NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 5, 2015

L. Lhotka
CZ.NIC
A. Lindem
Cisco Systems

March 04, 2015

A YANG Data Model for Routing Management
draft-ietf-netmod-routing-cfg-17

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 routing protocols and other functions. The core routing data model provides common building blocks for such extensions - routing instances, routes, routing information bases (RIB), and routing protocols.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document contains a specification of the following YANG modules:

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

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

- o 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 intended as a basis for future data model development covering more sophisticated routing systems. While these three modules can be directly used for simple IP devices with static routing (see Appendix B), their main purpose is to provide essential building blocks for more complicated data models 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]:

- o client
- o message
- o protocol operation
- o server

The following terms are defined in [RFC6020]:

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

2.1. Glossary of New Terms

core routing data model: YANG data model comprising "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 a list of routes together with other information. See Section 5.3 for details.

system-controlled entry: An entry of a list in 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 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:

- o Brackets "[" and "]" enclose list keys.
- o Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- o Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only), "-x" RPC operations, and "-n" notifications.
- o Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o 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 operations 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	[RFC7223]
ip	ietf-ip	[RFC7277]
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:

- o 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).

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

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

- o 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 state data hierarchies. See Appendix A for the complete data trees.

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

Figure 1: Configuration data hierarchy.

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

Figure 2: 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, and routing protocols. 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.

```
   +--------+
   | direct |       +--------------+       +--------------+
   | routes |------>|              |<------|              |
   +--------+       |   default    |       |  additional  |
                    |     RIB      |       |     RIB      |
   +--------+       |              |       |              |
   | static |------>|              |------>|              |
   | routes |       +--------------+       +--------------+
   +--------+           ^      |               ^      |
                        |      |               |      |
                        |      v               |      v
                      +----------+           +----------+
                      | routing  |           | routing  |
                      | protocol |           | protocol |
                      +----------+           +----------+
```

Figure 3: Example set-up 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:

- o Along with the default RIB, which is always present, an additional RIB is configured.
- o Each routing protocol instance, including the "static" and "direct" pseudo-protocols, is connected to one or more RIBs with which it can exchange routes (in both directions, except for the "static" and "direct" pseudo-protocols).
- o RIBs may also be connected to each other and exchange routes in either direction (or both).

4.1. System-Controlled and User-Controlled List Entries

The core routing data model defines several lists, for example "routing-instance" or "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 state data, i.e., inside the "routing-state" container.

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

Corresponding entries in both versions of the list (in state data and configuration) have the same value of the list key.

The user may also provide supplemental configuration of system-controlled entries. To do so, the user creates a new entry in the configuration with the desired contents. In order to bind this entry with the corresponding entry in the state data list, the key of the configuration entry has to be set to the same value as the key of the state entry.

An example can be seen in Appendix D: the "/routing-state/routing-instance" list has a single system-controlled entry whose "name" key has the value "rtr0". This entry is configured by the "/routing/routing-instance" entry whose "name" key is also "rtr0".

Deleting a user-controlled entry from the configuration list results in the removal of the corresponding entry in the state data 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 state data entry remains in the list.

5. Basic Building Blocks

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

5.1. Routing Instance

The core routing data model supports one or more routing instances appearing as entries of the "routing-instance" list. Each routing instance has separate configuration and state data under "/rt:routing/rt:routing-instance" and "/rt:routing-state/rt:routing-instance", respectively.

The semantics of the term "routing instance" is deliberately left undefined. It is expected that future YANG modules will define data models for specific types of routing instances, such as VRF (virtual routing and forwarding) instances that are used for BGP/MPLS virtual private networks [RFC4364]. For each type of routing instance, an identity derived from "rt:routing-instance" MUST be defined. This identity is then referred to by the value of the "type" leaf (a child node of "routing-instance" list).

An implementation MAY create one or more system-controlled routing instances, and MAY also impose restrictions on types of routing instances that can be configured, and on the maximum 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:default-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 [RFC7223].

A data model for a routing instance type MAY state additional rules for the assignment of interfaces to routing instances of that type. For example, it may be required that the sets of interfaces assigned to different routing instances of a certain type 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 state data, with definitions of the following variables as required by [RFC4861], sec. 6.2.1:

- o send-advertisements,
- o max-rtr-adv-interval,
- o min-rtr-adv-interval,
- o managed-flag,
- o other-config-flag,
- o link-mtu,
- o reachable-time,
- o retrans-timer,
- o cur-hop-limit,
- o default-lifetime,
- o 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 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 [RFC7277] (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.

- "route-preference": an integer value (also known as administrative distance) that is used for selecting a preferred route among routes with the same destination prefix. A lower value means a more preferred route.

- "next-hop": determines the action to be performed with a packet. See below for details.

The choice of next-hops comprises the following cases:

- o simple next-hop - IP address of the next-hop router, outgoing interface, or both.

- o special next-hop - a keyword indicating special packet handling, one of:

  * "blackhole" - silently discard the packet;

  * "unreachable" - discard the packet and notify the sender with a "destination unreachable" error message;

  * "prohibit" - discard the packet notify the sender with an "administratively prohibited" error message.

It is expected that future YANG modules defining will augment routes with more complex next-hop types, or additional attributes such as metrics.

Routes are primarily state data that appear as entries of RIBs (Section 5.3) but they may also be found in configuration data, for example as manually configured static routes. In the latter case, configurable route attributes are generally a subset of route attributes described above.

5.3. Routing Information Base (RIB)

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

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

- o "active": an implementation can use this empty leaf to indicate that the route is preferred among all routes in the same RIB that have the same destination prefix.

- 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 one address family. An address family is represented by an identity derived from the "rt:address-family" base identity.

In the core routing data model, RIBs are 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, a data model for a routing instance type MAY state rules and restrictions for sharing RIBs among routing instances of that type.

Each routing instance has, 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 "multiple-ribs" will typically create one system-controlled RIB per supported address family, and declare it as the 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 "multiple-ribs" 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.

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).

Multiple routing protocol instances of the same type MAY be configured within the same routing instance.

Each routing protocol instance can be connected to one or more RIBs for each address family that the routing protocol instance supports. By default, the interaction of a routing protocol instance with its connected RIBs is governed by the following rules:

- o Routes learned from the network are installed in all connected RIBs with a matching address family.

- o Conversely, routes from all connected RIBs are injected into the routing protocol instance.

However, a data model for a routing protocol MAY impose specific rules for exchanging routes between routing protocol instances and connected RIBs.

On devices supporting the "multiple-ribs" 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.

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 they are confined to the local device and do not exchange any routing information with adjacent routers. Routes from both "direct" and "static" protocol instances are passed to the connected RIBs, but an exchange in the opposite direction is not allowed.

Every routing instance MUST implement exactly one instance of the "direct" pseudo-protocol type. 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.

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.

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:

- 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".

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

  /rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route

  and

  /rt:fib-route/rt:output/rt:route,

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

- 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".

- 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 protocol-specific data nodes be encapsulated in appropriately named containers.

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

5.5. RPC Operations

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

- fib-route: query a routing instance for the active route in the Forwarding Information Base (FIB). It is the route that is 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 depends 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 [RFC7223]:

/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 [RFC7277]:

/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 Management YANG Module

RFC Editor: 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@2015-03-04.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: Thomas Nadeau
               <mailto:tnadeau@lucidvision.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) 2014 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 2015-03-04 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for Routing Management";
     }

     /* Features */

     feature multiple-ribs {
       description
         "This feature indicates that the server supports user-defined
          RIBS and the framework for passing routes between RIBs.

          Servers that do not advertise this feature MUST provide
          exactly one system-controlled RIB per supported address family
          and make them also the default RIBs. These RIBs then appear as
          entries of the list /routing-state/ribs/rib.";
     }

     feature router-id {
       description
         "This feature indicates that the server supports configuration
          of an explicit 32-bit router ID that is used by some routing
          protocols.

          Servers that do not advertise this feature set a router ID
          algorithmically, usually to one of configured IPv4 addresses.
          However, this algorithm is implementation-specific.";
     }

     /* 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 {
       description
         "Base identity from which identities describing routing
          instance types are derived.";
     }

     identity default-routing-instance {
       base routing-instance;
       description
         "This identity represents either a default routing instance, or
          the only routing instance on systems that do not support
          multiple instances.";
     }

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

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

     identity static {
       base routing-protocol;
       description
```

```
         "Static routing pseudo-protocol.";
     }

     /* 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:name";
       }
       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:name";
       }
       description
         "This type is used for leafs that reference a RIB in state
          data.";
     }

     typedef route-preference {
       type uint32;
       description
         "This type is used for route preferences.";
     }

     /* 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 router ID.";
       leaf router-id {
         type yang:dotted-quad;
         description
           "A 32-bit number in the form of a dotted quad that is used by
            some routing protocols identifying a router.";
         reference
           "RFC 2328: OSPF Version 2.";
       }
     }

     grouping special-next-hop {
       description
         "This grouping provides a leaf with an enumeration of special
          next-hops.";
       leaf special-next-hop {
         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 system.";
          }
        }
        description
          "Special next-hop options.";
      }
    }

    grouping next-hop-content {
      description
        "Generic parameters of next-hops in static routes.";
      choice next-hop-options {
        mandatory "true";
        description
          "Options for next-hops in static routes.

           It is expected that other cases will be added through
           augments from other modules, e.g., for ECMP.";
        case simple-next-hop {
          description
            "Simple next-hop is specified as an outgoing interface,
             next-hop address or both.

             Address-family-specific modules are expected to provide
             'next-hop-address' leaf via augmentation.";
          leaf outgoing-interface {
            type leafref {
              path "/rt:routing/rt:routing-instance/rt:interfaces/"
                 + "rt:interface/rt:name";
            }
            description
              "Name of the outgoing interface.";
          }
        }
        case special-next-hop {
          uses special-next-hop;
        }
      }
    }

    grouping next-hop-state-content {
      description
        "Generic parameters of next-hops in state data.";
      choice next-hop-options {
        mandatory "true";
```

```
        description
          "Options for next-hops in state data.

           It is expected that other cases will be added through
           augments from other modules, e.g., for ECMP or recursive
           next-hops.";
        case simple-next-hop {
          description
            "Simple next-hop is specified as an outgoing interface,
             next-hop address or both.

             Address-family-specific modules are expected to provide
             'next-hop-address' leaf via augmentation.";
          leaf outgoing-interface {
            type leafref {
              path "/rt:routing-state/rt:routing-instance/"
                 + "rt:interfaces/rt:interface/rt:name";
            }
            description
              "Name of the outgoing interface.";
          }
        }
        case special-next-hop {
          uses special-next-hop;
        }
      }
    }

    grouping route-metadata {
      description
        "Common route metadata.";
      leaf source-protocol {
        type identityref {
          base routing-protocol;
        }
        mandatory "true";
        description
          "Type of the routing protocol from which the route
           originated.";
      }
      leaf active {
        type empty;
        description
          "Presence of this leaf indicates that the route is preferred
           among all routes in the same RIB that have the same
           destination prefix.";
      }
      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.";
       }
     }

     /* State data */

     augment "/if:interfaces-state/if:interface" {
       description
         "This augment adds a wrapped leaf-list to interface state
          data.";
       container routing-instances {
         description
           "The enclosed leaf-list provides references to all routing
            instances to which the parent interface is assigned.

            The assignments are configured as a part of routing-instance
            configuration (module ietf-routing).";
         leaf-list routing-instance {
           type routing-instance-state-ref;
           must "../../if:name=/rt:routing-state/"
              + "rt:routing-instance[rt:name=current()]/rt:interfaces/"
              + "rt:interface/rt:name" {
             error-message "The interface is not assigned to the "
                         + "routing instance.";
             description
               "The reference must mirror a corresponding assignment
                under routing-instance.";
           }
           description
             "Reference to a routing instance.";
         }
       }
     }

     container routing-state {
       config "false";
       description
         "State data of the routing subsystem.";
       list routing-instance {
         key "name";
         min-elements "1";
         description
           "Each list entry is a container for 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 name {
           type string;
           description
             "The name of the routing instance.

              For system-controlled instances the name is persistent,
              i.e., it SHOULD NOT change across reboots.";
         }
         leaf type {
           type identityref {
             base routing-instance;
           }
           description
             "The routing instance type.";
         }
         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-name {
               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 "type name";
             description
               "State data of a routing protocol instance.

                An implementation MUST provide exactly one
                system-controlled instance of the type 'direct'. Other
                instances MAY be created by configuration.";
             leaf type {
               type identityref {
                 base routing-protocol;
               }
               description
                 "Type of the routing protocol.";
             }
             leaf name {
               type string;
               description
                 "The name of the routing protocol instance.

                  For system-controlled instances this name is
                  persistent, i.e., it SHOULD NOT change across
                  reboots.";
             }
             leaf route-preference {
               type route-preference;
               mandatory "true";
               description
                 "The value of route preference (administrative
                  distance) assigned to all routes generated by the
                  routing protocol instance. A lower value means a more
                  preferred route.";
             }
```

```
            container connected-ribs {
              description
                "Container for connected RIBs.";
              list connected-rib {
                key "rib-name";
                description
                  "List of RIBs to which the routing protocol instance
                   is connected.

                   By default, routes learned by the routing protocol
                   instance are installed in all connected RIBs of the
                   matching address family, and, conversely, all routes
                   from connected RIBs are installed in the routing
                   protocol instance. However, routing protocols may
                   specify other rules.";
                leaf rib-name {
                  type rib-state-ref;
                  description
                    "Name of an existing RIB.";
                }
              }
            }
          }
        }
      }
      container ribs {
        description
          "Container for RIBs.";
        list rib {
          key "name";
          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 provide a system-controlled default RIB
             for each supported address family, and MAY provide other
             system-controlled RIBs. Additional RIBs MAY be created in
             the configuration.";
          leaf name {
            type string;
            description
              "The name of the RIB.";
          }
          uses address-family;
          container routes {
            description
              "Current content of the RIB.";
```

```
          list route {
            description
              "A RIB route entry. This data node MUST be augmented
               with information specific for routes of each address
               family.";
            leaf route-preference {
              type route-preference;
              description
                "This route attribute, also known as administrative
                 distance, allows for selecting the preferred route
                 among routes with the same destination prefix. A
                 smaller value means a more preferred route.";
            }
            container next-hop {
              description
                "Route’s next-hop attribute.";
              uses next-hop-state-content;
            }
            uses route-metadata;
          }
        }
        container recipient-ribs {
          description
            "Container for recipient RIBs.";
          list recipient-rib {
            key "rib-name";
            description
              "List of RIBs that receive routes from this RIB.";
            leaf rib-name {
              type rib-state-ref;
              description
                "The name of the recipient RIB.";
            }
          }
        }
      }
    }
  }

  /* Configuration Data */

  container routing {
    description
      "Configuration parameters for the routing subsystem.";
    list routing-instance {
      key "name";
      description
        "Configuration of a routing instance.";
```

```
         leaf name {
           type string;
           description
             "The name of the routing instance.

              For system-controlled entries, the value of this leaf must
              be the same as the name of the corresponding entry in
              state data.

              For user-controlled entries, an arbitrary name can be
              used.";
         }
         leaf type {
           type identityref {
             base routing-instance;
           }
           default "rt:default-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 state data, despite any
              other configuration that might be present.";
         }
         uses router-id {
           if-feature router-id;
           description
             "Configuration of the global router ID. Routing protocols
              that use router ID can use this parameter or override it
              with another value.";
         }
         leaf description {
           type string;
           description
             "Textual description of the routing instance.";
         }
         container default-ribs {
           if-feature multiple-ribs;
           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()/"
                + "rib-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 rib-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 "type name";
             description
               "Each entry contains configuration of a routing protocol
```

```
               instance.";
            leaf type {
              type identityref {
                base routing-protocol;
              }
              description
                "Type of the routing protocol - an identity derived
                 from the 'routing-protocol' base identity.";
            }
            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
                 state data, despite any other configuration that might
                 be present.";
            }
            leaf route-preference {
              type route-preference;
              description
                "The value of route preference (administrative
                 distance).

                 The default value depends on the routing protocol
                 type, and may also be implementation-dependent.";
            }
            container connected-ribs {
              description
                "Configuration of connected RIBs.";
              list connected-rib {
                key "rib-name";
                description
                  "Each entry configures a RIB to which the routing
                   protocol instance is connected.
```

```
                   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/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.";
                }
              }
            }
            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
          "Configuration of RIBs.";
        list rib {
          key "name";
          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.

                For system-controlled entries, the value of this leaf
                must be the same as the name of the corresponding entry
                in state data.

                For user-controlled entries, an arbitrary name can be
                used.";
           }
           uses address-family;
           leaf description {
             type string;
             description
               "Textual description of the RIB.";
           }
           container recipient-ribs {
             if-feature multiple-ribs;
             description
               "Configuration of recipient RIBs.";
             list recipient-rib {
               must "rib-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()/rib-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.";
               }
             }
           }
         }
       }
     }
```

```
     /* RPC operations */

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

              If the routing instance with name equal to the value of
              this parameter 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 FIB route for the specified destination.

              If the routing instance has no active FIB 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;
           container next-hop {
             description
               "Route's next-hop attribute.";
             uses next-hop-state-content;
           }
           uses route-metadata;
         }
```

```
      }
    }

    rpc route-count {
      description
        "Return the current number of routes in a RIB.";
      input {
        leaf rib-name {
          type rib-state-ref;
          mandatory "true";
          description
            "Name of the RIB.

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

  <CODE ENDS>
```

8. IPv4 Unicast Routing Management YANG Module

RFC Editor: 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@2015-02-10.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: Thomas Nadeau
                <mailto:tnadeau@lucidvision.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 state data for IPv4 unicast routing.

      Copyright (c) 2014 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 2015-02-10 {
     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.";
     }

     /* 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:next-hop/rt:next-hop-options/rt:simple-next-hop" {
       when "../../../rt:address-family = 'v4ur:ipv4-unicast'" {
         description
           "This augment is valid only for IPv4 unicast.";
       }
       description
         "This leaf augments the 'simple-next-hop' case of IPv4 unicast
          routes.";
       leaf next-hop-address {
         type inet:ipv4-address;
         description
           "IPv4 address of the next-hop.";
       }
     }

     /* 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 "destination-prefix";
           ordered-by "user";
           description
             "A user-ordered list of static routes.";
           leaf destination-prefix {
             type inet:ipv4-prefix;
             mandatory "true";
             description
               "IPv4 destination prefix.";
           }
           leaf description {
             type string;
             description
               "Textual description of the route.";
           }
           container next-hop {
             description
               "Configuration of next-hop.";
             uses rt:next-hop-content {
               augment "next-hop-options" {
                 description
                   "Add next-hop address case.";
                 leaf next-hop-address {
                   type inet:ipv4-address;
                   description
                     "IPv4 address of the next-hop.";
                 }
               }
             }
           }
         }
       }
     }

     /* RPC operations */

     augment "/rt:fib-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:fib-route' operation.";
       leaf address {
         type inet:ipv4-address;
```

```
         description
           "IPv4 destination address.";
       }
     }

     augment "/rt:fib-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:fib-route'
          operation.";
       leaf destination-prefix {
         type inet:ipv4-prefix;
         description
           "IPv4 destination prefix.";
       }
     }

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

   <CODE ENDS>
```

9. IPv6 Unicast Routing Management YANG Module

RFC Editor: 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@2015-02-10.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: Thomas Nadeau
                <mailto:tnadeau@lucidvision.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 state data for IPv6 unicast routing.

      Copyright (c) 2014 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 2015-02-10 {
       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.";
     }

     /* State data */

     augment "/rt:routing-state/rt:routing-instance/rt:interfaces/"
           + "rt:interface" {
       description
         "IPv6-specific parameters of router interfaces.";
       container ipv6-router-advertisements {
         description
           "Parameters of IPv6 Router Advertisements.";
         leaf send-advertisements {
           type boolean;
           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";
           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";
          description
            "The minimum time allowed between sending unsolicited
             multicast Router Advertisements from the interface.";
        }
        leaf managed-flag {
          type boolean;
          description
            "The value that is placed in the 'Managed address
             configuration' flag field in the Router Advertisement.";
        }
        leaf other-config-flag {
          type boolean;
          description
            "The value that is placed in the 'Other configuration' flag
             field in the Router Advertisement.";
        }
        leaf link-mtu {
          type uint32;
          description
            "The value that is 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";
          description
            "The value that is placed in the Reachable Time field in
             the Router Advertisement messages sent by the router. A
             value of zero means unspecified (by this router).";
        }
        leaf retrans-timer {
          type uint32;
          units "milliseconds";
          description
            "The value that is placed in the Retrans Timer field in the
             Router Advertisement messages sent by the router. A value
             of zero means unspecified (by this router).";
        }
        leaf cur-hop-limit {
          type uint8;
          description
            "The value that is placed in the Cur Hop Limit field in the
             Router Advertisement messages sent by the router. A value
```

```
             of zero means unspecified (by this router).";
        }
        leaf default-lifetime {
          type uint16 {
            range "0..9000";
          }
          units "seconds";
          description
            "The value that is placed in the Router Lifetime field of
             Router Advertisements sent from the interface, in seconds.
             A value of zero indicates that the router is not to be
             used as a default router.";
        }
        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.";
          list prefix {
            key "prefix-spec";
            description
              "Advertised prefix entry and its parameters.";
            leaf prefix-spec {
              type inet:ipv6-prefix;
              description
                "IPv6 address prefix.";
            }
            leaf valid-lifetime {
              type uint32;
              units "seconds";
              description
                "The value that is placed in the Valid Lifetime in the
                 Prefix Information option. The designated value of all
                 1’s (0xffffffff) represents infinity.";
            }
            leaf on-link-flag {
              type boolean;
              description
                "The value that is placed in the on-link flag (’L-bit’)
                 field in the Prefix Information option.";
            }
            leaf preferred-lifetime {
              type uint32;
              units "seconds";
```

```
              description
                "The value that is placed in the Preferred Lifetime in
                 the Prefix Information option, in seconds. The
                 designated value of all 1’s (0xffffffff) represents
                 infinity.";
            }
            leaf autonomous-flag {
              type boolean;
              description
                "The value that is placed in the Autonomous Flag field
                 in the Prefix Information option.";
            }
          }
        }
      }
    }

    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:next-hop/rt:next-hop-options/rt:simple-next-hop" {
      when "../../../rt:address-family = ’v6ur:ipv6-unicast’" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This leaf augments the ’simple-next-hop’ case of IPv6 unicast
         routes.";
      leaf next-hop-address {
        type inet:ipv6-address;
        description
          "IPv6 address of the next-hop.";
      }
    }

    /* 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.";
         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";
           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.

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

              If this parameter is not configured, the device SHOULD use
              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.
```

```
             It 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, the device SHOULD use
             a value of 3 * max-rtr-adv-interval.";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
             AdvDefaultLifeTime.";
        }
        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.

             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
              "Configuration of an 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. 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";
                    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. 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/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 "destination-prefix";
          ordered-by "user";
          description
            "A user-ordered list of static routes.";
          leaf destination-prefix {
            type inet:ipv6-prefix;
            mandatory "true";
            description
              "IPv6 destination prefix.";
          }
          leaf description {
            type string;
            description
              "Textual description of the route.";
          }
          container next-hop {
            description
              "Configuration of next-hop.";
            uses rt:next-hop-content {
              augment "next-hop-options" {
                description
                  "Add next-hop address case.";
                leaf next-hop-address {
                  type inet:ipv6-address;
```

```
                   description
                     "IPv6 address of the next-hop.";
                 }
               }
             }
           }
         }
       }
     }

     /* RPC operations */

     augment "/rt:fib-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:fib-route' operation.";
       leaf address {
         type inet:ipv6-address;
         description
           "IPv6 destination address.";
       }
     }

     augment "/rt:fib-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:fib-route'
          operation.";
       leaf destination-prefix {
         type inet:ipv6-prefix;
         description
           "IPv6 destination prefix.";
       }
     }

     augment "/rt:fib-route/rt:output/rt:route/rt:next-hop/"
           + "rt:next-hop-options/rt:simple-next-hop" {
       when "../rt:address-family='v6ur:ipv6-unicast'" {
         description
           "This augment is valid only for IPv6 unicast.";
       }
```

```
       description
         "This leaf augments the 'simple-next-hop' case in the reply to
          the 'rt:fib-route' operation.";
       leaf next-hop-address {
         type inet:ipv6-address;
         description
           "IPv6 address of the next-hop.";
       }
     }
   }

   <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]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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/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 authors wish to thank Nitin Bahadur, Martin Bjorklund, Dean Bogdanovic, Jeff Haas, Joel Halpern, Wes Hardaker, Sriganesh Kini, David Lamparter, Andrew McGregor, Jan Medved, Xiang Li, Stephane Litkowski, Thomas Morin, Tom Petch, Bruno Rijsman, Juergen Schoenwaelder, Phil Shafer, Dave Thaler, Yi Yang, Derek Man-Kit Yeung and Jeffrey Zhang 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., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

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

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

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, June 2014.

13.2. Informative References

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[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.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Appendix A. The Complete Data Trees

This appendix presents the complete configuration and 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 type?                identityref
   |  +--rw enabled?             boolean
   |  +--rw router-id?           yang:dotted-quad
   |  +--rw description?         string
   |  +--rw default-ribs {multiple-ribs}?
   |  |  +--rw default-rib* [address-family]
   |  |     +--rw address-family    identityref
   |  |     +--rw rib-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* [type name]
   |        +--rw type                identityref
   |        +--rw name                string
   |        +--rw description?        string
   |        +--rw enabled?            boolean
   |        +--rw route-preference?   route-preference
   |        +--rw connected-ribs
   |        |  +--rw connected-rib* [rib-name]
   |        |     +--rw rib-name    rib-ref
   |        +--rw static-routes
   |           +--rw v6ur:ipv6
   |           |  +--rw v6ur:route* [destination-prefix]
   |           |     +--rw v6ur:destination-prefix    inet:ipv6-prefix
   |           |     +--rw v6ur:description?          string
   |           |     +--rw v6ur:next-hop
   |           |        +--rw (next-hop-options)
   |           |           +--:(simple-next-hop)
   |           |           |  +--rw v6ur:outgoing-interface?
   |           |           +--:(special-next-hop)
   |           |           |  +--rw v6ur:special-next-hop?   enumeration
   |           |           +--:(next-hop-address)
   |           |              +--rw v6ur:next-hop-address?
   |           +--rw v4ur:ipv4
   |              +--rw v4ur:route* [destination-prefix]
   |                 +--rw v4ur:destination-prefix    inet:ipv4-prefix
   |                 +--rw v4ur:description?          string
   |                 +--rw v4ur:next-hop
   |                    +--rw (next-hop-options)
   |                       +--:(simple-next-hop)
   |                       |  +--rw v4ur:outgoing-interface?
```

```
   |                       +--:(special-next-hop)
   |                       |  +--rw v4ur:special-next-hop?   enumeration
   |                       +--:(next-hop-address)
   |                          +--rw v4ur:next-hop-address?
   +--rw ribs
      +--rw rib* [name]
         +--rw name              string
         +--rw address-family    identityref
         +--rw description?      string
         +--rw recipient-ribs {multiple-ribs}?
            +--rw recipient-rib* [rib-name]
               +--rw rib-name    rib-ref
```

A.2. State Data

```
  +--ro routing-state
     +--ro routing-instance* [name]
     |  +--ro name                 string
     |  +--ro type?                identityref
     |  +--ro default-ribs
     |  |  +--ro default-rib* [address-family]
     |  |     +--ro address-family    identityref
     |  |     +--ro rib-name          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* [type name]
     |        +--ro type                identityref
     |        +--ro name                string
```

```
     |        +--ro route-preference    route-preference
     |        +--ro connected-ribs
     |           +--ro connected-rib* [rib-name]
     |              +--ro rib-name    rib-state-ref
     +--ro ribs
        +--ro rib* [name]
           +--ro name              string
           +--ro address-family    identityref
           +--ro routes
           |  +--ro route*
           |     +--ro route-preference?          route-preference
           |     +--ro next-hop
           |     |  +--ro (next-hop-options)
           |     |     +--:(simple-next-hop)
           |     |     |  +--ro outgoing-interface?
           |     |     |  +--ro v6ur:next-hop-address? inet:ipv6-address
           |     |     |  +--ro v4ur:next-hop-address? inet:ipv4-address
           |     |     +--:(special-next-hop)
           |     |        +--ro special-next-hop?        enumeration
           |     +--ro source-protocol            identityref
           |     +--ro active?                    empty
           |     +--ro last-updated?              yang:date-and-time
           |     +--ro v6ur:destination-prefix?   inet:ipv6-prefix
           |     +--ro v4ur:destination-prefix?   inet:ipv4-prefix
           +--ro recipient-ribs
              +--ro recipient-rib* [rib-name]
                 +--ro rib-name    rib-state-ref
```

Appendix B. Minimum Implementation

Some parts and options of the core routing model, such as user-defined routing tables, are intended only for advanced routers. This appendix gives basic non-normative guidelines for implementing a bare minimum of available functions. Such an implementation may be used for hosts or very simple routers.

A minimum implementation will provide a single system-controlled routing instance, and will not allow clients to create any user-controlled instances.

Typically, the feature "multiple-ribs" will not be supported. This means that a single system-controlled RIB is available for each supported address family - IPv4, IPv6 or both. These RIBs must be the default RIBs, so references to them will also appear as system-controlled entries of the "default-rib" list in state data. No user-controlled RIBs are allowed.

In addition to the mandatory instance of the "direct" pseudo-protocol, a minimum implementation should support configured instance(s) of the "static" pseudo-protocol. Even with a single RIB per address family, it may be occasionally useful to be able to configure multiple "static" instances. For example, a client may want to configure alternative sets of static routes and activate or deactivate them by means of connnecting the default RIB to the corresponding "static" instance.

Platforms with severely constrained resources may use deviations for restricting the data model, e.g., limiting the number of "static" routing protocol instances.

Appendix C. 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, this module does not obey 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 D. 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:

- o ietf-interfaces [RFC7223],
- o ietf-ip [RFC7277],
- o ietf-routing (Section 7),
- o ietf-ipv4-unicast-routing (Section 8),
- o ietf-ipv6-unicast-routing (Section 9).

We assume a simple network set-up as shown in Figure 4: 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.

```
                +-----------------+
                |                 |
                |    Router ISP   |
                |                 |
                +--------+--------+
                         |2001:db8:0:1::2
                         |192.0.2.2
                         |
                         |
                         |2001:db8:0:1::1
                     eth0|192.0.2.1
                +--------+--------+
                |                 |
                |     Router A    |
                |                 |
                +--------+--------+
                     eth1|198.51.100.1
                         |2001:db8:0:2::1
                         |
```

Figure 4: 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:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
     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>ianaift: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>ianaift: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>ianaift: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>
        2014-10-24T17:11:27+00:58
       </if:discontinuity-time>
      </if:statistics>
      <ip:ipv4>
       <ip:forwarding>true</ip:forwarding>
       <ip:mtu>1500</ip:mtu>
       <ip:address>
        <ip:ip>192.0.2.1</ip:ip>
        <ip:prefix-length>24</ip:prefix-length>
       </ip:address>
      </ip:ipv4>
      <ip:ipv6>
       <ip:forwarding>true</ip:forwarding>
       <ip:mtu>1500</ip:mtu>
       <ip:address>
        <ip:ip>2001:0db8:0:1::1</ip:ip>
        <ip:prefix-length>64</ip:prefix-length>
       </ip:address>
      </ip:ipv6>
     </if:interface>
     <if:interface>
      <if:name>eth1</if:name>
      <if:type>ianaift: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>
        2014-10-24T17:11:27+00:59
       </if:discontinuity-time>
      </if:statistics>
      <ip:ipv4>
       <ip:forwarding>true</ip:forwarding>
       <ip:mtu>1500</ip:mtu>
       <ip:address>
        <ip:ip>198.51.100.1</ip:ip>
        <ip:prefix-length>24</ip:prefix-length>
       </ip:address>
      </ip:ipv4>
      <ip:ipv6>
       <ip:forwarding>true</ip:forwarding>
       <ip:mtu>1500</ip:mtu>
       <ip:address>
        <ip:ip>2001:0db8:0:2::1</ip:ip>
        <ip:prefix-length>64</ip:prefix-length>
       </ip:address>
      </ip:ipv6>
     </if:interface>
```

```
    </if:interfaces-state>
    <rt:routing>
     <rt:routing-instance>
      <rt:name>rtr0</rt:name>
      <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:type>rt:static</rt:type>
        <rt:name>st0</rt:name>
        <rt:description>
         Static routing is used for the internal network.
        </rt:description>
        <rt:static-routes>
         <v4ur:ipv4>
          <v4ur:route>
           <v4ur:destination-prefix>0.0.0.0/0</v4ur:destination-prefix>
           <v4ur:next-hop>
        <v4ur:next-hop-address>192.0.2.2</v4ur:next-hop-address>
       </v4ur:next-hop>
          </v4ur:route>
         </v4ur:ipv4>
         <v6ur:ipv6>
          <v6ur:route>
           <v6ur:destination-prefix>::/0</v6ur:destination-prefix>
           <v6ur:next-hop>
        <v6ur:next-hop-address>2001:db8:0:1::2</v6ur:next-hop-address>
       </v6ur:next-hop>
          </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:name>rtr0</rt:name>
      <rt:default-ribs>
       <rt:default-rib>
        <rt:address-family>v4ur:ipv4-unicast</rt:address-family>
        <rt:rib-name>ipv4-master</rt:rib-name>
       </rt:default-rib>
       <rt:default-rib>
        <rt:address-family>v6ur:ipv6-unicast</rt:address-family>
        <rt:rib-name>ipv6-master</rt:rib-name>
       </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:type>rt:static</rt:type>
        <rt:name>st0</rt:name>
        <rt:route-preference>5</rt:route-preference>
       </rt:routing-protocol>
      </rt:routing-protocols>
     </rt:routing-instance>
     <rt:ribs>
      <rt:rib>
       <rt:name>ipv4-master</rt:name>
       <rt:address-family>v4ur:ipv4-unicast</rt:address-family>
       <rt:routes>
        <rt:route>
         <v4ur:destination-prefix>192.0.2.1/24</v4ur:destination-prefix>
         <rt:next-hop>
      <rt:outgoing-interface>eth0</rt:outgoing-interface>
         </rt:next-hop>
         <rt:route-preference>0</rt:route-preference>
         <rt:source-protocol>rt:direct</rt:source-protocol>
```

```
           <rt:last-updated>2014-10-24T17:11:27+01:00</rt:last-updated>
          </rt:route>
          <rt:route>
        <v4ur:destination-prefix>198.51.100.0/24</v4ur:destination-prefix>
           <rt:next-hop>
        <rt:outgoing-interface>eth1</rt:outgoing-interface>
           </rt:next-hop>
           <rt:source-protocol>rt:direct</rt:source-protocol>
           <rt:route-preference>0</rt:route-preference>
           <rt:last-updated>2014-10-24T17:11:27+01:00</rt:last-updated>
          </rt:route>
          <rt:route>
           <v4ur:destination-prefix>0.0.0.0/0</v4ur:destination-prefix>
           <rt:source-protocol>rt:static</rt:source-protocol>
           <rt:route-preference>5</rt:route-preference>
           <rt:next-hop>
        <v4ur:next-hop-address>192.0.2.2</v4ur:next-hop-address>
           </rt:next-hop>
           <rt:last-updated>2014-10-24T18:02:45+01:00</rt:last-updated>
          </rt:route>
         </rt:routes>
        </rt:rib>
        <rt:rib>
         <rt:name>ipv6-master</rt:name>
         <rt:address-family>v6ur:ipv6-unicast</rt:address-family>
         <rt:routes>
          <rt:route>
           <v6ur:destination-prefix>
        2001:db8:0:1::/64
           </v6ur:destination-prefix>
           <rt:next-hop>
        <rt:outgoing-interface>eth0</rt:outgoing-interface>
           </rt:next-hop>
           <rt:source-protocol>rt:direct</rt:source-protocol>
           <rt:route-preference>0</rt:route-preference>
           <rt:last-updated>2014-10-24T17:11:27+01:00</rt:last-updated>
          </rt:route>
          <rt:route>
           <v6ur:destination-prefix>
        2001:db8:0:2::/64
           </v6ur:destination-prefix>
           <rt:next-hop>
        <rt:outgoing-interface>eth1</rt:outgoing-interface>
           </rt:next-hop>
           <rt:source-protocol>rt:direct</rt:source-protocol>
           <rt:route-preference>0</rt:route-preference>
           <rt:last-updated>2014-10-24T17:11:27+01:00</rt:last-updated>
          </rt:route>
```

```
        <rt:route>
         <v6ur:destination-prefix>::/0</v6ur:destination-prefix>
         <rt:next-hop>
      <v6ur:next-hop-address>2001:db8:0:1::2</v6ur:next-hop-address>
         </rt:next-hop>
         <rt:source-protocol>rt:static</rt:source-protocol>
         <rt:route-preference>5</rt:route-preference>
         <rt:last-updated>2014-10-24T18:02:45+01:00</rt:last-updated>
        </rt:route>
       </rt:routes>
      </rt:rib>
     </rt:ribs>
    </rt:routing-state>
   </data>
  </rpc-reply>
```

Appendix E. Change Log

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

E.1. Changes Between Versions -16 and -17

- Added Acee as a co-author.
- Removed all traces of route filters.
- Removed numeric IDs of list entries in state data.
- Removed all next-hop cases except "simple-next-hop" and "special-next-hop".
- Removed feature "multipath-routes".
- Augmented "ietf-interfaces" module with a leaf-list of leafrefs pointing form state data of an interface entry to the routing instance(s) to which the interface is assigned.

E.2. Changes Between Versions -15 and -16

- Added 'type' as the second key component of 'routing-protocol', both in configuration and state data.
- The restriction of no more than one connected RIB per address family was removed.
- Removed the 'id' key of routes in RIBs. This list has no keys anymore.

- o Remove the ’id’ key from static routes and make ’destination-prefix’ the only key.
- o Added ’route-preference’ as a new attribute of routes in RIB.
- o Added ’active’ as a new attribute of routes in RIBs.
- o Renamed RPC operation ’active-route’ to ’fib-route’.
- o Added ’route-preference’ as a new parameter of routing protocol instances, both in configuration and state data.
- o Renamed identity ’rt:standard-routing-instance’ to ’rt:default-routing-instance’.
- o Added next-hop lists to state data.
- o Added two cases for specifying next-hops indirectly - via a new RIB or a recursive list of next-hops.
- o Reorganized next-hop in static routes.
- o Removed all ’if-feature’ statements from state data.

E.3. Changes Between Versions -14 and -15

- o Removed all defaults from state data.
- o Removed default from ’cur-hop-limit’ in config.

E.4. Changes Between Versions -13 and -14

- o Removed dependency of ’connected-ribs’ on the ’multiple-ribs’ feature.
- o Removed default value of ’cur-hop-limit’ in state data.
- o Moved parts of descriptions and all references on IPv6 RA parameters from state data to configuration.
- o Added reference to RFC 6536 in the Security section.

E.5. Changes Between Versions -12 and -13

- o Wrote appendix about minimum implementation.
- o Remove "when" statement for IPv6 router interface state data - it was dependent on a config value that may not be present.

- o Extra container for the next-hop list.

- o Names rather than numeric ids are used for referring to list entries in state data.

- o Numeric ids are always declared as mandatory and unique. Their description states that they are ephemeral.

- o Descriptions of "name" keys in state data lists are required to be persistent.

- o

- o Removed "if-feature multiple-ribs;" from connected-ribs.

- o "rib-name" instead of "name" is used as the name of leafref nodes.

- o "next-hop" instead of "nexthop" or "gateway" used throughout, both in node names and text.

E.6. Changes Between Versions -11 and -12

- o Removed feature "advanced-router" and introduced two features instead: "multiple-ribs" and "multipath-routes".

- o Unified the keys of config and state versions of "routing-instance" and "rib" lists.

- o Numerical identifiers of state list entries are not keys anymore, but they are constrained using the "unique" statement.

- o Updated acknowledgements.

E.7. Changes Between Versions -10 and -11

- o Migrated address families from IANA enumerations to identities.

- o Terminology and node names aligned with the I2RS RIB model: router -> routing instance, routing table -> RIB.

- o Introduced uint64 keys for state lists: routing-instance, rib, route, nexthop.

- o Described the relationship between system-controlled and user-controlled list entries.

- o Feature "user-defined-routing-tables" changed into "advanced-router".

- o Made nexthop into a choice in order to allow for nexthop-list (I2RS requirement).

- o Added nexthop-list with entries having priorities (backup) and weights (load balancing).

- o Updated bibliography references.

E.8. Changes Between Versions -09 and -10

- o Added subtree for state data ("/routing-state").

- o Terms "system-controlled entry" and "user-controlled entry" defined and used.

- o New feature "user-defined-routing-tables". Nodes that are useful only with user-defined routing tables are now conditional.

- o Added grouping "router-id".

- o In routing tables, "source-protocol" attribute of routes now reports only protocol type, and its datatype is "identityref".

- o Renamed "main-routing-table" to "default-routing-table".

E.9. Changes Between Versions -08 and -09

- o Fixed "must" expresion for "connected-routing-table".

- o Simplified "must" expression for "main-routing-table".

- o Moved per-interface configuration of a new routing protocol under 'routing-protocol'. This also affects the 'example-rip' module.

E.10. Changes Between Versions -07 and -08

- o Changed reference from RFC6021 to RFC6021bis.

E.11. Changes Between Versions -06 and -07

- o The contents of <get-reply> in Appendix D was updated: "eth[01]" is used as the value of "location", and "forwarding" is on for both interfaces and both IPv4 and IPv6.

- o 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.

E.12. 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.

E.13. 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.

E.14. Changes Between Versions -03 and -04

- Changed "error-tag" for both RPC operations 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".

E.15. 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.

- o Assignment of interfaces to router instances needn’t be disjoint.
- o Route filters are now global.
- o Added "allow-all-route-filter" for symmetry.
- o Added Section 6 about interactions with "ietf-interfaces" and "ietf-ip".
- o Added "router-id" leaf.
- o Specified the names for IPv4/IPv6 unicast main routing tables.
- o Route parameter "last-modified" changed to "age".
- o Added container "recipient-routing-tables".

E.16. Changes Between Versions -01 and -02

- o Added module "ietf-ipv6-unicast-routing".
- o The example in Appendix D now uses IP addresses from blocks reserved for documentation.
- o Direct routes appear by default in the forwarding table.
- o Network layer interfaces must be assigned to a router instance. Additional interface configuration may be present.
- o The "when" statement is only used with "augment", "must" is used elsewhere.
- o Additional "must" statements were added.
- o The "route-content" grouping for IPv4 and IPv6 unicast now includes the material from the "ietf-routing" version via "uses rt:route-content".
- o Explanation of symbols in the tree representation of data model hierarchy.

E.17. Changes Between Versions -00 and -01

- o AFN/SAFI-independent stuff was moved to the "ietf-routing" module.
- o Typedefs for AFN and SAFI were placed in a separate "iana-afn-safi" module.

- o Names of some data nodes were changed, in particular "routing-process" is now "router".

- o The restriction of a single AFN/SAFI per router was lifted.

- o RPC operation "delete-route" was removed.

- o Illegal XPath references from "get-route" to the datastore were fixed.

- o Section "Security Considerations" was written.

Authors' Addresses

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

Acee Lindem
Cisco Systems

Email: acee@cisco.com