NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

L. Lhotka
CZ.NIC
A. Lindem
Cisco Systems
Y. Qu
Futurewei Technologies, Inc.
October 30, 2017

A YANG Data Model for Routing Management (NDMA Version) draft-acee-netmod-rfc8022bis-06

Abstract

This document contains a specification of three YANG modules and one submodule. Together they form the core routing data model that 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 control-plane protocols, route filters, and other functions. The core routing data model provides common building blocks for such extensions -- routes, Routing Information Bases (RIBs), and control-plane protocols.

This version of these YANG modules uses new names for these YANG models. The main difference from the first version is that this version fully conforms to the Network Management Datastore Architecture (NMDA). Consequently, this document obsoletes RFC 8022.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2018.

Table of Contents

1. Introduction

This document contains a specification of the following YANG modules:

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

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

- The "ietf-ipv6-unicast-routing" module augments the "ietf-routing" module with additional data specific to IPv6 unicast. Its submodule "ietf-ipv6-router-advertisements" also augments the "ietf-interfaces" [RFC7223] and "ietf-ip" [RFC7277] modules with IPv6 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 control-plane 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 various IETF working groups.

This version of these YANG modules uses new names for these YANG models. The main difference from the first version is that this version fully conforms to the Network Management Datastore Architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. Consequently, this document obsoletes RFC 8022 [RFC8022].

2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms are defined in [RFC6241]:

- client

- message

- protocol operation
- server

The following terms are defined in [RFC7950]:

- action
- augment
- configuration data
- container
- container with presence
- data model
- data node
- feature
- leaf
- list
- mandatory node
- module
- schema tree
- state data
- RPC (Remote Procedure Call) 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 5.2 for details.

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

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

2.2. Tree Diagrams

A simplified graphical representation of the complete data tree is presented in Appendix A, and similar diagrams of its various subtrees appear in the main text.

- Brackets "[" and "]" enclose list keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only), "-x" RPC operations or actions, and "-n" notifications.
- Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2.3. Prefixes in Data Node Names

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

Prefix	YANG module	Reference
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:

- 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).

- A simple IP routing system, such as one that uses only static routing, should be configurable in a simple way, ideally without any need to develop additional YANG modules.

- On the other hand, the core routing framework must allow for complicated implementations involving multiple Routing Information Bases (RIBs) and multiple control-plane protocols, as well as controlled redistributions of routing information.

- Because device vendors will want to map the data models built on this generic framework to their proprietary data models and configuration interfaces, the framework should be flexible enough to facilitate that 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 and one submodule. 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. The "ietf-ipv6-unicast-routing" module has a submodule, "ietf-ipv6-router-advertisements", that augments the "ietf-interfaces" [RFC7223] and "ietf-ip" [RFC7277]

modules with configuration variables for IPv6 router advertisements as required by [RFC4861].

Figures 1 shows abridged views of the hierarchies. See Appendix A for the complete data trees.

```
+--rw routing
   +--rw router-id?                yang:dotted-quad
   +--ro interfaces
   |  +--ro interface*   if:interface-ref
   +--rw control-plane-protocols
   |  +--rw control-plane-protocol* [type name]
   |     +--rw type             identityref
   |     +--rw name             string
   |     +--rw description?     string
   |     +--rw static-routes
   |        +--rw v4ur:ipv4
   |        |     ...
   |        +--rw v6ur:ipv6
   |              ...
   +--rw ribs
      +--rw rib* [name]
         +--rw name               string
         +--rw address-family?    identityref
         +--ro default-rib?       boolean {multiple-ribs}?
         +--ro routes
         |  +--ro route*
         |        ...
         +---x active-route
         |  +---w input
         |  |  +---w v4ur:destination-address?   inet:ipv4-address
         |  |  +---w v6ur:destination-address?   inet:ipv6-address
         |  +--ro output
         |        ...
         +--rw description?       string
```

Figure 1: Data Hierarchy

As can be seen from Figures 1, the core routing data model introduces several generic components of a routing framework: routes, RIBs containing lists of routes, and control-plane protocols. Section 5 describes these components in more detail.

4.1. System-Controlled and User-Controlled List Entries

The core routing data model defines several lists in the schema tree, such as "rib", that have to be populated with at least one entry in

any properly functioning device, and additional entries may be configured by a client.

In such a list, the server creates the required item as a so-called system-controlled entry in state data in the operational datastore [I-D.ietf-netmod-revised-datastores], i.e., inside read-only lists in the "routing" container.

An example can be seen in Appendix D: the "/routing/ribs/rib" list has two system-controlled entries named "ipv4-master" and "ipv6-master".

Additional entries may be created in the configuration by a client, 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 operational datastore and intended datastore [I-D.ietf-netmod-revised-datastores] have the same value of the list key.

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

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 client delets a system-controlled entry from the configuration list in the intended datastore, only the extra configuration specified in that entry is removed but the corresponding state data entry remains in the list in the operational datastore.

5. Basic Building Blocks

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

5.1. 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": address 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 outgoing interface and/or next-hop address(es), or a special operation to be performed with a packet.

Routes are primarily state data that appear as entries of RIBs (Section 5.2) 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 attributes defined for RIB routes.

5.2. Routing Information Base (RIB)

Every implementation of the core routing data model manages one or more Routing Information Bases (RIBs). A RIB is a list of routes complemented with administrative data. Each RIB contains 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/ribs/rib" in the operational datastore [I-D.ietf-netmod-revised-datastores]. The contents of RIBs are controlled and manipulated by control-plane protocol operations that may result in route additions, removals, and modifications. This also includes manipulations via the "static" and/or "direct" pseudo-protocols; see Section 5.3.1.

For every supported address family, exactly one RIB MUST be marked as the so-called default RIB to which control-plane protocols place their routes by default.

Simple router implementations that do not advertise the feature "multiple-ribs" will typically create one system-controlled RIB per supported address family and mark it as the default RIB.

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.

The following action (see Section 7.15 of [RFC7950]) is defined for the "rib" list:

- active-route -- return the active RIB route for the destination address that is specified as the action's input parameter.

5.3. Control-Plane Protocol

The core routing data model provides an open-ended framework for defining multiple control-plane protocol instances, e.g., for Layer 3 routing protocols. Each control-plane protocol instance MUST be assigned a type, which is an identity derived from the "rt:control-plane-protocol" base identity. The core routing data model defines two identities for the direct and static pseudo-protocols (Section 5.3.1).

Multiple control-plane protocol instances of the same type MAY be configured.

5.3.1. Routing Pseudo-Protocols

The core routing data model defines two special routing protocol types -- "direct" and "static". Both are in fact pseudo-protocols, which means that they are confined to the local device and do not exchange any routing information with adjacent routers.

Every implementation of the core routing data model MUST provide 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.

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.

5.3.2. Defining New Control-Plane Protocols

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

- A new identity MUST be defined for the control-plane protocol, and its base identity MUST be set to "rt:control-plane-protocol" or to an identity derived from "rt:control-plane-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 node

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

and possibly other places in the configuration, state data, notifications, and input/output parameters of actions or RPC operations.

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

By using a "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 (or derived from) the new protocol's identity.

It is also RECOMMENDED that protocol-specific data nodes be encapsulated in an appropriately named container with presence. Such a container may contain mandatory data nodes that are otherwise forbidden at the top level of an augment.

The above steps are implemented by the example YANG module for the Routing Information Protocol (RIP) in Appendix C.

5.4. Parameters of IPv6 Router Advertisements

YANG module "ietf-ipv6-router-advertisements" (Section 9.1), which is a submodule of the "ietf-ipv6-unicast-routing" module, augments the configuration and state data of IPv6 interfaces with definitions of the following variables as required by Section 6.2.1 of [RFC4861]:

- send-advertisements
- max-rtr-adv-interval
- min-rtr-adv-interval
- managed-flag
- other-config-flag
- link-mtu
- reachable-time
- retrans-timer
- cur-hop-limit

- default-lifetime

- prefix-list: a list of prefixes to be advertised.

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

  * valid-lifetime

  * on-link-flag

  * preferred-lifetime

  * autonomous-flag

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-router-advertisements" submodule therefore stipulates the former behavior with constant values.

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, then all routing and forwarding functions MUST be disabled on this interface.

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 and forwarding functions MUST be disabled on this interface.

/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 through 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 and forwarding functions MUST be disabled on this interface.

/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 through 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

```
<CODE BEGINS> file "ietf-routing@2017-10-30.yang"
module ietf-routing {
  yang-version "1.1";
  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 - Networking Modeling Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/netmod/>
       WG List:  <mailto:rtgwg@ietf.org>

       Editor:   Ladislav Lhotka
                 <mailto:lhotka@nic.cz>
                 Acee Lindem
                 <mailto:acee@cisco.com>
                 Yingzhen Qu
                 <mailto:yingzhen.qu@huawei.com>";

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

       Copyright (c) 2017 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.";
    reference "RFC XXXX";

    revision 2017-10-30 {
      description
        "Network Managment Datastore Architecture (NDMA) Revision";
      reference
        "RFC XXXX: A YANG Data Model for Routing Management
         (NDMA Version)";
    }

    revision 2016-11-04 {
         description
```

```
        "Initial revision.";
      reference
        "RFC 8022: A YANG Data Model for Routing Management";
    }

    /* Features */
    feature multiple-ribs {
      description
        "This feature indicates that the server supports user-defined
         RIBs.

         Servers that do not advertise this feature SHOULD provide
         exactly one system-controlled RIB per supported address family
         and make it also the default RIB.  This RIB then appears as an
         entry of the list /routing/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 the 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 control-plane-protocol {
    description
      "Base identity from which control-plane protocol identities are
       derived.";
  }

  identity routing-protocol {
    base control-plane-protocol;
    description
      "Identity from which Layer 3 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 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
          "Options for special next hops.";
      }
    }

    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 further cases will be added through
           augments from other modules.";
        case simple-next-hop {
          description
            "This case represents a simple next hop consisting of the
             next-hop address and/or outgoing interface.

             Modules for address families MUST augment this case with a
             leaf containing a next-hop address of that address
             family.";
          leaf outgoing-interface {
            type if:interface-ref;
            description
              "Name of the outgoing interface.";
          }
        }
        case special-next-hop {
          uses special-next-hop;
        }
        case next-hop-list {
          container next-hop-list {
            description
              "Container for multiple next-hops.";
            list next-hop {
              key "index";
              description
                "An entry of a next-hop list.

                 Modules for address families MUST augment this list
                 with a leaf containing a next-hop address of that
                 address family.";
              leaf index {
                type string;
                description
                  "A user-specified identifier utilized to uniquely
                   reference the next-hop entry in the next-hop list.
                   The value of this index has no semantic meaning
                   other than for referencing the entry.";
              }
              leaf outgoing-interface {
                type if:interface-ref;
```

```
                description
                  "Name of the outgoing interface.";
              }
            }
          }
        }
      }
    }

    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 further cases will be added through
           augments from other modules, e.g., for recursive
           next hops.";
        case simple-next-hop {
          description
            "This case represents a simple next hop consisting of the
             next-hop address and/or outgoing interface.

             Modules for address families MUST augment this case with a
             leaf containing a next-hop address of that address
             family.";
          leaf outgoing-interface {
            type if:interface-ref;
            description
              "Name of the outgoing interface.";
          }
        }
        case special-next-hop {
          uses special-next-hop;
        }
        case next-hop-list {
          container next-hop-list {
            description
              "Container for multiple next hops.";
            list next-hop {
              description
                "An entry of a next-hop list.

                 Modules for address families MUST augment this list
                 with a leaf containing a next-hop address of that
                 address family.";
```

```
            leaf outgoing-interface {
              type if:interface-ref;
              description
                "Name of the outgoing interface.";
            }
          }
        }
      }
    }

    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.";
      }
    }

    /* Configuration Data */

    container routing {
      description
        "Configuration parameters for the routing subsystem.";
      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.";
      }
      container interfaces {
        config "false";
        description
          "Network-layer interfaces used for routing.";
        leaf-list interface {
          type if:interface-ref;
          description
            "Each entry is a reference to the name of a configured
             network-layer interface.";
        }
      }
      container control-plane-protocols {
        description
          "Configuration of control-plane protocol instances.";
        list control-plane-protocol {
          key "type name";
          description
            "Each entry contains configuration of a control-plane
             protocol instance.";
          leaf type {
            type identityref {
              base control-plane-protocol;
            }
            description
              "Type of the control-plane protocol - an identity derived
               from the 'control-plane-protocol' base identity.";
          }
          leaf name {
            type string;
            description
              "An arbitrary name of the control-plane protocol
               instance.";
          }
          leaf description {
            type string;
            description
              "Textual description of the control-plane protocol
               instance.";
          }
          container static-routes {
            when "derived-from-or-self(../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 contains configuration for a RIB identified by
             the 'name' key.

             Entries having the same key as a system-controlled entry
             of the list /routing/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 {
            description
              "Address family of the RIB.

               It is mandatory for user-controlled RIBs.  For
               system-controlled RIBs it can be omitted; otherwise, it
               must match the address family of the corresponding state
               entry.";
            refine "address-family" {
              mandatory "false";
            }
          }

          leaf default-rib {
            if-feature "multiple-ribs";
```

```
        type boolean;
        default "true";
        config "false";
        description
          "This flag has the value of 'true' if and only if the RIB
           is the default RIB for the given address family.

           By default, control-plane protocols place their routes
           in the default RIBs.";
      }
      container routes {
        config "false";
        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;
        }
      }
      action active-route {
        description
          "Return the active RIB route that is used for the
           destination address.

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

               If no route exists in the RIB for the destination
```

```
                   address, no output is returned.

                   Address-family-specific modules MUST augment this
                   container with appropriate route contents.";
                 container next-hop {
                   description
                     "Route's next-hop attribute.";
                   uses next-hop-state-content;
                 }
                 uses route-metadata;
               }
             }
           }
           leaf description {
             type string;
             description
               "Textual description of the RIB.";
           }
         }
       }
     }

     /* Obsolete State Data */

     container routing-state {
       config false;
       status obsolete;
       description
         "State data of the routing subsystem.";
       uses router-id {
         status obsolete;
         description
           "Global router ID.

            It may be either configured or assigned algorithmically by
            the implementation.";
       }
       container interfaces {
         status obsolete;
         description
           "Network-layer interfaces used for routing.";
         leaf-list interface {
           type if:interface-state-ref;
           status obsolete;
           description
             "Each entry is a reference to the name of a configured
              network-layer interface.";
         }
```

```
      }
      container control-plane-protocols {
        status obsolete;
        description
          "Container for the list of routing protocol instances.";
        list control-plane-protocol {
          key "type name";
          status obsolete;
          description
            "State data of a control-plane protocol instance.

             An implementation MUST provide exactly one
             system-controlled instance of the 'direct'
             pseudo-protocol.  Instances of other control-plane
             protocols MAY be created by configuration.";
          leaf type {
            type identityref {
              base control-plane-protocol;
            }
            status obsolete;
            description
              "Type of the control-plane protocol.";
          }
          leaf name {
            type string;
            status obsolete;
            description
              "The name of the control-plane protocol instance.

               For system-controlled instances this name is
               persistent, i.e., it SHOULD NOT change across
               reboots.";
          }
        }
      }
      container ribs {
        status obsolete;
        description
          "Container for RIBs.";
        list rib {
          key "name";
          min-elements 1;
          status obsolete;
          description
            "Each entry represents a RIB identified by the 'name'
             key. All routes in a RIB MUST belong to the same address
             family.
```

```
           An implementation SHOULD provide one system-controlled
           default RIB for each supported address family.";
         leaf name {
           type string;
           status obsolete;
           description
             "The name of the RIB.";
         }
         uses address-family {
           status obsolete;
           description
             "The address family of the RIB.";
         }
         leaf default-rib {
           if-feature "multiple-ribs";
           type boolean;
           default "true";
           status obsolete;
           description
             "This flag has the value of 'true' if and only if the
              RIB is the default RIB for the given address family.

              By default, control-plane protocols place their routes
              in the default RIBs.";
         }
         container routes {
           status obsolete;
           description
             "Current content of the RIB.";
           list route {
             status obsolete;
             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;
               status obsolete;
               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 {
               status obsolete;
               description
                 "Route's next-hop attribute.";
```

```
        uses next-hop-state-content {
          status obsolete;
          description
            "Route's next-hop attribute operational state.";
        }
      }
      uses route-metadata {
        status obsolete;
        description
          "Route metadata.";
      }
    }
  }
  action active-route {
    status obsolete;
    description
      "Return the active RIB route that is used for the
       destination address.

       Address-family-specific modules MUST augment input
       parameters with a leaf named 'destination-address'.";
    output {
      container route {
        status obsolete;
        description
          "The active RIB route for the specified
           destination.

           If no route exists in the RIB for the destination
           address, no output is returned.

           Address-family-specific modules MUST augment this
           container with appropriate route contents.";
        container next-hop {
          status obsolete;
          description
            "Route's next-hop attribute.";
          uses next-hop-state-content {
            status obsolete;
            description
              "Active route state data.";
          }
        }
        uses route-metadata {
          status obsolete;
          description
          "Active route metadata.";
        }
```

```
              }
            }
          }
        }
      }
    }
  }
<CODE ENDS>
```

8. IPv4 Unicast Routing Management YANG Module

```
<CODE BEGINS> file "ietf-ipv4-unicast-routing@2017-10-14.yang"
module ietf-ipv4-unicast-routing {
  yang-version "1.1";
  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 - Networking Modeling Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:rtgwg@ietf.org>

     Editor:   Ladislav Lhotka
               <mailto:lhotka@nic.cz>
               Acee Lindem
               <mailto:acee@cisco.com>
               Yingzhen Qu
               <mailto:yingzhen.qu@huawei.com>";

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

     Copyright (c) 2017 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.";
    reference "RFC XXXX";

    revision 2017-10-14 {
      description
        "Network Managment Datastore Architecture (NDMA) Revision";
      reference
        "RFC XXXX: A YANG Data Model for Routing Management
         (NDMA Version)";
    }

    revision 2016-11-04 {
         description
           "Initial revision.";
         reference
           "RFC 8022: A YANG Data Model for Routing Management";
    }

    /* Identities */

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

    augment "/rt:routing/rt:ribs/rt:rib/rt:routes/rt:route" {
      when "derived-from-or-self(../../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/rt:ribs/rt:rib/rt:routes/rt:route/"
          + "rt:next-hop/rt:next-hop-options/rt:simple-next-hop" {
```

```
      when "derived-from-or-self(../../../rt:address-family, "
         + "'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      description
        "Augment 'simple-next-hop' case in IPv4 unicast routes.";
      leaf next-hop-address {
        type inet:ipv4-address;
        description
          "IPv4 address of the next hop.";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:routes/rt:route/"
          + "rt:next-hop/rt:next-hop-options/rt:next-hop-list/"
          + "rt:next-hop-list/rt:next-hop" {
      when "derived-from-or-self(../../../../../rt:address-family, "
         + "'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      description
        "This leaf augments the 'next-hop-list' case of IPv4 unicast
         routes.";
      leaf address {
        type inet:ipv4-address;
        description
          "IPv4 address of the next-hop.";
      }
    }

    augment
      "/rt:routing/rt:ribs/rt:rib/rt:active-route/rt:input" {
      when "derived-from-or-self(../rt:address-family, "
         + "'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast RIBs.";
      }
      description
        "This augment adds the input parameter of the 'active-route'
         action.";
      leaf destination-address {
        type inet:ipv4-address;
        description
          "IPv4 destination address.";
      }
    }
```

```
    augment "/rt:routing/rt:ribs/rt:rib/rt:active-route/"
          + "rt:output/rt:route" {
      when "derived-from-or-self(../../rt:address-family, "
         + "'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      description
        "This augment adds the destination prefix to the reply of the
         'active-route' action.";
      leaf destination-prefix {
        type inet:ipv4-prefix;
        description
          "IPv4 destination prefix.";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:active-route/"
          + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
          + "rt:simple-next-hop" {
      when "derived-from-or-self(../../../rt:address-family, "
         + "'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      description
        "Augment 'simple-next-hop' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv4-address;
        description
          "IPv4 address of the next hop.";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:active-route/"
          + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
          + "rt:next-hop-list/rt:next-hop-list/rt:next-hop" {
      when "derived-from-or-self(../../../../../rt:address-family, "
         + "'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      description
        "Augment 'next-hop-list' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv4-address;
```

```
        description
          "IPv4 address of the next hop.";
      }
    }

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-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";
          description
            "A 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/simple-next-hop" {
                description
                  "Augment 'simple-next-hop' case in IPv4 static
                   routes.";
                leaf next-hop-address {
                  type inet:ipv4-address;
                  description
                    "IPv4 address of the next hop.";
                }
              }
              augment "next-hop-options/next-hop-list/next-hop-list/"
                    + "next-hop" {
                description
                  "Augment 'next-hop-list' case in IPv4 static
                   routes.";
```

```
                leaf next-hop-address {
                  type inet:ipv4-address;
                  description
                    "IPv4 address of the next hop.";
                }
              }
            }
          }
        }
      }

      /* Obsolete State Data */

      augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
        when "derived-from-or-self(../../rt:address-family, "
             + "'v4ur:ipv4-unicast')" {
          description
            "This augment is valid only for IPv4 unicast.";
        }
        status obsolete;
        description
          "This leaf augments an IPv4 unicast route.";
        leaf destination-prefix {
          type inet:ipv4-prefix;
          status obsolete;
          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 "derived-from-or-self(
                ../../../rt:address-family, 'v4ur:ipv4-unicast')" {
          description
            "This augment is valid only for IPv4 unicast.";
        }
        status obsolete;
        description
          "Augment 'simple-next-hop' case in IPv4 unicast routes.";
        leaf next-hop-address {
          type inet:ipv4-address;
          status obsolete;
          description
            "IPv4 address of the next hop.";
        }
      }
      augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
```

```
          + "rt:next-hop/rt:next-hop-options/rt:next-hop-list/"
          + "rt:next-hop-list/rt:next-hop" {
    when "derived-from-or-self(../../../../../rt:address-family,
            'v4ur:ipv4-unicast')" {
      description
        "This augment is valid only for IPv4 unicast.";
    }
    status obsolete;
    description
      "This leaf augments the 'next-hop-list' case of IPv4 unicast
       routes.";
    leaf address {
      type inet:ipv4-address;
      status obsolete;
      description
        "IPv4 address of the next-hop.";
    }
  }
  augment "/rt:routing-state/rt:ribs/rt:rib/rt:active-route/"
          + "rt:input" {
    when "derived-from-or-self(../rt:address-family,
          'v4ur:ipv4-unicast')" {
      description
        "This augment is valid only for IPv4 unicast RIBs.";
    }
    status obsolete;
    description
      "This augment adds the input parameter of the 'active-route'
       action.";
    leaf destination-address {
      type inet:ipv4-address;
      status obsolete;
      description
        "IPv4 destination address.";
    }
  }
  augment "/rt:routing-state/rt:ribs/rt:rib/rt:active-route/"
          + "rt:output/rt:route" {
    when "derived-from-or-self(../../rt:address-family,
            'v4ur:ipv4-unicast')" {
      description
        "This augment is valid only for IPv4 unicast.";
    }
    status obsolete;
    description
      "This augment adds the destination prefix to the reply of the
       'active-route' action.";
    leaf destination-prefix {
```

```
        type inet:ipv4-prefix;
        status obsolete;
        description
          "IPv4 destination prefix.";
      }
    }
    augment "/rt:routing-state/rt:ribs/rt:rib/rt:active-route/"
            + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
            + "rt:simple-next-hop" {
      when "derived-from-or-self(../../../rt:address-family,
              'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      status obsolete;
      description
        "Augment 'simple-next-hop' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv4-address;
        status obsolete;
        description
          "IPv4 address of the next hop.";
      }
    }
    augment "/rt:routing-state/rt:ribs/rt:rib/rt:active-route/"
            + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
            + "rt:next-hop-list/rt:next-hop-list/rt:next-hop" {
      when "derived-from-or-self(../../../../../rt:address-family,
              'v4ur:ipv4-unicast')" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      status obsolete;
      description
        "Augment 'next-hop-list' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv4-address;
        status obsolete;
        description
          "IPv4 address of the next hop.";
      }
    }
  }
  <CODE ENDS>
```

9. IPv6 Unicast Routing Management YANG Module

```
<CODE BEGINS> file "ietf-ipv6-unicast-routing@2017-10-14.yang"
module ietf-ipv6-unicast-routing {
  yang-version "1.1";
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing";
  prefix "v6ur";

  import ietf-routing {
    prefix "rt";
  }

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

  include ietf-ipv6-router-advertisements {
    revision-date 2017-10-14;
  }

  organization
    "IETF NETMOD - Networking Modeling Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:rtgwg@ietf.org>

     Editor:   Ladislav Lhotka
               <mailto:lhotka@nic.cz>
               Acee Lindem
               <mailto:acee@cisco.com>
               Yingzhen Qu
               <mailto:yingzhen.qu@huawei.com>";

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

     Copyright (c) 2017 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.";
     reference "RFC XXXX";

     revision 2017-10-14 {
       description
         "Network Managment Datastore Architecture (NDMA) revision";
       reference
         "RFC XXXX: A YANG Data Model for Routing Management
          (NDMA Version)";
     }

     /* Identities */

     revision 2016-11-04 {
          description
            "Initial revision.";
          reference
            "RFC 8022: A YANG Data Model for Routing Management";
     }

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

     augment "/rt:routing/rt:ribs/rt:rib/rt:routes/rt:route" {
       when "derived-from-or-self(../../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/rt:ribs/rt:rib/rt:routes/rt:route/"
           + "rt:next-hop/rt:next-hop-options/rt:simple-next-hop" {
       when "derived-from-or-self(../../../rt:address-family, "
          + "'v6ur:ipv6-unicast')" {
         description
           "This augment is valid only for IPv6 unicast.";
```

```
      }
      description
        "Augment 'simple-next-hop' case in IPv6 unicast routes.";
      leaf next-hop-address {
        type inet:ipv6-address;
        description
          "IPv6 address of the next hop.";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:routes/rt:route/"
          + "rt:next-hop/rt:next-hop-options/rt:next-hop-list/"
          + "rt:next-hop-list/rt:next-hop" {
      when "derived-from-or-self(../../../../../rt:address-family, "
         + "'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This leaf augments the 'next-hop-list' case of IPv6 unicast
         routes.";
      leaf address {
        type inet:ipv6-address;
        description
          "IPv6 address of the next hop.";
      }
    }

    augment
      "/rt:routing/rt:ribs/rt:rib/rt:active-route/rt:input" {
      when "derived-from-or-self(../rt:address-family, "
         + "'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast RIBs.";
      }
      description
        "This augment adds the input parameter of the 'active-route'
         action.";
      leaf destination-address {
        type inet:ipv6-address;
        description
          "IPv6 destination address.";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:active-route/"
          + "rt:output/rt:route" {
      when "derived-from-or-self(../../rt:address-family, "
```

```
          + "'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "This augment adds the destination prefix to the reply of the
         'active-route' action.";
      leaf destination-prefix {
        type inet:ipv6-prefix;
        description
          "IPv6 destination prefix.";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:active-route/"
          + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
          + "rt:simple-next-hop" {
      when "derived-from-or-self(../../../rt:address-family, "
         + "'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "Augment 'simple-next-hop' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv6-address;
        description
          "IPv6 address of the next hop.";
      }
    }

    augment "/rt:routing/rt:ribs/rt:rib/rt:active-route/"
          + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
          + "rt:next-hop-list/rt:next-hop-list/rt:next-hop" {
      when "derived-from-or-self(../../../../../rt:address-family, "
         + "'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      description
        "Augment 'next-hop-list' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv6-address;
        description
          "IPv6 address of the next hop.";
      }
```

```
    }

    /* Configuration data */

    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-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";
          description
            "A 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/simple-next-hop" {
                description
                  "Augment 'simple-next-hop' case in IPv6 static
                   routes.";
                leaf next-hop-address {
                  type inet:ipv6-address;
                  description
                    "IPv6 address of the next hop.";
                }
              }
              augment "next-hop-options/next-hop-list/next-hop-list/"
                    + "next-hop" {
                description
                  "Augment 'next-hop-list' case in IPv6 static
                   routes.";
                leaf next-hop-address {
```

```
                type inet:ipv6-address;
                description
                  "IPv6 address of the next hop.";
              }
            }
          }
        }
      }
    }

    /* Obsolete State Data */

    augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
      when "derived-from-or-self(../../rt:address-family,
              'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      status obsolete;
      description
        "This leaf augments an IPv6 unicast route.";
      leaf destination-prefix {
        type inet:ipv6-prefix;
        status obsolete;
        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 "derived-from-or-self(../../../rt:address-family,
              'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      status obsolete;
      description
        "Augment 'simple-next-hop' case in IPv6 unicast routes.";
      leaf next-hop-address {
        type inet:ipv6-address;
        status obsolete;
        description
          "IPv6 address of the next hop.";
      }
    }
    augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
            + "rt:next-hop/rt:next-hop-options/rt:next-hop-list/"
```

```
          + "rt:next-hop-list/rt:next-hop" {
      when "derived-from-or-self(../../../../../rt:address-family,
              'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      status obsolete;
      description
        "This leaf augments the 'next-hop-list' case of IPv6 unicast
         routes.";
      leaf address {
        type inet:ipv6-address;
        status obsolete;
        description
          "IPv6 address of the next hop.";
      }
    }
    augment "/rt:routing-state/rt:ribs/rt:rib/"
            + "rt:active-route/rt:input" {
      when "derived-from-or-self(../rt:address-family,
              'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast RIBs.";
      }
      status obsolete;
      description
        "This augment adds the input parameter of the 'active-route'
         action.";
      leaf destination-address {
        type inet:ipv6-address;
        status obsolete;
        description
          "IPv6 destination address.";
      }
    }
    augment "/rt:routing-state/rt:ribs/rt:rib/rt:active-route/"
            + "rt:output/rt:route" {
      when "derived-from-or-self(../../rt:address-family,
            'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      status obsolete;
      description
        "This augment adds the destination prefix to the reply of the
         'active-route' action.";
      leaf destination-prefix {
        type inet:ipv6-prefix;
```

```
        status obsolete;
        description
          "IPv6 destination prefix.";
      }
    }
    augment "/rt:routing-state/rt:ribs/rt:rib/rt:active-route/"
            + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
            + "rt:simple-next-hop" {
      when "derived-from-or-self(../../../rt:address-family,
              'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      status obsolete;
      description
        "Augment 'simple-next-hop' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv6-address;
        status obsolete;
        description
          "IPv6 address of the next hop.";
      }
    }
    augment "/rt:routing-state/rt:ribs/rt:rib/rt:active-route/"
            + "rt:output/rt:route/rt:next-hop/rt:next-hop-options/"
            + "rt:next-hop-list/rt:next-hop-list/rt:next-hop" {
      when "derived-from-or-self(../../../../../rt:address-family,
              'v6ur:ipv6-unicast')" {
        description
          "This augment is valid only for IPv6 unicast.";
      }
      status obsolete;
      description
        "Augment 'next-hop-list' case in the reply to the
         'active-route' action.";
      leaf next-hop-address {
        type inet:ipv6-address;
        status obsolete;
        description
          "IPv6 address of the next hop.";
      }
    }
  }
  <CODE ENDS>
```

9.1. IPv6 Router Advertisements Submodule

```
<CODE BEGINS> file "ietf-ipv6-router-advertisements@2017-10-14.yang"
submodule ietf-ipv6-router-advertisements {
  yang-version "1.1";

  belongs-to ietf-ipv6-unicast-routing {
    prefix "v6ur";
  }

  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:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     WG Chair: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

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

     Editor:   Acee Lindem
               <mailto:acee@cisco.com>

     Editor:   Yingzhen Qu
               <mailto:yingzhen.qu@huawei.com>";

  description
    "This YANG module augments the 'ietf-ip' module with
     configuration and state data of IPv6 router advertisements.

     Copyright (c) 2017 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).

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and
        'OPTIONAL' in the module text are to be interpreted as
        described in RFC 2119.

        This version of this YANG module is part of RFC 8022;
        see the RFC itself for full legal notices.";

    reference
      "RFC 4861: Neighbor Discovery for IP version 6 (IPv6).";

    revision 2017-10-14 {
      description
        "Network Managment Datastore Architecture (NDMA) Revision";
      reference
        "RFC XXXX: A YANG Data Model for Routing Management
         (NDMA Version)";
    }

    revision 2016-11-04 {
         description
           "Initial revision.";
         reference
           "RFC 8022: A YANG Data Model for Routing Management";
    }

    augment "/if:interfaces/if:interface/ip:ipv6" {
      description
        "Augment interface configuration with parameters of IPv6
         router advertisements.";
      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
           default indicates that the router is not to be used as
           a 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
              "Either the prefix is explicitly removed from the
               set of advertised prefixes, or the parameters with
               which it is advertised are 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.";
                }
              }
            }
          }
        }
      }
    }

    /* Obsolete State Data */

    augment "/if:interfaces-state/if:interface/ip:ipv6" {
```

```
      status obsolete;
      description
        "Augment interface state data with parameters of IPv6 router
         advertisements.";
      container ipv6-router-advertisements {
        status obsolete;
        description
          "Parameters of IPv6 Router Advertisements.";
        leaf send-advertisements {
          type boolean;
          status obsolete;
          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";
          status obsolete;
          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";
          status obsolete;
          description
            "The minimum time allowed between sending unsolicited
             multicast Router Advertisements from the interface.";
        }
        leaf managed-flag {
          type boolean;
          status obsolete;
          description
            "The value that is placed in the 'Managed address
             configuration' flag field in the Router Advertisement.";
        }
        leaf other-config-flag {
          type boolean;
          status obsolete;
          description
            "The value that is placed in the 'Other configuration' flag
```

```
            field in the Router Advertisement.";
        }
        leaf link-mtu {
          type uint32;
          status obsolete;
          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";
          status obsolete;
          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";
          status obsolete;
          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;
          status obsolete;
          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";
          status obsolete;
          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 {
          status obsolete;
          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";
            status obsolete;
            description
              "Advertised prefix entry and its parameters.";
            leaf prefix-spec {
              type inet:ipv6-prefix;
              status obsolete;
              description
                "IPv6 address prefix.";
            }
            leaf valid-lifetime {
              type uint32;
              units "seconds";
              status obsolete;
              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.

                 An implementation SHOULD keep this value constant in
                 consecutive advertisements except when it is
                 explicitly changed in configuration.";
            }
            leaf on-link-flag {
              type boolean;
              status obsolete;
              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";
              status obsolete;
              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.

                 An implementation SHOULD keep this value constant in
                 consecutive advertisements except when it is
                 explicitly changed in configuration.";
            }
            leaf autonomous-flag {
              type boolean;
              status obsolete;
              description
                "The value that is placed in the Autonomous Flag field
                 in the Prefix Information option.";
            }
          }
        }
      }
    }
  }
<CODE ENDS>
```

10. IANA Considerations

[RFC8022] registered 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.

[RFC8022] registered 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 8022

```
Name:         ietf-ipv4-unicast-routing
Namespace:    urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing
Prefix:       v4ur
Reference:    RFC 8022

Name:         ietf-ipv6-unicast-routing
Namespace:    urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing
Prefix:       v6ur
Reference:    RFC 8022
```

This document registers the following YANG submodule in the "YANG Module Names" registry [RFC6020]:

```
Name:         ietf-ipv6-router-advertisements
Module:       ietf-ipv6-unicast-routing
Reference:    RFC 8022
```

11. Security Considerations

The YANG module defined in this memo is 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].

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/routing/control-plane-protocols/control-plane-protocol: This list specifies the control-plane protocols configured on a device.

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

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/routing/control-plane-protocols/control-plane-protocol: This list specifies the control-plane protocols configured on a device. Refer to the control plane models for a list of sensitive information.

/routing/ribs/rib: This list specifies the RIB and their contents for the device. Access to this information may disclose the network topology and or other information.

12. References

12.1. Normative References

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

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

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

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

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <https://www.rfc-editor.org/info/rfc7277>.

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

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

12.2. Informative References

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <https://www.rfc-editor.org/info/rfc6087>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

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

[I-D.ietf-netmod-revised-datastores] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-05 (work in progress), October 2017.

Appendix A. The Complete Data Trees

This appendix presents the complete tree of the core routing data model. See Section 2.2 for an explanation of the symbols used. The data type of every leaf node is shown near the right end of the corresponding line.

```
module: ietf-routing
+--rw routing
   +--rw router-id?                yang:dotted-quad
   +--ro interfaces
   |  +--ro interface*   if:interface-ref
   +--rw control-plane-protocols
   |  +--rw control-plane-protocol* [type name]
   |     +--rw type             identityref
   |     +--rw name             string
   |     +--rw description?     string
   |     +--rw static-routes
   |        +--rw v4ur:ipv4
   |        |  +--rw v4ur:route* [destination-prefix]
   |        |     +--rw v4ur:destination-prefix    inet:ipv4-prefix
   |        |     +--rw v4ur:description?          string
   |        |     +--rw v4ur:next-hop
   |        |        +--rw (v4ur:next-hop-options)
   |        |           +--:(v4ur:simple-next-hop)
   |        |           |  +--rw v4ur:outgoing-interface?
   |        |           |            if:interface-ref
   |        |           |  +--rw v4ur:next-hop-address?
   |        |           |            inet:ipv4-address
   |        |           +--:(v4ur:special-next-hop)
   |        |           |  +--rw v4ur:special-next-hop?  enumeration
   |        |           +--:(v4ur:next-hop-list)
   |        |              +--rw v4ur:next-hop-list
   |        |                 +--rw v4ur:next-hop* [index]
   |        |                    +--rw v4ur:index            string
   |        |                    +--rw v4ur:outgoing-interface?
   |        |                              if:interface-ref
   |        |                    +--rw v4ur:next-hop-address?
   |        |                              inet:ipv4-address
   |        +--rw v6ur:ipv6
   |           +--rw v6ur:route* [destination-prefix]
   |              +--rw v6ur:destination-prefix    inet:ipv6-prefix
   |              +--rw v6ur:description?          string
   |              +--rw v6ur:next-hop
   |                 +--rw (v6ur:next-hop-options)
   |                    +--:(v6ur:simple-next-hop)
   |                    |  +--rw v6ur:outgoing-interface?
   |                    |            if:interface-ref
```

```
      |                    |  +--rw v6ur:next-hop-address?
      |                    |             inet:ipv6-address
      |                    +--:(v6ur:special-next-hop)
      |                    |  +--rw v6ur:special-next-hop?  enumeration
      |                    +--:(v6ur:next-hop-list)
      |                       +--rw v6ur:next-hop-list
      |                          +--rw v6ur:next-hop* [index]
      |                             +--rw v6ur:index             string
      |                             +--rw v6ur:outgoing-interface?
      |                                       if:interface-ref
      |                             +--rw v6ur:next-hop-address?
      |                                       inet:ipv6-address
      +--rw ribs
         +--rw rib* [name]
            +--rw name               string
            +--rw address-family?    identityref
            +--ro default-rib?       boolean {multiple-ribs}?
            +--ro routes
            |  +--ro route*
            |     +--ro route-preference?          route-preference
            |     +--ro next-hop
            |     |  +--ro (next-hop-options)
            |     |     +--:(simple-next-hop)
            |     |     |  +--ro outgoing-interface?
            |     |     |  |       if:interface-ref
            |     |     |  +--ro v4ur:next-hop-address?
            |     |     |  |       inet:ipv4-address
            |     |     |  +--ro v6ur:next-hop-address?
            |     |     |             inet:ipv6-address
            |     |     +--:(special-next-hop)
            |     |     |  +--ro special-next-hop?        enumeration
            |     |     +--:(next-hop-list)
            |     |        +--ro next-hop-list
            |     |           +--ro next-hop*
            |     |              +--ro outgoing-interface?
            |     |              |       if:interface-ref
            |     |              +--ro v4ur:address?
            |     |              |       inet:ipv4-address
            |     |              +--ro v6ur:address?
            |     |                       inet:ipv6-address
            |     +--ro source-protocol           identityref
            |     +--ro active?                   empty
            |     +--ro last-updated?             yang:date-and-time
            |     +--ro v4ur:destination-prefix?  inet:ipv4-prefix
            |     +--ro v6ur:destination-prefix?  inet:ipv6-prefix
            +---x active-route
            |  +---w input
            |  |  +---w v4ur:destination-address?   inet:ipv4-address
```

```
          |  |  +---w v6ur:destination-address?   inet:ipv6-address
          |  +--ro output
          |     +--ro route
          |        +--ro next-hop
          |        |  +--ro (next-hop-options)
          |        |     +--:(simple-next-hop)
          |        |     |  +--ro outgoing-interface?
          |        |     |  |       if:interface-ref
          |        |     |  +--ro v4ur:next-hop-address?
          |        |     |  |          inet:ipv4-address
          |        |     |  +--ro v6ur:next-hop-address?
          |        |     |             inet:ipv6-address
          |        |     +--:(special-next-hop)
          |        |     |  +--ro special-next-hop?      enumeration
          |        |     +--:(next-hop-list)
          |        |        +--ro next-hop-list
          |        |           +--ro next-hop*
          |        |              +--ro outgoing-interface?
          |        |              |       if:interface-ref
          |        |              +--ro v4ur:next-hop-address?
          |        |              |       inet:ipv4-address
          |        |              +--ro v6ur:next-hop-address?
          |        |                      inet:ipv6-address
          |        +--ro source-protocol          identityref
          |        +--ro active?                  empty
          |        +--ro last-updated?            yang:date-and-time
          |        +--ro v4ur:destination-prefix? inet:ipv4-prefix
          |        +--ro v6ur:destination-prefix? inet:ipv6-prefix
          +--rw description?      string
module: ietf-ipv6-unicast-routing
  augment /if:interfaces/if:interface/ip:ipv6:
+--rw ipv6-router-advertisements
   +--rw send-advertisements?    boolean
   +--rw max-rtr-adv-interval?   uint16
   +--rw min-rtr-adv-interval?   uint16
   +--rw managed-flag?           boolean
   +--rw other-config-flag?      boolean
   +--rw link-mtu?               uint32
   +--rw reachable-time?         uint32
   +--rw retrans-timer?          uint32
   +--rw cur-hop-limit?          uint8
   +--rw default-lifetime?       uint16
   +--rw prefix-list
      +--rw prefix* [prefix-spec]
         +--rw prefix-spec           inet:ipv6-prefix
         +--rw (control-adv-prefixes)?
            +--:(no-advertise)
            |  +--rw no-advertise?         empty
```

```
              +--:(advertise)
                 +--rw valid-lifetime?       uint32
                 +--rw on-link-flag?         boolean
                 +--rw preferred-lifetime?   uint32
                 +--rw autonomous-flag?      boolean
```

Appendix B. Minimum Implementation

Some parts and options of the core routing model, such as user-defined RIBs, 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 does not support the feature "multiple-ribs". This means that a single system-controlled RIB is available for each supported address family -- IPv4, IPv6, or both. These RIBs are also the default RIBs. No user-controlled RIBs are allowed.

In addition to the mandatory instance of the "direct" pseudo-protocol, a minimum implementation should support configuring instance(s) of the "static" pseudo-protocol.

For hosts that are never intended to act as routers, the ability to turn on sending IPv6 router advertisements (Section 5.4) should be removed.

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

Appendix C. Example: Adding a New Control-Plane Protocol

This appendix demonstrates how the core routing data model can be extended to support a new control-plane protocol. The YANG module "example-rip" shown below is intended as an illustration rather than a real definition of a data model for the Routing Information Protocol (RIP). For the sake of brevity, this module does not obey all the guidelines specified in [RFC6087]. See also Section 5.3.2.

```
module example-rip {

  yang-version "1.1";

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

  prefix "rip";
```

```
  import ietf-interfaces {
    prefix "if";
  }

  import ietf-routing {
    prefix "rt";
  }

  identity rip {
    base rt:routing-protocol;
    description
      "Identity for the Routing Information Protocol (RIP).";
  }

  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.,
         autonomous system (AS) number.";
    }
  }

  augment "/rt:routing/rt:ribs/rt:rib/rt:routes/rt:route" {
    when "derived-from-or-self(rt:source-protocol, 'rip:rip')" {
      description
        "This augment is only valid for a route whose source
         protocol is RIP.";
    }
    description
      "RIP-specific route attributes.";
    uses route-content;
  }

  augment "/rt:routing/rt:ribs/rt:rib/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:control-plane-protocols/"
          + "rt:control-plane-protocol" {
      when "derived-from-or-self(rt:type,'rip:rip')" {
        description
          "This augment is only valid for a routing protocol instance
           of type 'rip'.";
      }
      container rip {
        presence "RIP configuration";
        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 if:interface-ref;
            }
            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. Data Tree Example

This section contains an example of an instance data tree in the JSON encoding [RFC7951], containing both configuration and state data. The data conforms to a data model that is defined by the following YANG library specification [RFC7895]:

```
{
  "ietf-yang-library:modules-state": {
    "module-set-id": "c2e1f54169aa7f36e1a6e8d0865d441d3600f9c4",
    "module": [
      {
        "name": "ietf-routing",
        "revision": "2017-09-13",
        "feature": [
          "multiple-ribs",
          "router-id"
        ],
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ipv4-unicast-routing",
        "revision": "2017-09-13",
        "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ipv6-unicast-routing",
        "revision": "2017-09-13",
        "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing-3",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-interfaces",
        "revision": "2014-05-08",
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-interfaces",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-inet-types",
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types",
```

```
          "revision": "2013-07-15",
          "conformance-type": "import"
        },
        {
          "name": "ietf-yang-types",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "revision": "2013-07-15",
          "conformance-type": "import"
        },
        {
          "name": "iana-if-type",
          "namespace": "urn:ietf:params:xml:ns:yang:iana-if-type",
          "revision": "",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        }
      ]
    }
  }
```

A simple network setup as shown in Figure 2 is assumed: router "A" uses static default routes with the "ISP" router as the next hop. IPv6 router advertisements are configured only on the "eth1" interface and disabled on the upstream "eth0" interface.

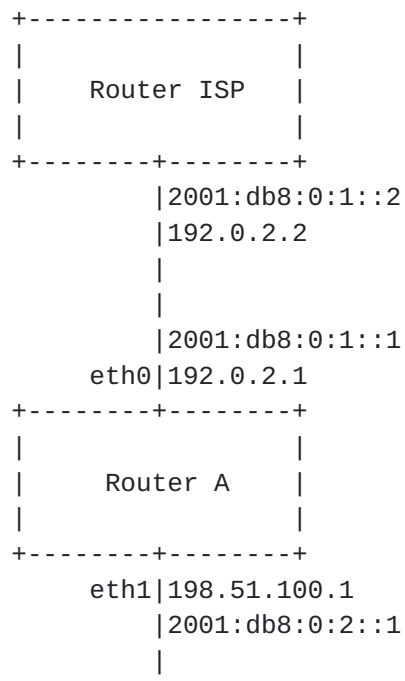

Figure 2: Example of Network Configuration

The instance data tree could then be as follows:

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "description": "Uplink to ISP.",
        "ietf-ip:ipv4": {
          "address": [
            {
              "ip": "192.0.2.1",
              "prefix-length": 24
            }
          ],
          "forwarding": true
        },
        "ietf-ip:ipv6": {
          "address": [
            {
              "ip": "2001:0db8:0:1::1",
              "prefix-length": 64
            }
          ],
          "forwarding": true,
```

```
            "autoconf": {
              "create-global-addresses": false
            }
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "description": "Interface to the internal network.",
          "ietf-ip:ipv4": {
            "address": [
              {
                "ip": "198.51.100.1",
                "prefix-length": 24
              }
            ],
            "forwarding": true
          },
          "ietf-ip:ipv6": {
            "address": [
              {
                "ip": "2001:0db8:0:2::1",
                "prefix-length": 64
              }
            ],
            "forwarding": true,
            "autoconf": {
              "create-global-addresses": false
            },
            "ietf-ipv6-unicast-routing:
                ipv6-router-advertisements": {
              "send-advertisements": true
            }
          }
        }
      ]
    },
    "ietf-interfaces:interfaces-state": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "phys-address": "00:0C:42:E5:B1:E9",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2015-10-24T17:11:27+02:00"
          },
          "ietf-ip:ipv4": {
```

```
            "forwarding": true,
            "mtu": 1500,
            "address": [
              {
                "ip": "192.0.2.1",
                "prefix-length": 24
              }
            ]
          },
          "ietf-ip:ipv6": {
            "forwarding": true,
            "mtu": 1500,
            "address": [
              {
                "ip": "2001:0db8:0:1::1",
                "prefix-length": 64
              }
            ],
            "ietf-ipv6-unicast-routing:
               ipv6-router-advertisements": {
              "send-advertisements": false
            }
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "phys-address": "00:0C:42:E5:B1:EA",
          "oper-status": "up",
          "statistics": {
            "discontinuity-time": "2015-10-24T17:11:29+02:00"
          },
          "ietf-ip:ipv4": {
            "forwarding": true,
            "mtu": 1500,
            "address": [
              {
                "ip": "198.51.100.1",
                "prefix-length": 24
              }
            ]
          },
          "ietf-ip:ipv6": {
            "forwarding": true,
            "mtu": 1500,
            "address": [
              {
                "ip": "2001:0db8:0:2::1",
```

```
                "prefix-length": 64
              }
            ],
            "ietf-ipv6-unicast-routing:
               ipv6-router-advertisements": {
              "send-advertisements": true,
              "prefix-list": {
                "prefix": [
                  {
                    "prefix-spec": "2001:db8:0:2::/64"
                  }
                ]
              }
            }
          }
        }
      ]
    },
    "ietf-routing:routing": {
      "router-id": "192.0.2.1",
      "control-plane-protocols": {
        "control-plane-protocol": [
          {
            "type": "ietf-routing:static",
            "name": "st0",
            "description":
              "Static routing is used for the internal network.",
            "static-routes": {
              "ietf-ipv4-unicast-routing:ipv4": {
                "route": [
                  {
                    "destination-prefix": "0.0.0.0/0",
                    "next-hop": {
                      "next-hop-address": "192.0.2.2"
                    }
                  }
                ]
              },
              "ietf-ipv6-unicast-routing:ipv6": {
                "route": [
                  {
                    "destination-prefix": "::/0",
                    "next-hop": {
                      "next-hop-address": "2001:db8:0:1::2"
                    }
                  }
                ]
              }
```

```
            }
          }
        ]
      }
        "ribs": {
      "rib": [
        {
          "name": "ipv4-master",
          "address-family":
            "ietf-ipv4-unicast-routing:ipv4-unicast",
          "default-rib": true,
          "routes": {
            "route": [
              {
                "ietf-ipv4-unicast-routing:destination-prefix":
                  "192.0.2.1/24",
                "next-hop": {
                  "outgoing-interface": "eth0"
                },
                "route-preference": 0,
                "source-protocol": "ietf-routing:direct",
                "last-updated": "2015-10-24T17:11:27+02:00"
              },
              {
                "ietf-ipv4-unicast-routing:destination-prefix":
                  "198.51.100.0/24",
                "next-hop": {
                  "outgoing-interface": "eth1"
                },
                "source-protocol": "ietf-routing:direct",
                "route-preference": 0,
                "last-updated": "2015-10-24T17:11:27+02:00"
              },
              {
                "ietf-ipv4-unicast-routing:destination-prefix":
                  "0.0.0.0/0",
                "source-protocol": "ietf-routing:static",
                "route-preference": 5,
                "next-hop": {
                  "ietf-ipv4-unicast-routing:next-hop-address":
                    "192.0.2.2"
                },
                "last-updated": "2015-10-24T18:02:45+02:00"
              }
            ]
          }
        },
        {
```

```
            "name": "ipv6-master",
            "address-family":
              "ietf-ipv6-unicast-routing:ipv6-unicast",
            "default-rib": true,
            "routes": {
              "route": [
                {
                  "ietf-ipv6-unicast-routing:destination-prefix":
                    "2001:db8:0:1::/64",
                  "next-hop": {
                    "outgoing-interface": "eth0"
                  },
                  "source-protocol": "ietf-routing:direct",
                  "route-preference": 0,
                  "last-updated": "2015-10-24T17:11:27+02:00"
                },
                {
                  "ietf-ipv6-unicast-routing:destination-prefix":
                    "2001:db8:0:2::/64",
                  "next-hop": {
                    "outgoing-interface": "eth1"
                  },
                  "source-protocol": "ietf-routing:direct",
                  "route-preference": 0,
                  "last-updated": "2015-10-24T17:11:27+02:00"
                },
                {
                  "ietf-ipv6-unicast-routing:destination-prefix":
                    "::/0",
                  "next-hop": {
                    "ietf-ipv6-unicast-routing:next-hop-address":
                      "2001:db8:0:1::2"
                  },
                  "source-protocol": "ietf-routing:static",
                  "route-preference": 5,
                  "last-updated": "2015-10-24T18:02:45+02:00"
                }
              ]
            }
          }
        ]
      }
    },
  }
```

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, Yingzhen Qu, Bruno Rijsman, Juergen Schoenwaelder, Phil Shafer, Dave Thaler, Yi Yang, Derek Man-Kit Yeung, and Jeffrey Zhang for their helpful comments and suggestions.

Authors' Addresses

Ladislav Lhotka
CZ.NIC

EMail: lhotka@nic.cz

Acee Lindem
Cisco Systems

EMail: acee@cisco.com

Yingzhen Qu
Futurewei Technologies, Inc.
2330 Central Expressway
Santa Clara CA 95050
USA

EMail: yingzhen.qu@huawei.com