A YANG Data Model for Routing Configuration
draft-ietf-netmod-routing-cfg-05

Abstract

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

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document contains a specification of the following YANG modules:

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

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

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

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

2. Terminology and Notation

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

The following terms are defined in [RFC6241]:

- client
- message
- protocol operation
- server

The following terms are defined in [RFC6020]:

- augment
- configuration data
- container
- data model
- data node
- data type
- identity
- mandatory node
- module
- operational state data
- prefix
- RPC operation

2.1. Glossary of New Terms

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

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

direct route: a route to a directly connected network.

2.2. Prefixes in Data Node Names

In this document, names of data nodes, RPC methods and other data model objects are used mostly 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
ianaaf	iana-afn-safi	[IANA-IF-AF]
if	ietf-interfaces	[YANG-IF]
ip	ietf-ip	[YANG-IP]
rip	example-rip	Appendix A
rt	ietf-routing	Section 6
v4ur	ietf-ipv4-unicast-routing	Section 7
v6ur	ietf-ipv6-unicast-routing	Section 8
yang	ietf-yang-types	[RFC6021]
inet	ietf-inet-types	[RFC6021]

Table 1: Prefixes and corresponding YANG modules

3. Objectives

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

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

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

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

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

4. The Design of the Core Routing Data Model

The core routing data model consists of three YANG modules. The first module, "ietf-routing", defines the generic components of a routing system. The other two modules, "ietf-ipv4-unicast-routing" and "ietf-ipv6-unicast-routing", augment the "ietf-routing" module with additional data nodes that are needed for IPv4 and IPv6 unicast routing, respectively. The combined data hierarchy is shown in Figure 1, where brackets enclose list keys, "rw" means configuration, "ro" operational state data, and "?" means optional node. Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

```
+--rw routing
   +--rw router [name]
   |  +--rw name
   |  +--rw type?
   |  +--rw enabled?
   |  +--rw router-id?
   |  +--rw description?
   |  +--rw main-routing-tables
   |  |  +--rw main-routing-table [address-family safi]
   |  |     +--rw address-family
   |  |     +--rw safi
   |  |     +--rw name?
   |  +--rw interfaces
   |  |  +--rw interface [name]
   |  |     +--rw name
   |  |     +--rw v6ur:ipv6-router-advertisements
   |  |        +--rw v6ur:send-advertisements?
   |  |        +--rw v6ur:max-rtr-adv-interval?
   |  |        +--rw v6ur:min-rtr-adv-interval?
   |  |        +--rw v6ur:managed-flag?
   |  |        +--rw v6ur:other-config-flag?
   |  |        +--rw v6ur:link-mtu?
   |  |        +--rw v6ur:reachable-time?
   |  |        +--rw v6ur:retrans-timer?
   |  |        +--rw v6ur:cur-hop-limit?
   |  |        +--rw v6ur:default-lifetime?
   |  |        +--rw v6ur:prefix-list
   |  |           +--rw v6ur:prefix [prefix-spec]
   |  |              +--rw v6ur:prefix-spec
   |  |              +--rw (control-adv-prefixes)?
   |  |                 +--:(no-advertise)
   |  |                 |  +--rw v6ur:no-advertise?
   |  |                 +--:(advertise)
   |  |                    +--rw v6ur:valid-lifetime?
   |  |                    +--rw v6ur:on-link-flag?
```

```
|  |                    +--rw v6ur:preferred-lifetime?
|  |                    +--rw v6ur:autonomous-flag?
|  +--rw routing-protocols
|     +--rw routing-protocol [name]
|        +--rw name
|        +--rw description?
|        +--rw enabled?
|        +--rw type
|        +--rw connected-routing-tables
|        |  +--rw connected-routing-table [name]
|        |     +--rw name
|        |     +--rw import-filter?
|        |     +--rw export-filter?
|        +--rw static-routes
|           +--rw v4ur:ipv4
|           |  +--rw v4ur:route [id]
|           |     +--rw v4ur:id
|           |     +--rw v4ur:description?
|           |     +--rw v4ur:outgoing-interface?
|           |     +--rw v4ur:dest-prefix
|           |     +--rw v4ur:next-hop?
|           +--rw v6ur:ipv6
|              +--rw v6ur:route [id]
|                 +--rw v6ur:id
|                 +--rw v6ur:description?
|                 +--rw v6ur:outgoing-interface?
|                 +--rw v6ur:dest-prefix
|                 +--rw v6ur:next-hop?
+--rw routing-tables
|  +--rw routing-table [name]
|     +--rw name
|     +--rw address-family
|     +--rw safi
|     +--rw description?
|     +--ro routes
|     |  +--ro route
|     |     +--ro outgoing-interface?
|     |     +--ro source-protocol
|     |     +--ro last-updated?
|     |     +--ro v4ur:dest-prefix?
|     |     +--ro v4ur:next-hop?
|     |     +--ro v6ur:dest-prefix?
|     |     +--ro v6ur:next-hop?
|     +--rw recipient-routing-tables
|        +--rw recipient-routing-table [name]
|           +--rw name
|           +--rw filter?
|
```

```
      |
      +--rw route-filters
         +--rw route-filter [name]
            +--rw name
            +--rw description?
            +--rw type
```

Figure 1: Data hierarchy of the core routing data model.

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

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

```
   +--------+
   | direct |    +---+    +--------------+    +---+    +--------------+
   | routes |--->| F |--->|              |<---| F |<---|              |
   +--------+    +---+    |    main      |    +---+    |  additional  |
                          |   routing    |             |   routing    |
   +--------+    +---+    |    table     |    +---+    |    table     |
   | static |--->| F |--->|              |--->| F |--->|              |
   | routes |    +---+    +--------------+    +---+    +--------------+
   +--------+                 ^      |                     ^      |
                              |      v                     |      v
                            +---+  +---+                 +---+  +---+
                            | F |  | F |                 | F |  | F |
                            +---+  +---+                 +---+  +---+
                              ^      |                     ^      |
                              |      v                     |      v
                            +----------+                 +----------+
                            | routing  |                 | routing  |
                            | protocol |                 | protocol |
                            +----------+                 +----------+
```

Figure 2: Example setup of the routing subsystem

The example in Figure 2 shows a typical (though certainly not the only possible) organization of a more complex routing subsystem for a single address family. Several of its features are worth mentioning:

- o Along with the main routing table, which must always be present, an additional routing table is configured.

o Each routing protocol instance, including the "static" and "direct" pseudo-protocols, is connected to one routing table with which it can exchange routes (in both directions, except for the "static" and "direct" pseudo-protocols).

o Routing tables may also be connected to each other and exchange routes in either direction (or both).

o Route exchanges along all connections may be controlled by means of route filters, denoted by "F" in Figure 2.

4.1. Router

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

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

An implementation MAY pose restrictions on allowed router types and on the number of supported instances for each type. For example, a simple router implementation may support only one router instance of the default type "standard-router".

Each network layer interface has to be assigned to one or more router instances in order to be able to participate in packet forwarding, routing protocols and other operations of those router instances. The assignment is accomplished by creating a corresponding entry in the list of router interfaces ("rt:interface"). The key of the list entry MUST be the name of a configured network layer interface, i.e., the value of a node /if:interfaces/if:interface/if:name defined in the "ietf-interfaces" module [YANG-IF].

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

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

4.1.1. Configuration of IPv6 Router Interfaces

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

- send-advertisements,
- max-rtr-adv-interval,
- min-rtr-adv-interval,
- managed-flag,
- other-config-flag,
- link-mtu,
- reachable-time,
- retrans-timer,
- cur-hop-limit,
- default-lifetime,
- prefix-list: a list of prefixes to be advertised. The following parameters are associated with each prefix in the list:
  * valid-lifetime,
  * on-link-flag,
  * preferred-lifetime,
  * autonomous-flag.

The definitions and descriptions of the above parameters can be found in the text of the module "ietf-ipv6-unicast-routing" (Section 8).

NOTES:

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

2. The original specification [RFC4861] allows the implementations to decide whether the "valid-lifetime" and "preferred-lifetime"

parameters remain the same in consecutive advertisements, or decrement in real time. However, the latter behavior seems problematic because the values might be reset again to the (higher) configured values after a configuration is reloaded. Moreover, no implementation is known to use the decrementing behavior. The "ietf-ipv6-unicast-routing" module therefore assumes the former behavior with constant values.

4.2. Routes

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

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

- "next-hop": IP address of an adjacent router or host to which packets with destination addresses belonging to "destination-prefix" should be sent.

- "outgoing-interface": network interface that should be used for sending packets with destination addresses belonging to "destination-prefix".

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

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

4.3. Routing Tables

Routing tables are lists of routes complemented with administrative data, namely:

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

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

Each routing table may contain only routes of the same address family. Address family information consists of two parameters -

"address-family" and "safi" (Subsequent Address Family Identifier, SAFI). The permitted values for these two parameters are defined by IANA and represented using YANG enumeration types "ianaaf:address-family" and "ianaaf:subsequent-address-family" [IANA-IF-AF].

In the core routing data model, the "routing-table" node represents configuration while the descendant list of routes is defined as operational state data. The contents of route lists are controlled and manipulated by routing protocol operations which may result in route additions, removals and modifications. This also includes manipulations via the "static" and/or "direct" pseudo-protocols, see Section 4.4.1.

One or more routing tables MUST be configured for each address family supported by the server. Each router instance MUST designate, for every address family that the router instance supports, exactly one routing table as its main routing table. This is accomplished by creating an entry in the "main-routing-table" list, which contains a reference to the routing table that is selected as main.

Main routing tables serve the following purposes:

- The router instance always installs direct routes for an address family to that address family's main routing table.

- By default, a routing protocol SHOULD be connected to the main routing table of each address family supported by that routing protocol. See Section 4.4 for further explanation.

Routing tables are global, which means that a configured routing table may be used by any or all router instances.

Server implementations MAY pose restrictions regarding the number of supported routing tables, and rules for configuration and use of routing tables. For example:

- A server may support no more than one routing table per address family.

- Router instances (of a certain type) may not be allowed to share routing tables, i.e., each routing table is used by no more than one router instance.

For servers supporting multiple routing tables per address family, additional tables can be configured by creating new entries in the "routing-table" list, either as a part of factory-default configuration, or by a client's action.

The way how the routing system uses information from routing tables for actual packet forwarding is outside the scope of this document.

Every routing table can serve as a source of routes for other routing tables. To achieve this, one or more recipient routing tables may be specified in the configuration of the source routing table. Optionally, a route filter may be configured for any or all recipient routing tables. Such a route filter then selects and/or manipulates the routes that are passed on between the source and recipient routing table.

A routing table MUST NOT appear among its own recipient routing tables. A recipient routing table also MUST be of the same address family as its source routing table.Consequently, configuration of recipient routing tables makes sense only for servers supporting multiple routing tables per address family. Servers supporting only one routing table per address family MAY therefore decide to remove the container "recipient-routing-tables", together with its contents, from the data model.

4.4. Routing Protocols

The core routing data model provides an open-ended framework for defining multiple routing protocol instances within each router instance. Each routing protocol instance MUST be assigned a type, which is an identity derived from the "rt:routing-protocol" base identity. The core routing data model defines two identities for the direct and static pseudo-protocols (Section 4.4.1).

Each routing protocol instance is connected to exactly one routing table for each address family that the routing protocol instance supports. By default, every routing protocol instance SHOULD be connected to the main routing table or tables. An implementation MAY allow any or all routing protocol instances to be configured to use a different routing table.

Routes learned from the network by a routing protocol are passed to the connected routing table(s) and vice versa, subject to routing protocol specific rules and restrictions.

In addition, two independent route filters (see Section 4.5) may be defined for a routing protocol instance to control the exchange of routes in both directions between the routing protocol instance and the connected routing table:

o import filter controls which routes are passed from a routing protocol instance to the connected routing table,

o export filter controls which routes the routing protocol instance may receive from the connected routing table.

Note that, for historical reasons, the terms import and export are used from the viewpoint of a routing table.

4.4.1. Routing Pseudo-Protocols

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

Every router instance MUST implement exactly one instance of the "direct" pseudo-protocol type. The name of this instance MUST also be "direct". It is the source of direct routes for all configured address families. Direct routes are normally supplied by the operating system kernel, based on the configuration of network interface addresses, see Section 5.2. The "direct" pseudoprotocol MUST always be connected to the main routing tables of all supported address families. This means that direct routes are always installed in the main routing tables. However, direct routes MAY be filtered before they appear in the main routing table.

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

4.4.2. Defining New Routing Protocols

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

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

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

      /rt:routing-tables/rt:routing-table/rt:route,

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

- Per-interface configuration parameters can be added by augmenting the data node "rt:interface" (the list of router interfaces).

- Other configuration parameters and operational state data can be defined by augmenting the "routing-protocol" data node.

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

The above steps are implemented by the example YANG module for the RIP routing protocol in Appendix A. First, the module defines a new identity for the RIP protocol:

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

New route attributes specific to the RIP protocol ("metric" and "tag") are defined in a grouping and then added to the route definitions appearing in "routing-table" and in the output part of the "active-route" RPC method:

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

   augment "/rt:routing/rt:routing-tables/rt:routing-table/"
         + "rt:routes/rt:route" {
     description
       "RIP-specific route components.";
     uses route-content;
   }

   augment "/rt:active-route/rt:output/rt:route" {
     description
       "Add RIP-specific route content.";
     uses route-content;
   }
```

Per-interface configuration data are defined by the following "augment" statement:

```
   augment "/rt:routing/rt:router/rt:interfaces/rt:interface" {
     when "../../rt:routing-protocols/rt:routing-protocol/rt:type  = "
        + "'rip:rip'";
     container rip {
       description
         "Per-interface RIP configuration.";
       leaf enabled {
         type boolean;
         default "true";
       }
       leaf metric {
         type rip-metric;
         default "1";
       }
     }
   }
```

Finally, global RIP configuration data are integrated into the "rt:routing-protocol" node by using the following "augment" statement, which is again valid only for routing protocol instances whose type is "rip:rip":

```
augment "/rt:routing/rt:router/rt:routing-protocols/"
      + "rt:routing-protocol" {
  when "rt:type = 'rip:rip'";
  container rip {
    leaf update-interval {
      type uint8 {
        range "10..60";
      }
      units "seconds";
      default "30";
      description
        "Time interval between periodic updates.";
    }
  }
}
```

4.5. Route Filters

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

Route filters are global, which means that a configured route filter may be used by any or all router instances.

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

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

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

Note that the latter type is equivalent to no route filter.

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

Each route filter is identified by a name which MUST be unique within the entire configuration. Its type MUST be specified by the "type"

identity reference - this opens the space for multiple route filtering framework implementations. The default value for the route filter type is the identity "deny-all-route-filter".

4.6. RPC Operations

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

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

- o route-count: retrieve the total number of entries in a routing table.

5. Interactions with Other YANG Modules

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

In all cases, the relevant parts of the core routing data model are disabled but MUST NOT be deleted from the configuration by the server.

5.1. Module "ietf-interfaces"

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

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

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

5.2. Module "ietf-ip"

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

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

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

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

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

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

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

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

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

functions, such as routing protocols.

In addition, the "ietf-ip" module allows for configuring IPv4 and IPv6 addresses and subnet masks on network layer interfaces. Configuration of these parameters on an enabled interface MUST result in an immediate creation of the corresponding direct route (usually in the main routing table). Its destination prefix is set according to the configured IP address and subnet mask, and the interface is set as the outgoing interface for that route.

6. Routing YANG Module

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

```
<CODE BEGINS> file "ietf-routing@2012-10-04.yang"

module ietf-routing {

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

  prefix "rt";

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

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

  import ietf-interfaces {
    prefix "if";
  }

  import iana-afn-safi {
    prefix "ianaaf";
  }

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

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

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

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

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

  description
```

```
    "This YANG module defines essential components that may be used
     for configuring a routing subsystem.

     Copyright (c) 2012 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

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

  revision 2012-10-04 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for Routing Configuration";
  }

  /* Identities */

  identity router-type {
    description
      "Base identity from which router type identities are derived.

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

  identity standard-router {
    base router-type;
    description
      "This identity represents a standard router.";
  }

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

  identity direct {
```

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

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

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

     identity deny-all-route-filter {
       base route-filter;
       description
         "Route filter that blocks all routes.";
     }

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

     /* Type Definitions */

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

     /* Groupings */

     grouping afn-safi {
       leaf address-family {
         type ianaaf:address-family;
         mandatory "true";
         description
```

```
           "Address family of routes in the routing table.";
       }
       leaf safi {
         type ianaaf:subsequent-address-family;
         mandatory "true";
         description
           "Subsequent address family identifier of routes in the
            routing table.";
       }
       description
         "This grouping provides two parameters specifying address
          family and subsequent address family.";
     }

     grouping route-content {
       description
         "Generic parameters of routes.";
       leaf outgoing-interface {
         type if:interface-ref;
         description
           "Outgoing interface.";
       }
     }

     /* RPC Methods */

     rpc active-route {
       description
         "Return the active route (or multiple routes, in the case of
          multi-path routing) to a destination address.

          Parameters

          1. 'router-name',

          2. 'destination-address'.

          If the router instance with 'router-name' doesn't exist, then
          this operation shall fail with error-tag 'data-missing' and
          error-app-tag 'router-not-found'.

          If no active route for 'destination-address' exists, no output
          is returned - the server shall send an <rpc-reply> containing
          a single element <ok>.
         ";
       input {
         leaf router-name {
           type router-ref;
```

```
          mandatory "true";
          description
            "Name of the router instance whose forwarding information
             base is being queried.";
        }
        container destination-address {
          uses afn-safi;
          description
            "Network layer destination address.

             Address family specific modules must augment this
             container with a leaf named 'address'.
            ";
        }
      }
      output {
        list route {
          uses afn-safi;
          uses route-content;
          description
            "Route contents specific for each address family should be
             defined through augmenting.";
        }
      }
    }

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

         Parameters:

         1. 'routing-table-name'.

         If the routing table with the name specified in
         'routing-table-name' doesn't exist, then this operation shall
         fail with error-tag 'data-missing' and error-app-tag
         'routing-table-not-found'.
        ";
      input {
        leaf routing-table {
          type leafref {
            path "/routing/routing-tables/routing-table/name";
          }
          mandatory "true";
          description
            "Name of the routing table.";
        }
```

```
       }
       output {
         leaf number-of-routes {
           type uint32;
           mandatory "true";
           description
             "Number of routes in the routing table.";
         }
       }
     }

     /* Data Nodes */

     container routing {
       description
         "Routing parameters.";
       list router {
         key "name";
         unique "router-id";
         description
           "Each list entry is a container for configuration and
            operational state data of a single (logical) router.

            Network layer interfaces assigned to the router must have
            their entries in the 'interfaces' list.
           ";
         leaf name {
           type string;
           description
             "An arbitrary name of the router instance.";
         }
         leaf type {
           type identityref {
             base router-type;
           }
           default "rt:standard-router";
           description
             "This leaf specifies the router type.

              It is primarily intended as a means for discriminating
              among different types of logical routers, route
              virtualization, master-slave arrangements etc., while
              keeping all such router instances in the same flat list.

              Standard router instances should use the default value.
             ";
         }
         leaf enabled {
```

```
          type boolean;
          default "true";
          description
            "Enable/disable the router instance.

             If this parameter is false, the parent router instance is
             disabled, despite any other configuration that might be
             present.
            ";
        }
        leaf router-id {
          type inet:ipv4-address;
          description
            "Global router ID in the form of an IPv4 address.

             An implementation may select a value if this parameter is
             not configured.

             Routing protocols may override this global parameter
             inside their configuration.
            ";
        }
        leaf description {
          type string;
          description
            "Textual description of the router.";
        }
        container main-routing-tables {
          description
            "Main routing tables used by the router instance.";
          list main-routing-table {
            must "address-family=//routing/routing-tables/"
               + "routing-table[name=current()/name]/"
               + "address-family and safi=//routing/routing-tables/"
               + "routing-table[name=current()/name]/safi" {
              error-message "Address family mismatch.";
              description
                "The entry's address family must match that of the
                 referenced routing table.";
            }
            key "address-family safi";
            description
              "Each list entry specifies the main routing table for one
               address family.

               The main routing table receives direct routes, and all
               routing protocols should be connected to the main
               routing table(s) by default.
```

```
                Address families that don't have their entry in this
                list must not be used in the rest of the router instance
                configuration.
               ";
             uses afn-safi;
             leaf name {
               type leafref {
                 path "/routing/routing-tables/routing-table/name";
               }
               description
                 "Name of an existing routing table to be used as the
                  main routing table for the given router and address
                  family.";
             }
           }
         }
         container interfaces {
           description
             "Router interface parameters.";
           list interface {
             key "name";
             description
               "List of network layer interfaces assigned to the router
                instance.";
             leaf name {
               type if:interface-ref;
               description
                 "A reference to the name of a configured network layer
                  interface.";
             }
           }
         }
         container routing-protocols {
           description
             "Container for the list of configured routing protocol
              instances.";
           list routing-protocol {
             key "name";
             description
               "An instance of a routing protocol.";
             leaf name {
               type string;
               description
                 "An arbitrary name of the routing protocol instance.";
             }
             leaf description {
               type string;
               description
```

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

                 If this parameter is false, the parent routing
                 protocol instance is disabled, despite any other
                 configuration that might be present.
                ";
            }
            leaf type {
              type identityref {
                base routing-protocol;
              }
              mandatory "true";
              description
                "Type of the routing protocol - an identity derived
                 from the 'routing-protocol' base identity.";
            }
            container connected-routing-tables {
              description
                "Container for connected routing tables.";
              list connected-routing-table {
                must "not(//routing/routing-tables/"
                   + "routing-table[name=current()/"
                   + "preceding-sibling::connected-routing-table/"
                   + "name]/address-family=//routing/routing-tables/"
                   + "routing-table[name=current()/name]/"
                   + "address-family and //routing/routing-tables/"
                   + "routing-table[name=current()/"
                   + "preceding-sibling::connected-routing-table/"
                   + "name]/safi=//routing/routing-tables/"
                   + "routing-table[name=current()/name]/safi)" {
                  error-message "Duplicate address family for "
                              + "connected routing table.";
                  description
                    "For each AFN/SAFI pair there may be at most one
                     connected routing table.";
                }
                key "name";
                description
                  "List of routing tables to which the routing protocol
                   instance is connected.
```

```
                    If no connected routing table is defined for an
                    address family, the routing protocol should be
                    connected by default to the main routing table for
                    that address family.
                   ";
                 leaf name {
                   type leafref {
                     path "/routing/routing-tables/routing-table/name";
                   }
                   description
                     "Name of an existing routing table.";
                 }
                 leaf import-filter {
                   type leafref {
                     path "/routing/route-filters/route-filter/name";
                   }
                   description
                     "Reference to a route filter that is used for
                      filtering routes passed from this routing protocol
                      instance to the routing table specified by the
                      'name' sibling node.

                      If this leaf is not present, the behavior is
                      protocol-specific, but typically it means that all
                      routes are accepted.
                     ";
                 }
                 leaf export-filter {
                   type leafref {
                     path "/routing/route-filters/route-filter/name";
                   }
                   description
                     "Reference to a route filter that is used for
                      filtering routes passed from the routing table
                      specified by the 'name' sibling node to this
                      routing protocol instance.

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

                      The 'direct' and 'static' pseudo-protocols accept
                      no routes from any routing table.
                     ";
                 }
               }
             }
             container static-routes {
```

```
              when "../type='rt:static'" {
                description
                  "This container is only valid for the 'static'
                   routing protocol.";
              }
              description
                "Configuration of 'static' pseudo-protocol.

                 Address family specific modules should augment this
                 node with lists of routes.
                ";
            }
          }
        }
      }
      container routing-tables {
        description
          "Container for configured routing tables.";
        list routing-table {
          key "name";
          description
            "Each entry represents a routing table identified by the
             'name' key. All routes in a routing table must have the
             same AFN and SAFI.";
          leaf name {
            type string;
            description
              "An arbitrary name of the routing table.";
          }
          uses afn-safi;
          leaf description {
            type string;
            description
              "Textual description of the routing table.";
          }
          container routes {
            config "false";
            description
              "Current contents of the routing table (operational state
               data).";
            list route {
              description
                "A routing table entry. This data node must augmented
                 with information specific for routes of each address
                 family.";
              uses route-content;
              leaf source-protocol {
                type leafref {
```

```
              path "/routing/router/routing-protocols/"
                 + "routing-protocol/name";
            }
            mandatory "true";
            description
              "The name of an existing routing protocol instance
               from which the route comes.";
          }
          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 routing table.";
          }
        }
      }
      container recipient-routing-tables {
        description
          "Container for recipient routing tables.";
        list recipient-routing-table {
          must "name != ../../name" {
            error-message "Source and recipient routing tables "
                        + "are identical.";
            description
              "A routing table must not appear among its recipient
               routing tables.";
          }
          must "//routing/routing-tables/"
             + "routing-table[name=current()/name]/"
             + "address-family=../../address-family and //routing/"
             + "routing-tables/routing-table[name=current()/name]/"
             + "safi=../../safi" {
            error-message "Address family mismatch.";
            description
              "Address family of the recipient routing table must
               match the source table.";
          }
          key "name";
          description
            "List of routing tables that receive routes from this
             routing table.";
          leaf name {
            type leafref {
              path "/routing/routing-tables/routing-table/name";
            }
            description
              "The name of the recipient routing table.";
```

```
              }
              leaf filter {
                type leafref {
                  path "/routing/route-filters/route-filter/name";
                }
                description
                  "A route filter which is applied to the routes passed
                   on to the recipient routing table.";
              }
            }
          }
        }
      }
      container route-filters {
        description
          "Container for configured route filters.";
        list route-filter {
          key "name";
          description
            "Route filters are used for filtering and/or manipulating
             routes that are passed between a routing protocol and a
             routing table or vice versa, or between two routing
             tables.

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

```
     }
   }

   <CODE ENDS>
```

7. IPv4 Unicast Routing YANG Module

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

<CODE BEGINS> file "ietf-ipv4-unicast-routing@2012-10-04.yang"

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

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

  prefix "v4ur";

  import ietf-routing {
    prefix "rt";
  }

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

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

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

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

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

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

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

     Copyright (c) 2012 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with or
```

```
        without modification, is permitted pursuant to, and subject to
        the license terms contained in, the Simplified BSD License set
        forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

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

     revision 2012-10-04 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for Routing Configuration";
     }

     /* Groupings */

     grouping route-content {
       description
         "Parameters of IPv4 unicast routes.";
       leaf dest-prefix {
         type inet:ipv4-prefix;
         description
           "IPv4 destination prefix.";
       }
       leaf next-hop {
         type inet:ipv4-address;
         description
           "IPv4 address of the next hop.";
       }
     }

     /* RPC Methods */

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

```
          }
        }

        augment "/rt:active-route/rt:output/rt:route" {
          when "address-family='ipv4' and safi='nlri-unicast'" {
            description
              "This augment is valid only for IPv4 unicast.";
          }
          description
            "Contents of the reply to 'rt:active-route' operation.";
          uses route-content;
        }

        /* Data nodes */

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

                   It is not required that the routes be sorted by their
                   'id'.
                  ";
              }
              leaf description {
                type string;
                description
                  "Textual description of the route.";
              }
              uses rt:route-content;
              uses route-content {
                refine "dest-prefix" {
```

```
               mandatory "true";
             }
           }
         }
       }
     }

     augment "/rt:routing/rt:routing-tables/rt:routing-table/rt:routes/"
           + "rt:route" {
       when "../../rt:address-family='ipv4' and "
          + "../../rt:safi='nlri-unicast'" {
         description
           "This augment is valid only for IPv4 unicast.";
       }
       description
         "This augment defines the content of IPv4 unicast routes.";
       uses route-content;
     }
   }

   <CODE ENDS>
```

8. IPv6 Unicast Routing YANG Module

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

```
<CODE BEGINS> file "ietf-ipv6-unicast-routing@2012-10-04.yang"

module ietf-ipv6-unicast-routing {

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

  prefix "v6ur";

  import ietf-routing {
    prefix "rt";
  }

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix "ip";
  }

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

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

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

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

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

  description
```

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

        Copyright (c) 2012 IETF Trust and the persons identified as
        authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject to
        the license terms contained in, the Simplified BSD License set
        forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

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

     revision 2012-10-04 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for Routing Configuration";
     }

     /* Groupings */

     grouping route-content {
       description
         "Specific parameters of IPv6 unicast routes.";
       leaf dest-prefix {
         type inet:ipv6-prefix;
         description
           "IPv6 destination prefix.";
       }
       leaf next-hop {
         type inet:ipv6-address;
         description
           "IPv6 address of the next hop.";
       }
     }

     /* RPC Methods */

     augment "/rt:active-route/rt:input/rt:destination-address" {
       when "address-family='ipv6' and safi='nlri-unicast'" {
         description
           "This augment is valid only for IPv6 unicast.";
```

```
       }
       description
         "The 'address' leaf augments the 'rt:destination-address'
          parameter of the 'rt:active-route' operation.";
       leaf address {
         type inet:ipv6-address;
         description
           "IPv6 destination address.";
       }
     }

     augment "/rt:active-route/rt:output/rt:route" {
       when "address-family='ipv6' and safi='nlri-unicast'" {
         description
           "This augment is valid only for IPv6 unicast.";
       }
       description
         "Contents of the reply to 'rt:active-route' operation.";
       uses route-content;
     }

     /* Data nodes */

     augment "/rt:routing/rt:router/rt:interfaces/rt:interface" {
       when "/if:interfaces/if:interface[name=current()/name]/ip:ipv6/"
          + "ip:enabled='true'" {
         description
           "This augment is only valid for router interfaces with
            enabled IPv6.";
       }
       description
         "IPv6-specific parameters of router interfaces.";
       container ipv6-router-advertisements {
         description
           "Parameters of IPv6 Router Advertisements.";
         reference
           "RFC 4861: Neighbor Discovery for IP version 6 (IPv6).";
         leaf send-advertisements {
           type boolean;
           default "false";
           description
             "A flag indicating whether or not the router sends periodic
              Router Advertisements and responds to Router
              Solicitations.";
         }
         leaf max-rtr-adv-interval {
           type uint16 {
             range "4..1800";
```

```
          }
          units "seconds";
          default "600";
          description
            "The maximum time allowed between sending unsolicited
             multicast Router Advertisements from the interface.";
        }
        leaf min-rtr-adv-interval {
          type uint16 {
            range "3..1350";
          }
          must ". <= 0.75 * ../max-rtr-adv-interval" {
            description
              "The value must be no greater than
               3/4*max-rtr-adv-interval.";
          }
          units "seconds";
          description
            "The minimum time allowed between sending unsolicited
             multicast Router Advertisements from the interface.

             Must be no greater than 0.75 * max-rtr-adv-interval.

             Its default value is dynamic:

             - 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.
            ";
        }
        leaf managed-flag {
          type boolean;
          default "false";
          description
            "The boolean value to be placed in the 'Managed address
             configuration' flag field in the Router Advertisement.";
        }
        leaf other-config-flag {
          type boolean;
          default "false";
          description
            "The boolean value to be placed in the 'Other
             configuration' flag field in the Router Advertisement.";
        }
        leaf link-mtu {
          type uint32;
          default "0";
```

```
          description
            "The value to be placed in MTU options sent by the router.
             A value of zero indicates that no MTU options are sent.";
        }
        leaf reachable-time {
          type uint32 {
            range "0..3600000";
          }
          units "milliseconds";
          default "0";
          description
            "The value to be placed in the Reachable Time field in the
             Router Advertisement messages sent by the router. The
             value zero means unspecified (by this router).";
        }
        leaf retrans-timer {
          type uint32;
          units "milliseconds";
          default "0";
          description
            "The value to be placed in the Retrans Timer field in the
             Router Advertisement messages sent by the router. The
             value zero means unspecified (by this router).";
        }
        leaf cur-hop-limit {
          type uint8;
          default "64";
          description
            "The default value to be placed in the Cur Hop Limit field
             in the Router Advertisement messages sent by the router.
             The value should be set to the current diameter of the
             Internet. The value zero means unspecified (by this
             router).

             The default should be set to the value specified in IANA
             Assigned Numbers that was in effect at the time of
             implementation.
            ";
          reference
            "IANA: IP Parameters,
             http://www.iana.org/assignments/ip-parameters";
        }
        leaf default-lifetime {
          type uint16 {
            range "0..9000";
          }
          units "seconds";
          description
```

```
            "The value to be placed in the Router Lifetime field of
             Router Advertisements sent from the interface, in seconds.
             MUST be either zero or between max-rtr-adv-interval and
             9000 seconds. A value of zero indicates that the router is
             not to be used as a default router. These limits may be
             overridden by specific documents that describe how IPv6
             operates over different link layers.

             The default value is dynamic and should be set to 3 *
             max-rtr-adv-interval.
            ";
        }
        container prefix-list {
          description
            "A list of prefixes to be placed in Prefix Information
             options in Router Advertisement messages sent from the
             interface.

             By default, all prefixes that the router advertises via
             routing protocols as being on-link for the interface from
             which the advertisement is sent. The link-local prefix
             should not be included in the list of advertised prefixes.
            ";
          list prefix {
            key "prefix-spec";
            description
              "Advertised prefix entry.";
            leaf prefix-spec {
              type inet:ipv6-prefix;
              description
                "IPv6 address prefix.";
            }
            choice control-adv-prefixes {
              default "advertise";
              description
                "The prefix either may be explicitly removed from the
                 set of advertised prefixes, or parameters with which
                 it is advertised may be specified (default case).";
              leaf no-advertise {
                type empty;
                description
                  "The prefix will not be advertised.

                   This may 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, in seconds. The
                     designated value of all 1's (0xffffffff)
                     represents infinity.
                    ";
                }
                leaf on-link-flag {
                  type boolean;
                  default "true";
                  description
                    "The value to be placed in the on-link flag
                     ('L-bit') field in the Prefix Information
                     option.";
                }
                leaf preferred-lifetime {
                  type uint32;
                  units "seconds";
                  must ". <= ../valid-lifetime" {
                    description
                      "This value must not be larger than
                       valid-lifetime.";
                  }
                  default "604800";
                  description
                    "The value to be placed in the Preferred Lifetime
                     in the Prefix Information option, in seconds. The
                     designated value of all 1's (0xffffffff)
                     represents infinity.
                    ";
                }
                leaf autonomous-flag {
                  type boolean;
                  default "true";
                  description
                    "The value to be placed in the Autonomous Flag
                     field in the Prefix Information option.";
                }
              }
            }
          }
        }
      }
    }
```

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

                It is not required that the routes be sorted by their
                'id'.
               ";
           }
           leaf description {
             type string;
             description
               "Textual description of the route.";
           }
           uses rt:route-content;
           uses route-content {
             refine "dest-prefix" {
               mandatory "true";
             }
           }
         }
       }
     }

     augment "/rt:routing/rt:routing-tables/rt:routing-table/rt:routes/"
           + "rt:route" {
       when "../../rt:address-family='ipv6' and "
          + "../../rt:safi='nlri-unicast'" {
         description
           "This augment is valid only for IPv6 unicast.";
       }
       description
```

```
         "This augment defines the content of IPv6 unicast routes.";
       uses route-content;
     }
   }

   <CODE ENDS>
```

9. IANA Considerations

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

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

```
----------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-routing

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

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

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

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

10. Security Considerations

The YANG modules defined in this document are designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

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

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

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

/rt:routing/rt:router/rt:routing-protocols/rt:routing-protocol This list specifies the routing protocols configured on a device.

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

Unauthorized access to any of these lists can adversely affect the routing subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations and other problems.

11. Acknowledgments

The author wishes to thank Martin Bjorklund, Joel Halpern, Wes Hardaker, Andrew McGregor, Thomas Morin, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Dave Thaler and Yi Yang for their helpful comments and suggestions.

12. References

12.1. Normative References

[IANA-IF-AF]
Bjorklund, M., "IANA Interface Type and Address Family YANG Modules", draft-ietf-netmod-iana-if-type-04 (work in progress), June 2012.

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

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

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

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

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, September 2010.

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

[YANG-IF] Bjorklund, M., "A YANG Data Model for Interface Configuration", draft-ietf-netmod-interfaces-cfg-06 (work in progress), September 2012.

[YANG-IP] Bjorklund, M., "A YANG Data Model for IP Configuration", draft-ietf-netmod-ip-cfg-06 (work in progress), September 2012.

12.2. Informative References

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

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

Appendix A. Example: Adding a New Routing Protocol

This appendix demonstrates how the core routing data model can be extended to support a new routing protocol. The YANG module "example-rip" shown below is intended only as an illustration rather than a real definition of a data model for the RIP routing protocol. For the sake of brevity, we do not follow all the guidelines specified in [RFC6087]. See also Section 4.4.2.

```
<CODE BEGINS> file "example-rip@2012-10-04.yang"

module example-rip {

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

  prefix "rip";

  import ietf-routing {
    prefix "rt";
  }

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

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

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

```
     augment "/rt:routing/rt:routing-tables/rt:routing-table/rt:routes/"
           + "rt:route" {
       description
         "RIP-specific route components.";
       uses route-content;
     }

     augment "/rt:active-route/rt:output/rt:route" {
       description
         "Add RIP-specific route content.";
       uses route-content;
     }

     augment "/rt:routing/rt:router/rt:interfaces/rt:interface" {
       when "../../rt:routing-protocols/rt:routing-protocol/rt:type = "
          + "'rip:rip'";
       container rip {
         description
           "Per-interface RIP configuration.";
         leaf enabled {
           type boolean;
           default "true";
         }
         leaf metric {
           type rip-metric;
           default "1";
         }
       }
     }

     augment "/rt:routing/rt:router/rt:routing-protocols/"
           + "rt:routing-protocol" {
       when "rt:type = 'rip:rip'";
       container rip {
         leaf update-interval {
           type uint8 {
             range "10..60";
           }
           units "seconds";
           default "30";
           description
             "Time interval between periodic updates.";
         }
       }
     }
   }

   <CODE ENDS>
```

Appendix B. Example: NETCONF <get> Reply

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

- ietf-interfaces [YANG-IF],
- ietf-ip [YANG-IP],
- ietf-routing (Section 6),
- ietf-ipv4-unicast-routing (Section 7),
- ietf-ipv6-unicast-routing (Section 8).

We assume a simple network setup as shown in Figure 3: router "A" uses static default routes with the "ISP" router as the next hop. IPv6 router advertisements are configured only on the "eth1" interface and disabled on the upstream "eth0" interface.

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

Figure 3: Example network configuration

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

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

```
      message-id="101"
      xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
      xmlns:v4ur="urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing"
      xmlns:v6ur="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
      xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip"
      xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing">
   <data>
    <if:interfaces>
     <if:interface>
      <if:name>eth0</if:name>
      <if:type>ethernetCsmacd</if:type>
      <if:location>05:00.0</if:location>
      <ip:ipv4>
       <ip:address>
        <ip:ip>192.0.2.1</ip:ip>
        <ip:prefix-length>24</ip:prefix-length>
       </ip:address>
      </ip:ipv4>
      <ip:ipv6>
       <ip:address>
        <ip:ip>2001:0db8:0:1::1</ip:ip>
        <ip:prefix-length>64</ip:prefix-length>
       </ip:address>
       <ip:autoconf>
        <ip:create-global-addresses>false</ip:create-global-addresses>
       </ip:autoconf>
      </ip:ipv6>
     </if:interface>
     <if:interface>
      <if:name>eth1</if:name>
      <if:type>ethernetCsmacd</if:type>
      <if:location>05:00.1</if:location>
      <ip:ipv4>
       <ip:address>
        <ip:ip>198.51.100.1</ip:ip>
        <ip:prefix-length>24</ip:prefix-length>
       </ip:address>
      </ip:ipv4>
      <ip:ipv6>
       <ip:address>
        <ip:ip>2001:0db8:0:2::1</ip:ip>
        <ip:prefix-length>64</ip:prefix-length>
       </ip:address>
       <ip:autoconf>
        <ip:create-global-addresses>false</ip:create-global-addresses>
       </ip:autoconf>
      </ip:ipv6>
```

```
     </if:interface>
    </if:interfaces>
    <rt:routing>
     <rt:router>
      <rt:name>rtr0</rt:name>
      <rt:router-id>192.0.2.1</rt:router-id>
      <rt:description>Router A</rt:description>
      <rt:main-routing-tables>
       <rt:main-routing-table>
        <rt:address-family>ipv4</rt:address-family>
        <rt:safi>nlri-unicast</rt:safi>
        <rt:name>ipv4-unicast</rt:name>
       </rt:main-routing-table>
       <rt:main-routing-table>
        <rt:address-family>ipv6</rt:address-family>
        <rt:safi>nlri-unicast</rt:safi>
        <rt:name>ipv6-unicast</rt:name>
       </rt:main-routing-table>
      </rt:main-routing-tables>
      <rt:interfaces>
       <rt:interface>
        <rt:name>eth0</rt:name>
       </rt:interface>
       <rt:interface>
        <rt:name>eth1</rt:name>
        <v6ur:ipv6-router-advertisements>
         <v6ur:send-advertisements>true</v6ur:send-advertisements>
         <v6ur:prefix-list>
          <v6ur:prefix>
           <v6ur:prefix-spec>2001:db8:0:2::/64</v6ur:prefix-spec>
          </v6ur:prefix>
         </v6ur:prefix-list>
        </v6ur:ipv6-router-advertisements>
       </rt:interface>
      </rt:interfaces>
      <rt:routing-protocols>
       <rt:routing-protocol>
        <rt:name>direct</rt:name>
        <rt:type>rt:direct</rt:type>
       </rt:routing-protocol>
       <rt:routing-protocol>
        <rt:name>st0</rt:name>
        <rt:description>
         Static routing is used for the internal network.
        </rt:description>
        <rt:type>rt:static</rt:type>
        <rt:static-routes>
         <v4ur:ipv4>
```

```
          <v4ur:route>
           <v4ur:id>1</v4ur:id>
           <v4ur:dest-prefix>0.0.0.0/0</v4ur:dest-prefix>
           <v4ur:next-hop>192.0.2.2</v4ur:next-hop>
          </v4ur:route>
         </v4ur:ipv4>
         <v6ur:ipv6>
          <v6ur:route>
           <v6ur:id>1</v6ur:id>
           <v6ur:dest-prefix>::/0</v6ur:dest-prefix>
           <v6ur:next-hop>2001:db8:0:1::2</v6ur:next-hop>
          </v6ur:route>
         </v6ur:ipv6>
        </rt:static-routes>
        <rt:connected-routing-tables>
         <rt:connected-routing-table>
          <rt:name>ipv4-unicast</rt:name>
         </rt:connected-routing-table>
         <rt:connected-routing-table>
          <rt:name>ipv6-unicast</rt:name>
         </rt:connected-routing-table>
        </rt:connected-routing-tables>
       </rt:routing-protocol>
      </rt:routing-protocols>
     </rt:router>
     <rt:routing-tables>
      <rt:routing-table>
       <rt:name>ipv4-unicast</rt:name>
       <rt:address-family>ipv4</rt:address-family>
       <rt:safi>nlri-unicast</rt:safi>
       <rt:routes>
        <rt:route>
         <v4ur:dest-prefix>192.0.2.1/24</v4ur:dest-prefix>
         <rt:outgoing-interface>eth0</rt:outgoing-interface>
         <rt:source-protocol>direct</rt:source-protocol>
         <rt:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
        </rt:route>
        <rt:route>
         <v4ur:dest-prefix>198.51.100.0/24</v4ur:dest-prefix>
         <rt:outgoing-interface>eth1</rt:outgoing-interface>
         <rt:source-protocol>direct</rt:source-protocol>
         <rt:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
        </rt:route>
        <rt:route>
         <v4ur:dest-prefix>0.0.0.0/0</v4ur:dest-prefix>
         <rt:source-protocol>st0</rt:source-protocol>
         <v4ur:next-hop>192.0.2.2</v4ur:next-hop>
         <rt:last-updated>2012-10-02T18:02:45+01:00</rt:last-updated>
```

```
        </rt:route>
       </rt:routes>
      </rt:routing-table>
      <rt:routing-table>
       <rt:name>ipv6-unicast</rt:name>
       <rt:address-family>ipv6</rt:address-family>
       <rt:safi>nlri-unicast</rt:safi>
       <rt:routes>
        <rt:route>
         <v6ur:dest-prefix>2001:db8:0:1::/64</v6ur:dest-prefix>
         <rt:outgoing-interface>eth0</rt:outgoing-interface>
         <rt:source-protocol>direct</rt:source-protocol>
         <rt:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
        </rt:route>
        <rt:route>
         <v6ur:dest-prefix>2001:db8:0:2::/64</v6ur:dest-prefix>
         <rt:outgoing-interface>eth1</rt:outgoing-interface>
         <rt:source-protocol>direct</rt:source-protocol>
         <rt:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
        </rt:route>
        <rt:route>
         <v6ur:dest-prefix>::/0</v6ur:dest-prefix>
         <v6ur:next-hop>2001:db8:0:1::2</v6ur:next-hop>
         <rt:source-protocol>st0</rt:source-protocol>
         <rt:last-updated>2012-10-02T18:02:45+01:00</rt:last-updated>
        </rt:route>
       </rt:routes>
      </rt:routing-table>
     </rt:routing-tables>
    </rt:routing>
   </data>
  </rpc-reply>
```

Appendix C. Change Log

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

C.1. Changes Between Versions -04 and -05

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

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

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

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

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

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

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

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

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

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

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

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

C.2. Changes Between Versions -03 and -04

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

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

- Changed the key of the static route lists from "seqno" to "id" because the routes needn't be sorted.

- Added 'must' constraint saying that "preferred-lifetime" must not be greater than "valid-lifetime".

C.3. Changes Between Versions -02 and -03

- Module "iana-afn-safi" moved to I-D "iana-if-type".

- Removed forwarding table.

- RPC "get-route" changed to "active-route". Its output is a list of routes (for multi-path routing).

- New RPC "route-count".

- For both RPCs, specification of negative responses was added.

- Relaxed separation of router instances.

- Assignment of interfaces to router instances needn't be disjoint.

- Route filters are now global.

- Added "allow-all-route-filter" for symmetry.

- Added Section 5 about interactions with "ietf-interfaces" and "ietf-ip".

- Added "router-id" leaf.

- Specified the names for IPv4/IPv6 unicast main routing tables.

- Route parameter "last-modified" changed to "age".

- Added container "recipient-routing-tables".

C.4. Changes Between Versions -01 and -02

- Added module "ietf-ipv6-unicast-routing".

- The example in Appendix B now uses IP addresses from blocks reserved for documentation.

- Direct routes appear by default in the forwarding table.

- o Network layer interfaces must be assigned to a router instance. Additional interface configuration may be present.

- o The "when" statement is only used with "augment", "must" is used elsewhere.

- o Additional "must" statements were added.

- o The "route-content" grouping for IPv4 and IPv6 unicast now includes the material from the "ietf-routing" version via "uses rt:route-content".

- o Explanation of symbols in the tree representation of data model hierarchy.

C.5. Changes Between Versions -00 and -01

- o AFN/SAFI-independent stuff was moved to the "ietf-routing" module.

- o Typedefs for AFN and SAFI were placed in a separate "iana-afn-safi" module.

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

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

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

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

- o Section "Security Considerations" was written.

Author's Address

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz