NETMOD
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2012

L. Lhotka
CZ.NIC

February 20, 2012

A YANG Data Model for Routing Configuration
draft-ietf-netmod-routing-cfg-02

Abstract

This document contains a specification of four YANG modules. Together they form the core routing data model which serves as a basis for configuring a routing subsystem. It is therefore expected that this module 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

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 August 23, 2012.

Copyright Notice

Table of Contents

1. Introduction

This document contains a specification of four YANG modules:

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

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

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

- Module "iana-afn-safi" contains two type definitions translating IANA registries "Address Family Numbers" [IANA-AFN] and "Subsequent Address Family Identifiers" [IANA-SAFI] to YANG enumerations.

The first three modules together define the so-called core routing data model. This data model will serve 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, advanced functions such as route filtering or policy routing etc. 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
- 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 packet forwarding. If there are multiple candidate routes with a matching destination prefix, then it is up to the routing algorithm to select the active route.

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

direct route: a route to a directly connected network.

2.2. Prefixes in Data Node Names

In this document, names of data nodes 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 with their standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
eth	ex-ethernet	[YANG-IF]
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 contain list keys and question marks indicate optional data nodes. Nodes that represent configuration are labeled with "rw" while operational state data have the "ro" label.

```
+--rw routing
   +--rw router [name]
      +--rw name
      +--rw description?
      +--rw enabled?
      +--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 [seqno]
      |              +--rw v6ur:seqno
      |              +--rw v6ur:prefix-spec?
      |              +--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 type
      |     +--rw connected-routing-tables
      |     |  +--rw routing-table [name]
      |     |     +--rw name
      |     |     +--rw import-filter?
```

```
|     |     +--rw export-filter?
|     +--rw static-routes
|        +--rw v4ur:ipv4
|        |  +--rw v4ur:route [seqno]
|        |     +--rw v4ur:seqno
|        |     +--rw v4ur:description?
|        |     +--rw v4ur:outgoing-interface?
|        |     +--rw v4ur:dest-prefix?
|        |     +--rw v4ur:next-hop?
|        +--rw v6ur:ipv6
|           +--rw v6ur:route [seqno]
|              +--rw v6ur:seqno
|              +--rw v6ur:description?
|              +--rw v6ur:outgoing-interface?
|              +--rw v6ur:dest-prefix?
|              +--rw v6ur:next-hop?
+--rw route-filters
|  +--rw route-filter [name]
|     +--rw name
|     +--rw description?
|     +--rw type?
+--rw routing-tables
   +--rw routing-table [name]
      +--rw name
      +--rw address-family?
      +--rw safi?
      +--rw description?
      +--ro routes
      |  +--ro route
      |     +--ro source-protocol?
      |     +--ro last-modified?
      |     +--ro v4ur:outgoing-interface?
      |     +--ro v4ur:dest-prefix?
      |     +--ro v4ur:next-hop?
      |     +--ro v6ur:outgoing-interface?
      |     +--ro v6ur:dest-prefix?
      |     +--ro v6ur:next-hop?
      +--rw recipient-routing-tables [recipient-name]
         +--rw recipient-name
         +--rw filter?
```

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, route filters and RPC operations. 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.

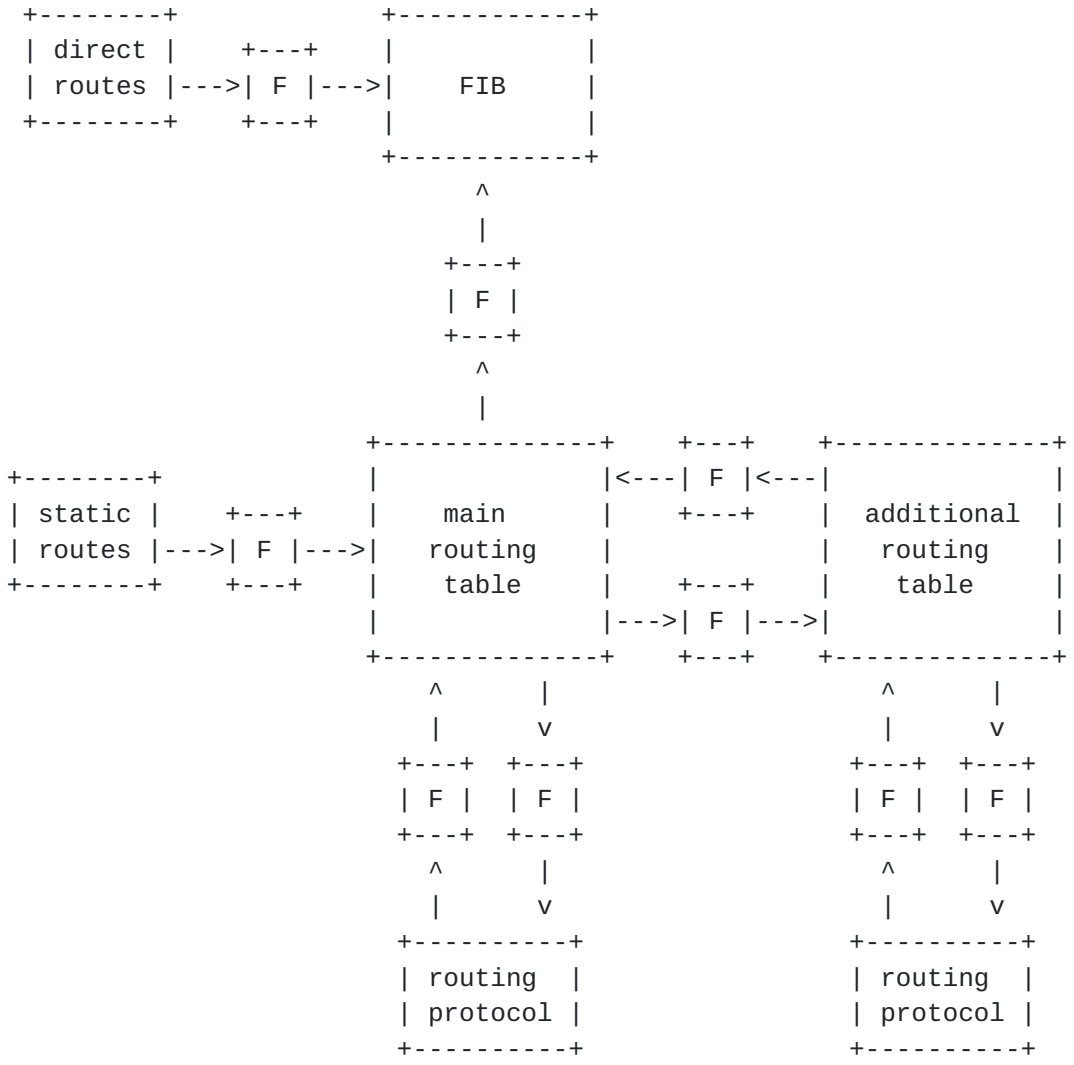

Figure 2: Example setup of the routing subsystem

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

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

- Each routing protocol instance, including the "static" and "direct" pseudo-protocols, is connected to exactly one routing

table with which it can exchange routes (in both directions, except for the "static" and "direct" pseudo-protocols).

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

- The forwarding information base (FIB) is a special routing table which must always be present. Typically, the FIB contains the "direct" routes for all configured interfaces and also receives the active routes from the main routing table. The operating system kernel uses this information for packet forwarding.

- 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 whose configuration and operation is independent of other router instances. Although it it not enforced by the data model, different router instances normally do not internally share any data. They may, however, communicate with each other via routing protocols.

Logical network interfaces must be assigned to a router instance in order to be able to participate in packet forwarding, routing protocols and other operations of that router instance. The assignment is accomplished by creating a corresponding entry in the list of router interfaces ("/router/interfaces/interface"). The key of the list entry MUST be the name of a configured logical interface. A logical interface MUST NOT be assigned to more than one router instance.

Apart from the key, each entry of the "/router/interfaces/interface" list MAY contain other configuration or operational state data related to the corresponding logical interface.

4.1.1. Configuration of IPv6 Router Interfaces

The module "ietf-ipv6-unicast-routing" augments the definition of the data node "/router/interfaces/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).

NOTE: The "IsRouter" flag, which is also required by [RFC4861], was omitted. Is is expected that this variable will be implemented in another module, either "ietf-interfaces" or "ietf-ip".

4.2. Route

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 the 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 is sufficient 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 in both 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-modified - date and time of last modification, or installation, of the route.

Each routing table may only contain routes of the same address family (AFN and SAFI).

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 such lists are controlled by routing protocol operations which may result in route additions, removals and modifications. This also includes manipulations via the "static" pseudo-protocol.

At least the following two routing tables MUST be configured for each router instance and each supported AFN/SAFI pair:

1. Forwarding information base (FIB) contains active routes that are used by the operating system kernel for forwarding datagrams.

2. Main routing table to which all routing protocol instances are connected by default, with the exception of the "direct" pseudo-protocol (Section 4.4): direct routes only appear in the FIB table by default.

The main routing table SHOULD serve as the default source of active routes for the FIB.

One or more additional routing tables MAY be configured by creating new entries in the "routing-table" list, either being a part of factory-default configuration or configured by the client.

The naming scheme for routing tables, as well as restrictions on the number and configurability of routing tables are implementation-specific.

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. In addition, a route filter may be configured for each recipient routing table, which selects and/or manipulates the routes that are passed on between the source and recipient routing table.

4.4. Routing Protocols

The core routing data model provides an open-ended framework for defining multiple routing protocol instances. Each of them is identified by a name, which MUST be unique within a router instance. Each protocol MUST be assigned a type, which MUST be 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.

Each routing protocol instance is connected to exactly one routing table. By default, every routing protocol instance SHOULD be connected to the main routing table. 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 and vice versa - routes appearing in a routing table are passed to all routing protocols connected to the table (except "direct" and "static" pseudo-protocols) and advertised by that protocol to the network.

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:

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

- 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.

The core routing data model defines two special routing protocols - "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 contain exactly one instance of the "direct" pseudo-protocol. It is the source of direct routes which are normally supplied by the operating system kernel, based on the detected and configured network interfaces, and they SHOULD by default appear in the FIB routing table. However, using the framework defined in this document, the target routing table for direct routes MAY be changed by connecting the "direct" protocol instance to a non-default routing table. Direct routes can also be filtered before they appear in the routing table.

The "static" routing pseudo-protocol allows for specifying routes manually. It MAY be configured in zero or multiple instances, although a typical implementation will have exactly one instance per router.

4.4.1. Defining New Routing Protocols

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

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

- Additional route attributes MAY be defined. Their definitions then have to be inserted as operational state data by augmenting the definition of "rt:route" inside "rt:routing-table", 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 can be defined by augmenting the "routing-protocol" data node. By using the "when" statement, this

augment SHOULD be made conditional and valid only if the value of the "rt:type" child leaf equals to the new protocol's identity.

It is recommended that both per-interface and other configuration data specific to the new protocol be encapsulated in an appropriately named container.

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 "get-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:router/rt:routing-tables/rt:routing-table/"
         + "rt:routes/rt:route" {
     when "../../../../rt:routing-protocols/"
        + "rt:routing-protocol[rt:name=current()/rt:source-protocol]/"
        + "rt:type='rip:rip'" {
       description
         "This augment is only valid if the source protocol from which
          the route originated is RIP.";
     }
     description
       "RIP-specific route components.";
     uses route-content;
   }

   augment "/rt:get-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 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 properties.

By itself, the route filtering framework defined in this document

allows to establish only the two extreme routing policies in which either all routes are allowed or all routes are rejected. It is expected that real route filtering frameworks will be developed separately.

Each route filter is identified by a name which MUST be unique within a router instance. Its type MUST be specified by the "type" identity reference - this opens the space for multiple route filtering framework implementations. The default value for route filter type is the identity "deny-all-route-filter" defined in the "ietf-routing" module, which represents a route filtering policy in which all routes are rejected.

4.6. RPC Operation

The "ietf-routing" module defines the "get-route" RPC operation. It is used for querying the forwarding information base of a router instance. The first input parameter is the name of the router instance whose FIB is to be queried, and the second parameter is a destination address. Modules for particular address families are expected to augment the "destination-address" container with the "address" leaf, as it is done in the "ietf-ipv4-unicast-routing" and "ietf-ipv6-unicast-routing" modules.

The server replies with an active route which is used for forwarding datagrams to the destination address within the selected router instance. Again, modules for particular address families are expected to augment the definition of output parameters with AFN/ SAFI-specific contents.

5. IANA AFN and SAFI 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 "iana-afn-safi@2012-02-20.yang"

```
module iana-afn-safi {

  namespace "urn:ietf:params:xml:ns:yang:iana-afn-safi";

  prefix "ianaaf";

  organization
    "IANA";

  contact
    "Internet Assigned Numbers Authority

     Postal:
     ICANN
     4676 Admiralty Way, Suite 330
     Marina del Rey, CA 90292
     U. S. A.


     Tel: +1 310 823 9358
     E-Mail: iana&iana.org
    ";

  description
    "This YANG module provides two typedefs containing YANG
     definitions for the following IANA-registered enumerations:

     - Address Family Numbers (AFN)

     - Subsequent Address Family Identifiers (SAFI)

     The latest revision of this YANG module can be obtained from the
     IANA web site.

     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 2012-02-20 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for Routing Configuration";
    }

    typedef address-family {
      type enumeration {
        enum other {
          value "0";
          description
            "none of the following";
        }
        enum ipV4 {
          value "1";
          description
            "IP Version 4";
        }
        enum ipV6 {
          value "2";
          description
            "IP Version 6";
        }
        enum nsap {
          value "3";
          description
            "NSAP";
        }
        enum hdlc {
          value "4";
          description
            "(8-bit multidrop)";
        }
        enum bbn1822 {
          value "5";
          description
            "BBN Report 1822";
        }
        enum all802 {
```

```
          value "6";
          description
            "(includes all 802 media plus Ethernet 'canonical
             format')";
        }
        enum e163 {
          value "7";
          description
            "E.163";
        }
        enum e164 {
          value "8";
          description
            "(SMDS, FrameRelay, ATM)";
        }
        enum f69 {
          value "9";
          description
            "(Telex)";
        }
        enum x121 {
          value "10";
          description
            "(X.25, Frame Relay)";
        }
        enum ipx {
          value "11";
          description
            "IPX (Internet Protocol Exchange)";
        }
        enum appleTalk {
          value "12";
          description
            "Apple Talk";
        }
        enum decnetIV {
          value "13";
          description
            "DEC Net Phase IV";
        }
        enum banyanVines {
          value "14";
          description
            "Banyan Vines";
        }
        enum e164withNsap {
          value "15";
          description
```

```
            "(E.164 with NSAP format subaddress)";
          }
          enum dns {
            value "16";
            description
              "(Domain Name System)";
          }
          enum distinguishedName {
            value "17";
            description
              "(Distinguished Name, per X.500)";
          }
          enum asNumber {
            value "18";
            description
              "(16-bit quantity, per the AS number space)";
          }
          enum xtpOverIPv4 {
            value "19";
            description
              "XTP over IP version 4";
          }
          enum xtpOverIpv6 {
            value "20";
            description
              "XTP over IP version 6";
          }
          enum xtpNativeModeXTP {
            value "21";
            description
              "XTP native mode XTP";
          }
          enum fibreChannelWWPN {
            value "22";
            description
              "Fibre Channel World-Wide Port Name";
          }
          enum fibreChannelWWNN {
            value "23";
            description
              "Fibre Channel World-Wide Node Name";
          }
          enum gwid {
            value "24";
            description
              "Gateway Identifier";
          }
          enum afi {
```

```
          value "25";
          description
            "AFI for L2VPN";
        }
      }
      description
        "This typedef is a YANG enumeration of IANA-registered address
         family numbers (AFN).";
      reference
        "Address Family Numbers. IANA, 2011-01-20.
         <http://www.iana.org/assignments/address-family-numbers/
         address-family-numbers.xml>

         IANA-ADDRESS-FAMILY-NUMBERS-MIB DEFINITIONS
         <http://www.iana.org/assignments/ianaaddressfamilynumbers-mib>
        ";
    }

    typedef subsequent-address-family {
      type enumeration {
        enum nlri-unicast {
          value "1";
          description
            "Network Layer Reachability Information used for unicast
             forwarding";
          reference
            "RFC4760";
        }
        enum nlri-multicast {
          value "2";
          description
            "Network Layer Reachability Information used for multicast
             forwarding";
          reference
            "RFC4760";
        }
        enum nlri-mpls {
          value "4";
          description
            "Network Layer Reachability Information (NLRI) with MPLS
             Labels";
          reference
            "RFC3107";
        }
        enum mcast-vpn {
          value "5";
          description
            "MCAST-VPN";
```

```
          reference
            "draft-ietf-l3vpn-2547bis-mcast-bgp-08";
        }
        enum nlri-dynamic-ms-pw {
          value "6";
          status "obsolete";
          description
            "Network Layer Reachability Information used for Dynamic
             Placement of Multi-Segment Pseudowires (TEMPORARY -
             Expires 2008-08-23)";
          reference
            "draft-ietf-pwe3-dynamic-ms-pw-13";
        }
        enum tunnel-safi {
          value "64";
          description
            "Tunnel SAFI";
          reference
            "draft-nalawade-kapoor-tunnel-safi-05";
        }
        enum vpls {
          value "65";
          description
            "Virtual Private LAN Service (VPLS)";
          reference
            "RFC4761, RFC6074";
        }
        enum bgp-mdt {
          value "66";
          description
            "BGP MDT SAFI";
          reference
            "RFC6037";
        }
        enum bgp-4over6 {
          value "67";
          description
            "BGP 4over6 SAFI";
          reference
            "RFC5747";
        }
        enum bgp-6over4 {
          value "68";
          description
            "BGP 6over4 SAFI";
          reference
            "mailto:cuiyong&tsinghua.edu.cn";
        }
```

```
          enum l1vpn-auto-discovery {
            value "69";
            description
              "Layer-1 VPN auto-discovery information";
            reference
              "draft-ietf-l1vpn-bgp-auto-discovery-05";
          }
          enum mpls-vpn {
            value "128";
            description
              "MPLS-labeled VPN address";
            reference
              "RFC4364";
          }
          enum multicast-bgp-mpls-vpn {
            value "129";
            description
              "Multicast for BGP/MPLS IP Virtual Private Networks
               (VPNs)";
            reference
              "draft-ietf-l3vpn-2547bis-mcast-10,
               draft-ietf-l3vpn-2547bis-mcast-10";
          }
          enum route-target-constraints {
            value "132";
            description
              "Route Target constraints";
            reference
              "RFC4684";
          }
          enum ipv4-diss-flow {
            value "133";
            description
              "IPv4 dissemination of flow specification rules";
            reference
              "RFC5575";
          }
          enum vpnv4-diss-flow {
            value "134";
            description
              "IPv4 dissemination of flow specification rules";
            reference
              "RFC5575";
          }
          enum vpn-auto-discovery {
            value "140";
            description
              "VPN auto-discovery";
```

```
          reference
            "draft-ietf-l3vpn-bgpvpn-auto-09";
        }
      }
      description
        "This typedef is a YANG enumeration of IANA-registered
         subsequent address family identifiers (SAFI).";
      reference
        "Subsequent Address Family Identifiers (SAFI) Parameters. IANA,
         2011-03-04. <http://www.iana.org/assignments/safi-namespace/
         safi-namespace.xml>
        ";
    }
  }

  <CODE ENDS>
```

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@2012-02-20.yang"

module ietf-routing {

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

  prefix "rt";

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

  import ietf-interfaces {
    prefix "if";
  }

  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 module contains YANG definitions of 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 2012-02-20 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for Routing Configuration";
    }

    /* Identities */

    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.";
    }

    /* 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;
        default "ipV4";
        description
          "Address family of routes in the routing table.";
      }
      leaf safi {
        type ianaaf:subsequent-address-family;
        default "nlri-unicast";
        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.

         A module for an address family should define a specific
         version of this grouping containing 'uses rt:route-content'.
        ";
      leaf outgoing-interface {
        type if:interface-ref;
        description
          "Outgoing interface.";
      }
```

```
    }

    /* RPC Methods */

    rpc get-route {
      description
        "Query the forwarding information base of a router instance
         whose name is given as the first parameter 'router-name'. The
         second parameter 'destination-address' should be augmented in
         order to support destination addresses of all supported
         address families. The server returns the route which is
         currently used for forwarding datagrams to that destination
         address, or an error message, if no such route exists.";
      input {
        leaf router-name {
          type router-ref;
          mandatory "true";
          description
            "First parameter: name of the router instance whose
             forwarding information base is queried.";
        }
        container destination-address {
          uses afn-safi;
          description
            "Second parameter: destination address.

             AFN/SAFI-specific modules must augment this container with
             a leaf named 'address'.
            ";
        }
      }
      output {
        container route {
          uses afn-safi;
          description
            "Contents of the reply specific for each address family
             should be defined through augmenting.";
        }
      }
    }

    /* Data Nodes */

    container routing {
      description
        "Routing parameters.";
      list router {
        key "name";
```

```
        unique "interfaces/interface/name";
        description
          "Each list entry is a container for configuration and
           operational state data of a single (logical) router.";
        leaf name {
          type string;
          description
            "The unique router name.";
        }
        leaf description {
          type string;
          description
            "Textual description of the router.";
        }
        leaf enabled {
          type boolean;
          default "true";
          description
            "Enable or disable the router. The default value is 'true',
             which means that the router is enabled.";
        }
        container interfaces {
          description
            "Router interface parameters.";
          list interface {
            key "name";
            description
              "List of logical interfaces assigned to the router
               instance. Any logical interface can only be assigned to
               one router instance.";
            leaf name {
              type if:interface-ref;
              description
                "A reference to the name of a configured logical
                 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
          "The name of the routing protocol instance.";
      }
      leaf description {
        type string;
        description
          "Textual description of the routing protocol
           instance.";
      }
      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 routing-table {
          must "not(../../../../routing-tables/"
             + "routing-table[current()/"
             + "preceding-sibling::routing-table/name]/"
             + "address-family=../../../../routing-tables/"
             + "routing-table[current()/name]/"
             + "address-family and ../../../../routing-tables/"
             + "routing-table[current()/"
             + "preceding-sibling::routing-table/name]/safi=../"
             + "../../../routing-tables/routing-table[current()/"
             + "name]/safi)" {
            error-message
              "Each routing protocol may have no more than one
               connected routing table for each AFN and SAFI.";
            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.

             Implementation may provide default routing tables
             for some AFN/SAFI pairs, which are used if the
             corresponding entry is not configured.
            ";
```

```
            leaf name {
              type leafref {
                path "../../../../../routing-tables/routing-table/"
                   + "name";
              }
              description
                "Reference to an existing routing table.";
            }
            leaf import-filter {
              type leafref {
                path "../../../../../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 "../../../../../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,
                 except for the 'direct' and 'static'
                 pseudo-protocols which accept no routes from any
                 routing table.";
            }
          }
        }
        container static-routes {
          must "../type='static'" {
            error-message
              "Static routes may be configured only for 'static'
               routing protocol.";
            description
              "This container is only valid for the 'static'
               routing protocol.";
          }
```

```
            description
              "Configuration of 'static' pseudo-protocol.";
          }
        }
      }
      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
              "The name of the route filter.";
          }
          leaf description {
            type string;
            description
              "Textual description of the route filter.";
          }
          leaf type {
            type identityref {
              base route-filter;
            }
            default "deny-all-route-filter";
            description
              "Type of the route-filter - an identity derived from
               the 'route-filter' base identity. The default value
               represents an all-blocking filter.";
          }
        }
      }
      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
                "The 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.";
                leaf source-protocol {
                  type leafref {
                    path "../../../../../routing-protocols/"
                       + "routing-protocol/name";
                  }
                  description
                    "The name of the routing protocol instance from
                     which the route comes. This routing protocol must
                     be configured (automatically or manually) in the
                     device.";
                }
                leaf last-modified {
                  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 to the routing
                     table.";
                }
              }
            }
            list recipient-routing-tables {
              key "recipient-name";
              description
                "A list of routing tables that receive routes from this
                 routing table.";
              leaf recipient-name {
```

```
                type leafref {
                  path "../../../routing-table/name";
                }
                description
                  "The name of the recipient routing table.";
              }
              leaf filter {
                type leafref {
                  path "../../../../route-filters/route-filter/name";
                }
                description
                  "A route filter which is applied to the routes passed
                   on to the recipient routing table.";
              }
            }
          }
        }
      }
    }
  }

  <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@2012-02-20.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 module augments the 'ietf-routing' module with YANG
     definitions for basic configuration of 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 2012-02-20 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: A YANG Data Model for Routing Configuration";
   }

   /* Groupings */

   grouping route-content {
     description
       "Parameters of IPv4 unicast routes.";
     uses rt:route-content;
     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:get-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:get-route' operation.";
     leaf address {
       type inet:ipv4-address;
       description
         "IPv4 destination address.";
```

```
      }
    }

    augment "/rt:get-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:get-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 "seqno";
          ordered-by "user";
          description
            "A user-ordered list of static routes.";
          leaf seqno {
            type uint16;
            description
              "Sequential number of the route.";
          }
          leaf description {
            type string;
            description
              "Textual description of the route.";
          }
          uses route-content;
        }
      }
    }

    augment "/rt:routing/rt:router/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@2012-02-20.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 module augments the 'ietf-routing' module with YANG
        definitions for basic configuration of 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 2012-02-20 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for Routing Configuration";
     }

     /* Groupings */

     grouping route-content {
       description
         "Specific parameters of IPv6 unicast routes.";
       uses rt:route-content;
       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:get-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:get-route' operation.";
      leaf address {
        type inet:ipv6-address;
        description
          "IPv6 destination address.";
      }
    }

    augment "/rt:get-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:get-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.

           NOTE: Parameter 'is-router' is not included, it is expected
           that it will be implemented by the 'ietf-ip' module.
          ";
      }
      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).

           RFC 4862: IPv6 Stateless Address Autoconfiguration.
          ";
        leaf send-advertisements {
          type boolean;
          default "false";
          description
```

```
            "A flag indicating whether or not the router sends periodic
             Router Advertisements and responds to Router
             Solicitations.";
          }
          leaf max-rtr-adv-interval {
            type uint16 {
              range "4..1800";
            }
            units "seconds";
            default "600";
            description
              "The maximum time allowed between sending unsolicited
               multicast Router Advertisements from the interface.";
          }
          leaf min-rtr-adv-interval {
            type uint16 {
              range "3..1350";
            }
            units "seconds";
            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 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 MaxRtrAdvInterval 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.

             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 "seqno";
            unique "prefix-spec";
            description
              "Advertised prefix entry.";
            leaf seqno {
              type uint8;
              description
                "Sequential number of the entry.";
            }
            leaf prefix-spec {
              type inet:ipv6-prefix;
              description
                "IPv6 address prefix.";
            }
            leaf valid-lifetime {
              type uint32;
              units "seconds";
              default "2592000";
              description
                "The value to be placed in the Valid Lifetime in the
                 Prefix Information option, in seconds. The designated
                 value of all 1's (0xffffffff) represents infinity.
```

```
                Implementations may allow valid-lifetime to be
                specified in two ways:

                1. a time that decrements in real time, that is, one
                   that will result in a Lifetime of zero at the
                   specified time in the future,

                2. a fixed time that stays the same in consecutive
                   advertisements.
              ";
            }
            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";
              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.

                 Implementations MAY allow AdvPreferredLifetime to be
                 specified in two ways:

                 1. a time that decrements in real time, that is, one
                    that will result in a Lifetime of zero at a
                    specified time in the future,

                 2. a fixed time that stays the same in consecutive
                    advertisements.
              ";
            }
            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 "seqno";
          ordered-by "user";
          description
            "A user-ordered list of static routes.";
          leaf seqno {
            type uint16;
            description
              "Sequential number of the route.";
          }
          leaf description {
            type string;
            description
              "Textual description of the route.";
          }
          uses route-content;
        }
      }
    }

    augment "/rt:routing/rt:router/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.

URI: urn:ietf:params:xml:ns:yang:iana-afn-safi

Registrant Contact: IANA.

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
   ------------------------------------------------------------------

   ------------------------------------------------------------------
   name:         iana-afn-safi
   namespace:    urn:ietf:params:xml:ns:yang:iana-afn-safi
   prefix:       ianaaf
   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 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 logical 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:router/rt:route-filters/rt:route-filter This list specifies the configured route filters which represent the 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, Tom Petch and Juergen Schoenwaelder for their helpful comments and suggestions.

12. References

12.1. Normative References

[IANA-AFN] IANA, "Address Family Numbers.", January 2011.

[IANA-SAFI] IANA, "Subsequent Address Family Identifiers (SAFI) Parameters.", March 2011.

[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-03 (work in progress), February 2012.

[YANG-IP] Bjorklund, M., "A YANG Data Model for IP Configuration", draft-ietf-netmod-ip-cfg-02 (work in progress), February 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 4.4.1.

<CODE BEGINS> file "example-rip@2012-02-20.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:router/rt:routing-tables/rt:routing-table/"
        + "rt:routes/rt:route" {
    when "../../../../rt:routing-protocols/"
       + "rt:routing-protocol[rt:name=current()/rt:source-protocol]/"
       + "rt:type='rip:rip'" {
      description
        "This augment is only valid if the source protocol from which
         the route originated is RIP.";
    }
    description
      "RIP-specific route components.";
    uses route-content;
  }

  augment "/rt:get-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: Reply to the NETCONF <get> Message

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],
- ex-ethernet [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.

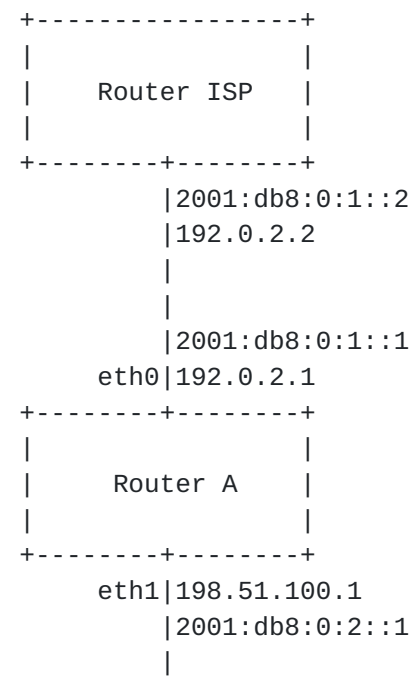

Figure 3: Example network configuration

Router "A" then could send the following XML document as its reply to the NETCONF <get> message:

```
  <?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:eth="http://example.com/ethernet"
      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: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:seqno>1</v6ur:seqno>
           <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:seqno>1</v4ur:seqno>
           <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:seqno>1</v6ur:seqno>
           <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:routing-table>
          <rt:name>ipv4-unicast-main</rt:name>
         </rt:routing-table>
         <rt:routing-table>
          <rt:name>ipv6-unicast-main</rt:name>
         </rt:routing-table>
        </rt:connected-routing-tables>
       </rt:routing-protocol>
      </rt:routing-protocols>
      <rt:routing-tables>
       <rt:routing-table>
        <rt:name>ipv4-unicast-fib</rt:name>
        <rt:routes>
         <rt:route>
          <v4ur:dest-prefix>192.0.2.1/24</v4ur:dest-prefix>
          <v4ur:outgoing-interface>eth0</v4ur:outgoing-interface>
          <rt:source-protocol>direct</rt:source-protocol>
          <rt:last-modified>2012-02-20T17:11:27+01:00</rt:last-modified>
         </rt:route>
         <rt:route>
          <v4ur:dest-prefix>198.51.100.0/24</v4ur:dest-prefix>
          <v4ur:outgoing-interface>eth1</v4ur:outgoing-interface>
          <rt:source-protocol>direct</rt:source-protocol>
          <rt:last-modified>2012-02-20T17:11:27+01:00</rt:last-modified>
         </rt:route>
         <rt:route>
          <v4ur:dest-prefix>0.0.0.0/0</v4ur:dest-prefix>
          <v4ur:next-hop>192.0.2.2</v4ur:next-hop>
          <rt:source-protocol>st0</rt:source-protocol>
          <rt:last-modified>2012-02-20T18:02:45+01:00</rt:last-modified>
         </rt:route>
        </rt:routes>
       </rt:routing-table>
       <rt:routing-table>
        <rt:name>ipv6-unicast-fib</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>
          <v6ur:outgoing-interface>eth0</v6ur:outgoing-interface>
          <rt:source-protocol>direct</rt:source-protocol>
          <rt:last-modified>2012-02-20T17:11:27+01:00</rt:last-modified>
```

```
        </rt:route>
        <rt:route>
         <v6ur:dest-prefix>2001:db8:0:2::/64</v6ur:dest-prefix>
         <v6ur:outgoing-interface>eth1</v6ur:outgoing-interface>
         <rt:source-protocol>direct</rt:source-protocol>
         <rt:last-modified>2012-02-20T17:11:27+01:00</rt:last-modified>
        </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-modified>2012-02-20T18:02:45+01:00</rt:last-modified>
        </rt:route>
       </rt:routes>
      </rt:routing-table>
      <rt:routing-table>
       <rt:name>ipv4-unicast-main</rt:name>
       <rt:recipient-routing-tables>
        <rt:recipient-name>ipv4-unicast-fib</rt:recipient-name>
       </rt:recipient-routing-tables>
       <rt:routes>
        <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-modified>2012-02-20T18:02:45+01:00</rt:last-modified>
        </rt:route>
       </rt:routes>
      </rt:routing-table>
      <rt:routing-table>
       <rt:name>ipv6-unicast-main</rt:name>
       <rt:address-family>ipV6</rt:address-family>
       <rt:safi>nlri-unicast</rt:safi>
       <rt:recipient-routing-tables>
        <rt:recipient-name>ipv6-unicast-fib</rt:recipient-name>
       </rt:recipient-routing-tables>
       <rt:routes>
        <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-modified>2012-02-20T18:02:45+01:00</rt:last-modified>
        </rt:route>
       </rt:routes>
      </rt:routing-table>
     </rt:routing-tables>
    </rt:router>
   </rt:routing>
```

```
   </data>
  </rpc-reply>
```

Appendix C. Change Log

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

C.1. 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 FIB table.
- Logical interfaces must be assigned to a router instance. Additional interface configuration may be present.
- The "when" statement is only used with "augment", "must" is used elsewhere.
- Additional "must" statements were added.
- The "route-content" grouping for IPv4 and IPv6 unicast now includes the material from the "ietf-routing" version via "uses rt:route-content".
- Explanation of symbols in the tree representation of data model hierarchy.

C.2. Changes Between Versions -00 and -01

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

Author's Address

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz