rpc-reply
```

```
     message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:v4ur="urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing"
     xmlns:v6ur="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing"
     xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
     xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip"
     xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing">
    <data>
     <if:interfaces>
      <if:interface>
       <if:name>eth0</if:name>
       <if:type>ethernetCsmacd</if:type>
       <if:description>
         Uplink to ISP.
       </if:description>
       <ip:ipv4>
        <ip:address>
         <ip:ip>192.0.2.1</ip:ip>
         <ip:prefix-length>24</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
       </ip:ipv4>
       <ip:ipv6>
        <ip:address>
         <ip:ip>2001:0db8:0:1::1</ip:ip>
         <ip:prefix-length>64</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
        <ip:autoconf>
         <ip:create-global-addresses>false</ip:create-global-addresses>
        </ip:autoconf>
       </ip:ipv6>
      </if:interface>
      <if:interface>
       <if:name>eth1</if:name>
       <if:type>ethernetCsmacd</if:type>
       <if:description>
         Interface to the internal network.
       </if:description>
       <ip:ipv4>
        <ip:address>
         <ip:ip>198.51.100.1</ip:ip>
         <ip:prefix-length>24</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
       </ip:ipv4>
       <ip:ipv6>
        <ip:address>
```

```
         <ip:ip>2001:0db8:0:2::1</ip:ip>
         <ip:prefix-length>64</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
        <ip:autoconf>
         <ip:create-global-addresses>false</ip:create-global-addresses>
        </ip:autoconf>
       </ip:ipv6>
      </if:interface>
     </if:interfaces>
     <if:interfaces-state>
      <if:interface>
       <if:name>eth0</if:name>
       <if:type>ethernetCsmacd</if:type>
       <if:phys-address>00:0C:42:E5:B1:E9</if:phys-address>
       <if:oper-status>up</if:oper-status>
       <if:statistics>
        <if:discontinuity-time>
          2013-07-02T17:11:27+00:58
        </if:discontinuity-time>
       </if:statistics>
      </if:interface>
      <if:interface>
       <if:name>eth1</if:name>
       <if:type>ethernetCsmacd</if:type>
       <if:oper-status>up</if:oper-status>
       <if:phys-address>00:0C:42:E5:B1:EA</if:phys-address>
       <if:statistics>
        <if:discontinuity-time>
          2013-07-02T17:11:27+00:59
        </if:discontinuity-time>
       </if:statistics>
      </if:interface>
     </if:interfaces-state>
     <rt:routing>
      <rt:routing-instance>
       <rt:name>rtr0</rt:name>
       <rt:routing-instance-id>1415926535</rt:routing-instance-id>
       <rt:description>Router A</rt:description>
       <rt:interfaces>
        <rt:interface>
         <rt:name>eth1</rt:name>
         <v6ur:ipv6-router-advertisements>
          <v6ur:send-advertisements>true</v6ur:send-advertisements>
          <v6ur:prefix-list>
           <v6ur:prefix>
            <v6ur:prefix-spec>2001:db8:0:2::/64</v6ur:prefix-spec>
           </v6ur:prefix>
```

```
          </v6ur:prefix-list>
         </v6ur:ipv6-router-advertisements>
        </rt:interface>
       </rt:interfaces>
       <rt:routing-protocols>
        <rt:routing-protocol>
         <rt:name>st0</rt:name>
         <rt:description>
          Static routing is used for the internal network.
         </rt:description>
         <rt:type>rt:static</rt:type>
         <rt:static-routes>
          <v4ur:ipv4>
           <v4ur:route>
            <v4ur:id>1</v4ur:id>
            <v4ur:destination-prefix>0.0.0.0/0</v4ur:destination-prefix>
            <v4ur:gateway>192.0.2.2</v4ur:gateway>
           </v4ur:route>
          </v4ur:ipv4>
          <v6ur:ipv6>
           <v6ur:route>
            <v6ur:id>1</v6ur:id>
            <v6ur:destination-prefix>::/0</v6ur:destination-prefix>
            <v6ur:gateway>2001:db8:0:1::2</v6ur:gateway>
           </v6ur:route>
          </v6ur:ipv6>
         </rt:static-routes>
        </rt:routing-protocol>
       </rt:routing-protocols>
      </rt:routing-instance>
     </rt:routing>
     <rt:routing-state>
      <rt:routing-instance>
       <rt:id>1415926535</rt:id>
       <rt:name>rtr0</rt:name>
       <rt:router-id>192.0.2.1</rt:router-id>
       <rt:default-ribs>
        <rt:default-rib>
         <rt:address-family>v4ur:ipv4-unicast</rt:address-family>
         <rt:rib-id>897932384</rt:rib-id>
        </rt:default-rib>
        <rt:default-rib>
         <rt:address-family>v6ur:ipv6-unicast</rt:address-family>
         <rt:rib-id>751058209</rt:rib-id>
        </rt:default-rib>
       </rt:default-ribs>
       <rt:interfaces>
        <rt:interface>
```

```
        <rt:name>eth0</rt:name>
       </rt:interface>
       <rt:interface>
        <rt:name>eth1</rt:name>
        <v6ur:ipv6-router-advertisements>
         <v6ur:send-advertisements>true</v6ur:send-advertisements>
         <v6ur:prefix-list>
          <v6ur:prefix>
           <v6ur:prefix-spec>2001:db8:0:2::/64</v6ur:prefix-spec>
          </v6ur:prefix>
         </v6ur:prefix-list>
        </v6ur:ipv6-router-advertisements>
       </rt:interface>
      </rt:interfaces>
      <rt:routing-protocols>
       <rt:routing-protocol>
        <rt:name>st0</rt:name>
        <rt:type>rt:static</rt:type>
       </rt:routing-protocol>
      </rt:routing-protocols>
     </rt:routing-instance>
     <rt:ribs>
      <rt:rib>
       <rt:id>897932384</rt:id>
       <rt:address-family>v4ur:ipv4-unicast</rt:address-family>
       <rt:routes>
        <rt:route>
         <rt:id>626433832</rt:id>
         <v4ur:destination-prefix>
           192.0.2.1/24
         </v4ur:destination-prefix>
         <rt:outgoing-interface>eth0</rt:outgoing-interface>
         <rt:source-protocol>rt:direct</rt:source-protocol>
         <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
        </rt:route>
        <rt:route>
         <rt:id>795028841</rt:id>
         <v4ur:destination-prefix>
           198.51.100.0/24
         </v4ur:destination-prefix>
         <rt:outgoing-interface>eth1</rt:outgoing-interface>
         <rt:source-protocol>rt:direct</rt:source-protocol>
         <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
        </rt:route>
        <rt:route>
         <rt:id>971693993</rt:id>
         <v4ur:destination-prefix>0.0.0.0/0</v4ur:destination-prefix>
         <rt:source-protocol>rt:static</rt:source-protocol>
```

```
          <v4ur:gateway>192.0.2.2</v4ur:gateway>
          <rt:last-updated>2013-07-02T18:02:45+01:00</rt:last-updated>
         </rt:route>
        </rt:routes>
       </rt:rib>
       <rt:rib>
        <rt:id>751058209</rt:id>
        <rt:address-family>v6ur:ipv6-unicast</rt:address-family>
        <rt:routes>
         <rt:route>
          <rt:id>749445923</rt:id>
          <v6ur:destination-prefix>
            2001:db8:0:1::/64
          </v6ur:destination-prefix>
          <rt:outgoing-interface>eth0</rt:outgoing-interface>
          <rt:source-protocol>rt:direct</rt:source-protocol>
          <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
         </rt:route>
         <rt:route>
          <rt:id>78164062</rt:id>
          <v6ur:destination-prefix>
            2001:db8:0:2::/64
          </v6ur:destination-prefix>
          <rt:outgoing-interface>eth1</rt:outgoing-interface>
          <rt:source-protocol>rt:direct</rt:source-protocol>
          <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
         </rt:route>
         <rt:route>
          <rt:id>862089986</rt:id>
          <v6ur:destination-prefix>::/0</v6ur:destination-prefix>
          <v6ur:gateway>2001:db8:0:1::2</v6ur:gateway>
          <rt:source-protocol>rt:static</rt:source-protocol>
          <rt:last-updated>2013-07-02T18:02:45+01:00</rt:last-updated>
         </rt:route>
        </rt:routes>
       </rt:rib>
      </rt:ribs>
     </rt:routing-state>
    </data>
   </rpc-reply>
```

Appendix D. Change Log

RFC Editor: remove this section upon publication as an RFC.

D.1. Changes Between Versions -10 and -11

- Migrated address families from IANA enumerations to identities.
- Terminology and node names aligned with the I2RS RIB model: router -> routing instance, routing table -> RIB.
- Introduced uint64 keys for state lists: routing-instance, rib, route, nexthop.
- Described the relationship between system-controlled and user-controlled list entries.
- Feature "user-defined-routing-tables" changed into "advanced-router".
- Made nexthop into a choice in order to allow for nexthop-list (I2RS requirement).
- Added nexthop-list with entries having priorities (backup) and weights (load balancing).
- Updated bibliography references.

D.2. Changes Between Versions -09 and -10

- Added subtree for operational state data ("/routing-state").
- Terms "system-controlled entry" and "user-controlled entry" defined and used.
- New feature "user-defined-routing-tables". Nodes that are useful only with user-defined routing tables are now conditional.
- Added grouping "router-id".
- In routing tables, "source-protocol" attribute of routes now reports only protocol type, and its datatype is "identityref".
- Renamed "main-routing-table" to "default-routing-table".

D.3. Changes Between Versions -08 and -09

- Fixed "must" expresion for "connected-routing-table".
- Simplified "must" expression for "main-routing-table".
- Moved per-interface configuration of a new routing protocol under 'routing-protocol'. This also affects the 'example-rip' module.

D.4. Changes Between Versions -07 and -08

- Changed reference from RFC6021 to RFC6021bis.

D.5. Changes Between Versions -06 and -07

- The contents of <get-reply> in Appendix C was updated: "eth[01]" is used as the value of "location", and "forwarding" is on for both interfaces and both IPv4 and IPv6.
- The "must" expression for "main-routing-table" was modified to avoid redundant error messages reporting address family mismatch when "name" points to a non-existent routing table.
- The default behavior for IPv6 RA prefix advertisements was clarified.
- Changed type of "rt:router-id" to "ip:dotted-quad".
- Type of "rt:router-id" changed to "yang:dotted-quad".
- Fixed missing prefixes in XPath expressions.

D.6. Changes Between Versions -05 and -06

- Document title changed: "Configuration" was replaced by "Management".
- New typedefs "routing-table-ref" and "route-filter-ref".
- Double slashes "//" were removed from XPath expressions and replaced with the single "/".
- Removed uniqueness requirement for "router-id".
- Complete data tree is now in Appendix A.
- Changed type of "source-protocol" from "leafref" to "string".

- Clarified the relationship between routing protocol instances and connected routing tables.

- Added a must constraint saying that a routing table connected to the direct pseudo-protocol must not be a main routing table.

D.7. Changes Between Versions -04 and -05

- Routing tables are now global, i.e., "routing-tables" is a child of "routing" rather than "router".

- "must" statement for "static-routes" changed to "when".

- Added "main-routing-tables" containing references to main routing tables for each address family.

- Removed the defaults for "address-family" and "safi" and made them mandatory.

- Removed the default for route-filter/type and made this leaf mandatory.

- If there is no active route for a given destination, the "active-route" RPC returns no output.

- Added "enabled" switch under "routing-protocol".

- Added "router-type" identity and "type" leaf under "router".

- Route attribute "age" changed to "last-updated", its type is "yang:date-and-time".

- The "direct" pseudo-protocol is always connected to main routing tables.

- Entries in the list of connected routing tables renamed from "routing-table" to "connected-routing-table".

- Added "must" constraint saying that a routing table must not be its own recipient.

D.8. Changes Between Versions -03 and -04

- Changed "error-tag" for both RPC methods from "missing element" to "data-missing".

- Removed the decrementing behavior for advertised IPv6 prefix parameters "valid-lifetime" and "preferred-lifetime".

- Changed the key of the static route lists from "seqno" to "id" because the routes needn't be sorted.
- Added 'must' constraint saying that "preferred-lifetime" must not be greater than "valid-lifetime".

D.9. Changes Between Versions -02 and -03

- Module "iana-afn-safi" moved to I-D "iana-if-type".
- Removed forwarding table.
- RPC "get-route" changed to "active-route". Its output is a list of routes (for multi-path routing).
- New RPC "route-count".
- For both RPCs, specification of negative responses was added.
- Relaxed separation of router instances.
- Assignment of interfaces to router instances needn't be disjoint.
- Route filters are now global.
- Added "allow-all-route-filter" for symmetry.
- Added Section 6 about interactions with "ietf-interfaces" and "ietf-ip".
- Added "router-id" leaf.
- Specified the names for IPv4/IPv6 unicast main routing tables.
- Route parameter "last-modified" changed to "age".
- Added container "recipient-routing-tables".

D.10. Changes Between Versions -01 and -02

- Added module "ietf-ipv6-unicast-routing".
- The example in Appendix C now uses IP addresses from blocks reserved for documentation.
- Direct routes appear by default in the forwarding table.

- Network layer interfaces must be assigned to a router instance. Additional interface configuration may be present.
- The "when" statement is only used with "augment", "must" is used elsewhere.
- Additional "must" statements were added.
- The "route-content" grouping for IPv4 and IPv6 unicast now includes the material from the "ietf-routing" version via "uses rt:route-content".
- Explanation of symbols in the tree representation of data model hierarchy.

D.11. Changes Between Versions -00 and -01

- AFN/SAFI-independent stuff was moved to the "ietf-routing" module.
- Typedefs for AFN and SAFI were placed in a separate "iana-afn-safi" module.
- Names of some data nodes were changed, in particular "routing-process" is now "router".
- The restriction of a single AFN/SAFI per router was lifted.
- RPC operation "delete-route" was removed.
- Illegal XPath references from "get-route" to the datastore were fixed.
- Section "Security Considerations" was written.

Author's Address

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz