NETMOD	L. Lhotka
Internet-Draft	CESNET
Intended status: Standards Track	April 27, 2011
Expires: October 29, 2011	

A YANG Data Model for Routing Configuration
draft-ietf-netmod-routing-cfg-00

Abstract

This document contains a specification of two YANG modules that together provide a data model for essential configuration of a routing subsystem. It is expected that this module will serve as a basis for further development of data models for individual routing protocols and other related functions. The present data model defines the building blocks for such configurations - routing processes, routes and routing tables, routing protocol instances and route filters.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document contains an initial specification of two YANG modules, "ietf-routing" and "ietf-ipv4-unicast-routing", that 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 two modules can be directly used for simple IPv4-only devices with static routing, their main purpose is to provide basic building blocks for more complicated setups involving other address families such as IPv6, multiple routing protocols, and advanced functions, for example route filtering and policy routing. To this end, it is expected that this module 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 [RFC4741]:

- *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 the same destination prefix, then it is up to the routing algorithm to select the active route.

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 the standard prefixes associated with YANG modules, as shown in Table 1.

Prefix	YANG module	Reference
eth	ex-ethernet	[YANG-IF]
if	ietf-interfaces	[YANG-IF]
inet	ietf-inet-types	[RFC6021]
ip	ex-ip	[YANG-IF]
rip	example-rip	Appendix Appendix A
rt	ietf-routing	Section 5
v4ur	ietf-ipv4-unicast-routing	Section 6
yang	ietf-yang-types	[RFC6021]

Prefixes and corresponding YANG modules

3. Objectives

The initial design of the core routing data model was driven by the following main 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 two YANG modules. The first module, "ietf-routing", is rather minimal and provides only a top-level container ("routing") and a list of routing processes. Each routing process represents an instance of a (virtual) router with a separate forwarding table (FIB, forwarding information base). For a given address family, specified by an Address Family Identifier (AFI) [IANA-AFI] and Subsequent Address Family Identifier (SAFI) [IANA-SAFI], several independent routing processes may be configured.

The second YANG module, "ietf-ipv4-unicast-routing", provides a data modeling framework for IPv4 unicast routing with several essential components: routes, routing tables, routing protocol instances, route filters and RPC operations. The following subsections provide further details about these components.

By combining the components in various ways, and possibly filling them with appropriate contents defined in other modules, a broad range of routing setups can be covered.

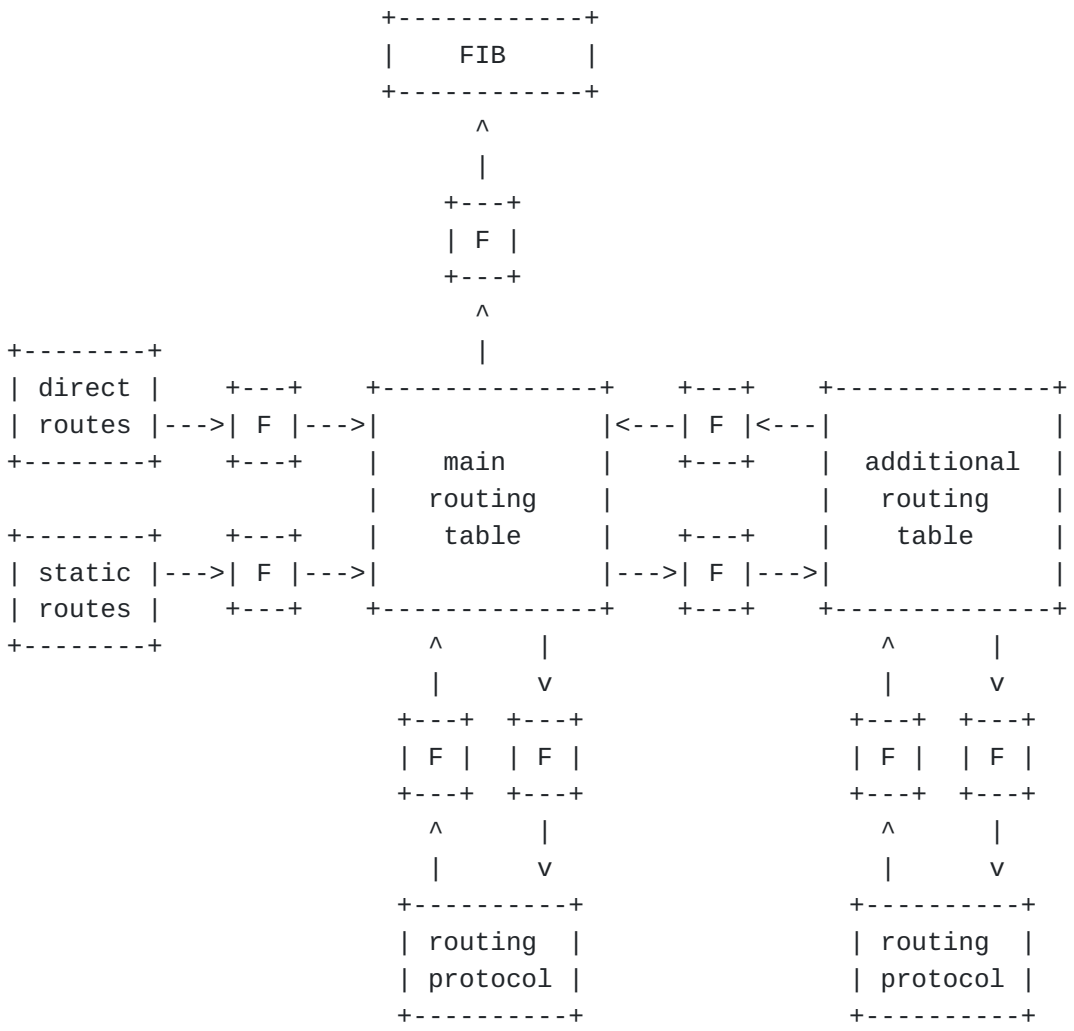


Figure 1 shows an example of a more complicated setup:

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

*Each routing protocol instance, including the "static" and "direct" pseudo-protocol instances, 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 one or both directions.

*The forwarding information base (FIB) is a special routing table which must always be present. Typically, the FIB receives the active routes from the main routing table and 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 the figure.

4.1. Route

Routes are basic units of information in a routing system. The "ietf-ipv4-unicast-routing" module 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, as well as in operational state data, for example as entries in routing tables.

4.2. 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.

In the core routing data model, the list of routes in routing tables is represented as operational state data. Routing protocol operations result in route additions, removals and modifications. This also includes manipulations via the "static" pseudo-protocol.

The "ietf-ipv4-unicast-routing" module requires that at least the following two routing tables MUST be configured for each routing process:

- The "ipv4-unicast-fib" table is the forwarding information base used by the operating system kernel for forwarding IPv4 unicast datagrams.

- The "ipv4-unicast-main" table is the main routing table. By default, all IPv4 unicast routing protocols exchange routes with this table, and active routes from the "ipv4-unicast-main" routing table are installed in the "ipv4-unicast-fib" table and used for packet forwarding.

Additional routing tables MAY be configured.
Every routing table MAY 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.

4.3. Routing Protocol Instances

The "ietf-ipv4-unicast-routing" module provides an open-ended framework for defining multiple routing protocol instances. Each of them is identified by a name, which is unique within a routing process, and MUST be assigned a type from a selection which includes all routing protocol types supported by the server, such as RIP, OSPF or BGP.
Each routing protocol instance is connected to exactly one routing table. By default, every routing protocol instance is connected to the main routing table, but any routing protocol instance can be configured to use a different routing table, provided such an extra table is configured.
Routes learned from the network by a routing protocol instance are passed to the connected routing table and vice versa - routes appearing in a routing table are passed to all routing protocol connected to the table and advertised by that protocol to the network.
Two independent route filters (see Section 4.4) 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 "ietf-ipv4-unicast-routing" module 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 routing process MUST contain exactly one instance of the "direct" pseudo-protocol. It is the source of routes to directly connected networks (so-called direct routes). Such routes are supplied by the operating system kernel based on the detected and configured network interfaces, and they usually appear in the main routing table. However, using the framework defined in this document, the target routing table for direct routes can be changed by connecting the "direct" protocol instance to a non-default routing table, and the direct routes can also be filtered before they appear in the routing table.
The "static" routing pseudo-protocol allows for specifying routes manually. It can be configured in zero or more instances, although typically one instance suffices.

4.3.1. 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 to the core routing framework in the following way:

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

* Additional route attributes MAY be defined. Their definitions have to be inserted as operational state data by augmenting the definition of "v4ur:route" inside "v4ur:routing-table". Naturally, route attributes (including the extra attributes) may be used in configuration data, too, as demonstrated by the "static" pseudo-protocol.

* The recommended way of defining configuration data specific to the new protocol is to augment the "routing-protocol-instance" list entry with a container that encapsulates the configuration hierarchy of the new protocol. The "augment" statement SHOULD be made conditional by using a "when" substatement requiring that the new nodes be used only if the "type" leaf node is equal to the new protocol's identity.

The above steps are implemented by the example YANG module for the RIP routing protocol in Appendix 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.";
}
```

Second, new route attributes specific for the RIP protocol ("metric" and "tag") are added:

```
augment "/rt:routing/rt:routing-process/v4ur:ipv4-unicast-routing/"
      + "v4ur:routing-tables/v4ur:routing-table/"
      + "v4ur:routes/v4ur:route" {
    when "../../../../v4ur:routing-protocol-instances/"
       + "v4ur:routing-protocol-instance[rt:name="
       + "current()/v4ur:source-protocol]/v4ur:type='rip:rip'";
  description
    "RIP-specific route components.";
  leaf metric { ... }
  leaf tag { ... }
}
```

The "when" statement is used to make sure that the new route attributes are only valid when the source protocol is RIP.
Finally, RIP-specific configuration data are integrated into the "v4ur:routing-protocol-instance" node by using the following "augment" statement, which applies only to routing protocol instances whose type is "rip:rip", and which is a part of a routing process whose address family is "ipV4" and subsequent address family identifier is "nlri-unicast":

```
augment "/rt:routing/rt:routing-process/v4ur:ipv4-unicast-routing/"
      + "v4ur:routing-protocol-instances/"
      + "v4ur:routing-protocol-instance" {
  when "v4ur:type = 'rip:rip' and ../../../rt:address-family = 'ipV4'"
     + " and ../../../safi = 'nlri-unicast'";
  container rip-configuration {
    ...
  }
}
```

4.4. Route Filters

The "ietf-ipv4-unicast-routing" module 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 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 the "ietf-ipv4-unicast-routing" module allows to establish only the two extreme routing policies in which either all routes are allowed or all routes are denied. It is expected that a real route filtering framework (or several alternative frameworks) will be developed separately.
Each route filter is identified by a name which is unique within a routing process. 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 the "deny all" route filtering policy.

4.5. RPC Operations

The "ietf-ipv4-unicast-routing-module" defines two RPC operations:

* "delete-route" operations allows the client to immediately delete specific route(s) from a routing table within a routing process. The first input parameter of this operation is the name of the routing process, the second parameter is the routing table to act upon, and the third (optional) parameter is the "route" container with zero or more of the following route attributes: "destination-prefix", "next-hop" and "outgoing-interface". All routes that match these attributes MUST be deleted from the selected routing table. If the "route" container is missing or empty, all routes from the selected routing table MUST be deleted.

* "get-route" is used for querying the forwarding information base of a routing process. The first input parameter is the name of a routing process whose FIB is to be queried, and the second parameter is an IPv4 destination address. The server replies with an active route which is used for forwarding datagrams to the destination address within the selected routing process.

5. Routing YANG Module

```
<CODE BEGINS> file "ietf-routing@2011-04-27.yang"

module ietf-routing {
  namespace "urn:ietf:params:xml:ns:yang:ietf-routing";
  prefix rt;

  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@cesnet.cz>";
  description
    "This module contains YANG definitions for top-level containers
     for the configuration of routing together with several type
     definitions and identities.";

  revision 2011-04-27 {
    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
      "Identity for the pseudo-protocol providing routes to directly
       connected networks. An implementation MUST preconfigure
       exactly one instance of this pseudo-protocol for each routing
       process."; }

  identity static {
    base routing-protocol;
```

```
  description
    "Identity for 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
    "This identity represents a route filter that blocks all
     routes.";
}

/* Type definitions */

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;
}
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 families.";
  reference
    "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 families.";
    reference "http://www.iana.org/assignments/safi-namespace/"
            + "safi-namespace.xml";
  }

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

  /* Data nodes */

  container routing {
    description
      "Routing parameters.";
    list routing-process {
```

```
      key "name";
      description
        "Each entry is a container for configuration and operational
         state data of a single (virtual) router for a given address
         family and subsequent address family identifier (SAFI). Each
         entry has a unique name.

         The definitions of data for a particular address family and
         subsequent address family shall be provided via augmentation
         by other modules.";
      leaf name {
        type string;
        description
          "The unique name of the routing process.";
      }
      leaf address-family {
        type address-family;
        default "ipV4";
        description
          "Address family of the routing process.";
      }
      leaf safi {
        type subsequent-address-family;
        default "nlri-unicast";
        description
          "Subsequent address family identifier of the routing
          process.";
      }
      leaf description {
        type string;
        description
          "Textual description of the routing process.";
      }
      leaf enabled {
        type boolean;
        default "true";
        description
          "Enable or disable the routing process. The default value
           is 'true', which means that the process is enabled.";
      }
    }
  }
}

<CODE ENDS>
```

6. IPv4 Unicast Routing YANG Module

```
<CODE BEGINS> file "ietf-ipv4-unicast-routing@2011-04-27.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-yang-types {
    prefix yang;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-interfaces {
    prefix if;
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

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

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

     Editor:   Ladislav Lhotka
     <mailto:lhotka@cesnet.cz>";
  description
    "This module augments the 'ietf-routing' module with YANG
     definitions for basic configuration of IPv4 unicast routing.

     It is immediately usable for a device that needs just a single
     routing table populated with static routes.

     On the other hand, the framework is designed to handle
     arbitrarily complex configurations with any number of routing
     tables and various routing protocols (in multiple instances).";

  revision 2011-04-27 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for Routing Configuration";
  }
```

```
/* Groupings */

grouping routing-process-name {
  leaf routing-process-name {
    type rt:routing-process-ref;
    must "/rt:routing/rt:routing-process[rt:name = current()]"
       + "/rt:address-family = 'ipV4' and "
       + "/rt:routing/rt:routing-process[rt:name = current()]"
       + "/rt:safi = 'nlri-unicast'" {
      description
        "The referred routing process must be IPv4 unicast.";
    }
  description "The name of a routing process.";
  }
  description
    "This grouping defines the first common parameter of both
     RPC operations below.";
}

/* RPC operations */

rpc get-route {
  description
    "Query the forwarding information base of an IPv4 unicast
     routing process whose name is given as the first
     parameter. The second parameter is an IPv4 destination
     address. 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 {
    uses routing-process-name;
    leaf destination-address {
      type inet:ipv4-address;
      description
        "Second parameter - IPv4 destination address.";
    }
  }
  output {
    container route {
      description
        "Contents of the reply.";
      leaf destination-prefix {
        type inet:ipv4-prefix;
        mandatory true;
        description
          "Destination prefix of the returned route.";
      }
      leaf next-hop {
```

```
        type inet:ipv4-address;
        description
          "Next hop address of the returned route.";
      }
      leaf outgoing-interface {
        type if:interface-ref;
        description
          "Outgoing interface of the returned route.";
      }
    }
  }
}

rpc delete-route {
  description

    "Delete all routes that match the given attributes from a
     routing table within a routing process.

     Parameters:
     1. routing process name,
     2. routing table name,
     3. Container 'route' with route attributes.

     <ok> is returned by the server upon successful completion.";

  input {
    uses routing-process-name;
    leaf routing-table {
      type leafref {
        path "/rt:routing/rt:routing-process[rt:name=current()/../"
           + "routing-process-name]/ipv4-unicast-routing/"
           + "routing-tables/routing-table/name";
      }
      mandatory true;
      description
        "First parameter.";
    }
    container route {
      description
        "Second parameter. All routes matching the route
         attributes must be deleted from the routing table.

         If this container is empty or missing, all routes
         from the selected routing table are deleted.";
      leaf destination-prefix {
        type inet:ipv4-prefix;
        description
          "Match destination prefix.";
```

```
      }
      leaf next-hop {
        type inet:ipv4-address;
        description
          "Match next hop.";
      }
      leaf outgoing-interface {
        type if:interface-ref;
        description
          "Match outgoing interface.";
      }
    }
  }
}


/* Data nodes */

augment "/rt:routing/rt:routing-process" {
  when "afi='ipV4' and safi='nlri-unicast'" {
    description
      "IPv4 unicast.";
  }
  description
    "Definitions of data nodes that augment a routing process
     for IPv4 unicast.";
  container ipv4-unicast-routing {
    description
      "Container for IPv4 unicast routing configuration and
       operational state data.";
    container routing-protocol-instances {
      description
        "Container for the list of configured routing protocol
         instances.";
      list routing-protocol-instance {
        key "name";
        description
          "An instance of a routing protocol.";
        container static-routes {
          when "../type='rt:static'" {
            description
              "These data nodes are only valid for the static
               pseudo-protocol.";
          }
          description
            "Configuration of a 'static' pseudo-protocol
             instance consists of a list of routes.";
          list static-route {
            key "id";
```

```
    ordered-by user;
    description
      "An user-ordered list of static routes.";
    leaf id {
      type string;
      description
        "An identification string for the route.";
    }
    leaf description {
      type string;
      description
        "Textual description of the route.";
    }
    leaf destination-prefix {
      type inet:ipv4-prefix;
      mandatory true;
      description
        "The destination prefix for which the route may
         be used.";
    }
    leaf next-hop {
      type inet:ipv4-address;
      description
        "IPv4 address of the host or router to which
         packets whose address matches 'destination-prefix'
         are to be forwarded.";
    }
    leaf outgoing-interface {
      type if:interface-ref;
      description
        "Name of the outgoing interface. This attribute
         is mainly used in direct routes.";
    }
  }
}
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 rt:routing-protocol;
```

```
      }
      mandatory true;
      description
        "Type of the routing protocol - an identity derived
         from the 'rt:routing-protocol' base identity.";
    }
    leaf routing-table {
      type leafref {
        path "../../../routing-tables/routing-table/name";
      }
      default "ipv4-unicast-main";
      description
        "The routing table to which the routing protocol
         instance is connected. By default it is the
         'ipv4-unicast-main' 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
         'routing-table' 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
         'routing-table' sibling 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 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 rt:route-filter;
      }
      default "rt:deny-all-route-filter";
      description
        "Type of the route-filter - an identity derived
         from the 'rt:route-filter' base identity. The default
         value represents an all-blocking filter.";
    }
  }
}
container routing-tables {
  must "routing-table/name='ipv4-unicast-fib'" {
    description
      "IPv4 unicast forwarding information base.";
  }
  must "routing-table/name='ipv4-unicast-main'" {
    description
      "The main IPv4 unicast routing table.";
  }
  description
    "Container for configured routing tables.";
  list routing-table {
    key "name";
    description
      "Each entry represents a configured routing table. At
       least two entries with names 'ipv4-unicast-fib' and
       'ipv4-unicast-main' must exist.";
    container routes {
      config false;
      description
```

```
    "Current contents of the routing table. Note that
     it is operational state data.";
  list route {
    description
      "A routing table entry.";
    leaf destination-prefix {
      type inet:ipv4-prefix;
      description
        "Destination prefix.";
    }
    leaf next-hop {
      type inet:ipv4-address;
      description
        "IPv4 address of the next hop.";
    }
    leaf outgoing-interface {
      type if:interface-ref;
      description
        "Name of the outgoing interface.";
    }
    leaf source-protocol {
      type leafref {
        path "../../../../../routing-protocol-instances/"
           + "routing-protocol-instance/name";
      }
      description
        "Protocol instance from which the route comes.";
    }
    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.";
    }
  }
}
leaf name {
  type string;
  description
    "The name of the routing table.";
}
leaf description {
  type string;
  description
    "Textual description of the routing table.";
}
list recipient-routing-tables {
```

```
            key "recipient-name";
            description
              "A list of routing tables that receive routes from
               the parent 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. IANA Considerations

This document registers the following two 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.
---------------------------------------------------------
```

This document registers two 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
--------------------------------------------------------------------
```

8. Security Considerations

TBD.

9. Acknowledgments

The author wishes to thank Juergen Schoenwaelder and Martin Bjorklund for their helpful comments and suggestions.

10. References

10.1. Normative References

[IANA-AFI]	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.
[RFC4741]	Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.
[RFC6020]	Bjorklund, M, "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, September 2010.
[RFC6021]	Schoenwaelder, J., "Common YANG Data Types", RFC 6021, September 2010.
[YANG-IF]	Bjorklund, M, "A YANG Data Model for Interface Configuration", Internet-Draft draft-bjorklund-netmod-interfaces-cfg-00, December 2010.

10.2. Informative References

[RFC6087]	Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 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. Appendix Appendix A.1 contains a YANG module which is used for this purpose. It is intended only as an illustration and not as a real definition of a data model for the RIP routing protocol. Also, for the sake of brevity, we do not follow all the guidelines specified in [RFC6087].
Appendix Appendix A.2 then contains a complete instance XML document - a reply to the NETCONF <get> message from a server that uses the RIP protocol as well as static routing.

Appendix A.1. Example YANG Module for Routing Information Protocol

```
module example-rip {
  namespace "http://example.com/rip";
  prefix rip;

  import ietf-interfaces {
    prefix if;
  }
  import ietf-routing {
    prefix rt;
  }

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

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

  augment "/rt:routing/rt:routing-protocol-instances/" +
          "rt:routing-protocol-instance" {
    when "rt:type='rip:rip'";
    container rip-configuration {
      container rip-interfaces {
        list rip-interface {
          key "name";
          leaf name {
            type if:interface-ref;
          }
          leaf enabled {
            type boolean;
            default "true";
          }
          leaf metric {
            type rip-metric;
            default "1";
          }
          /* Additional per-interface RIP configuration */
        }
      }
      leaf update-interval {
        type uint8 {
          range "10..60";
        }
        units "seconds";
```

```
        default "30";
        description
          "Time interval between periodic updates.";
      }
      /* Additional RIP configuration */
    }
  }
  augment "/rt:routing/rt:routing-tables/rt:routing-table/rt:route" {
      when "../../../rt:routing-protocol-instances/" +
           "rt:routing-protocol-instance[rt:name=" +
           "current()/rt:source-protocol]/rt:type='rip:rip'";
    description
      "RIP-specific route components.";
    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.";
    }
  }
}
```

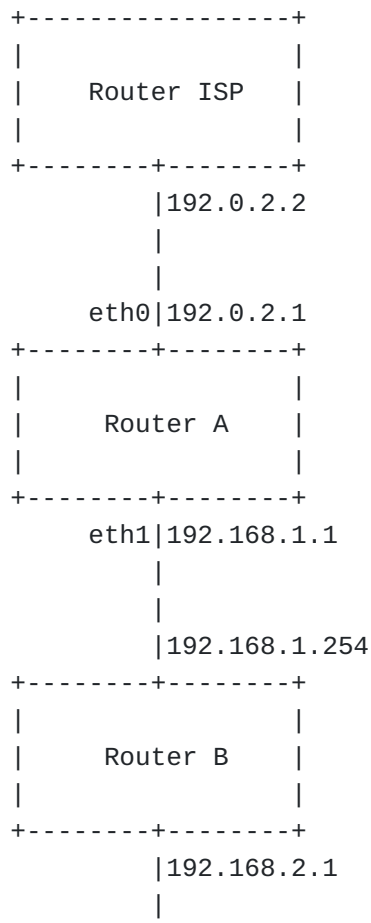
Appendix A.2. Sample Reply to the NETCONF <get> Message

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

- *ietf-interfaces [YANG-IF],
- *ex-ethernet [YANG-IF],
- *ex-ip [YANG-IF],
- *ietf-routing (Section 5),
- *ietf-ipv4-unicast-routing (Section 6),
- *example-rip (Appendix Appendix A.1).

We assume a simple network setup as shown in Figure 10: routers "ISP" and "A" use RIP for exchanging routing information whereas static routing is used in the private network. In order to avoid the redistribution of the routes to the private subnetworks 192.168.1.0/24 and 192.168.2.0/24 in RIP, an export filter is used in the RIP protocol

configuration preventing the routes from the main routing table from appearing in RIP updates.



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

```
<?xml version="1.0"?>

<nc:rpc-reply
    message-id="101"
    xmlns="urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:eth="http://example.com/ethernet"
    xmlns:ip="http://example.com/ip"
    xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing"
    xmlns:rip="http://example.com/rip">
  <nc:data>
    <if:interfaces>
      <if:interface>
        <if:name>eth0</if:name>
        <if:type>ethernetCsmacd</if:type>
        <if:location>05:00.0</if:location>
        <ip:ip>
          <ip:address>
            <ip:ip>192.0.2.1</ip:ip>
            <ip:prefix-length>24</ip:prefix-length>
          </ip:address>
        </ip:ip>
      </if:interface>
      <if:interface>
        <if:name>eth1</if:name>
        <if:type>ethernetCsmacd</if:type>
        <if:location>05:00.1</if:location>
        <ip:ip>
          <ip:address>
            <ip:ip>192.168.1.1</ip:ip>
            <ip:prefix-length>24</ip:prefix-length>
          </ip:address>
        </ip:ip>
      </if:interface>
    </if:interfaces>
    <rt:routing>
      <rt:routing-process>
        <rt:name>inet-0</rt:name>
        <rt:address-family>ipV4</rt:address-family>
        <rt:safi>nlri-unicast</rt:safi>
        <ipv4-unicast-routing>
          <routing-protocol-instances>
            <routing-protocol-instance>
              <name>direct</name>
              <type>rt:direct</type>
            </routing-protocol-instance>
            <routing-protocol-instance>
              <name>st0</name>
```

```
    <description>
      Static routing is used for the internal network.
    </description>
    <type>rt:static</type>
    <static-routes>
     <static-route>
      <id>id-6378</id>
      <destination-prefix>192.168.2.0/24</destination-prefix>
      <next-hop>192.168.1.254</next-hop>
     </static-route>
    </static-routes>
  </routing-protocol-instance>
  <routing-protocol-instance>
    <name>rip0</name>
    <description>
      RIP is used on the uplink.
      Static routes to the internal networks are not
      advertized in RIP.
    </description>
    <type>rip:rip</type>
    <export-filter>deny-all</export-filter>
    <rip:rip-configuration>
      <rip:rip-interfaces>
        <rip:rip-interface>
          <rip:name>eth0</rip:name>
        </rip:rip-interface>
      </rip:rip-interfaces>
    </rip:rip-configuration>
  </routing-protocol-instance>
</routing-protocol-instances>
<route-filters>
  <route-filter>
    <name>deny-all</name>
  </route-filter>
</route-filters>
<routing-tables>
  <routing-table>
    <name>ipv4-unicast-fib</name>
    <routes>
     <route>
      <destination-prefix>192.0.2.1/24</destination-prefix>
      <source-protocol>direct</source-protocol>
      <outgoing-interface>eth0</outgoing-interface>
      <last-modified>2010-04-01T17:11:27+01:00</last-modified>
     </route>
     <route>
      <destination-prefix>192.168.1.0/24</destination-prefix>
      <source-protocol>direct</source-protocol>
      <outgoing-interface>eth1</outgoing-interface>
```

```
    <last-modified>2010-04-01T17:11:27+01:00</last-modified>
   </route>
   <route>
    <destination-prefix>192.168.2.0/24</destination-prefix>
    <source-protocol>st0</source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <last-modified>2010-04-01T17:11:32+01:00</last-modified>
   </route>
   <route>
    <destination-prefix>0.0.0.0/0</destination-prefix>
    <source-protocol>rip0</source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <rip:metric>2</rip:metric>
    <rip:tag>64500</rip:tag>
    <last-modified>2010-04-01T18:02:45+01:00</last-modified>
   </route>
  </routes>
</routing-table>
<routing-table>
  <name>ipv4-unicast-main</name>
  <recipient-routing-tables>
    <recipient-name>ipv4-unicast-fib</recipient-name>
  </recipient-routing-tables>
  <routes>
   <route>
    <destination-prefix>192.0.2.1/24</destination-prefix>
    <source-protocol>direct</source-protocol>
    <outgoing-interface>eth0</outgoing-interface>
    <last-modified>2010-04-01T17:11:27+01:00</last-modified>
   </route>
   <route>
    <destination-prefix>192.168.1.0/24</destination-prefix>
    <source-protocol>direct</source-protocol>
    <outgoing-interface>eth1</outgoing-interface>
    <last-modified>2010-04-01T17:11:27+01:00</last-modified>
   </route>
   <route>
    <destination-prefix>192.168.2.0/24</destination-prefix>
    <source-protocol>st0</source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <last-modified>2010-04-01T17:11:32+01:00</last-modified>
   </route>
   <route>
    <destination-prefix>0.0.0.0/0</destination-prefix>
    <source-protocol>rip0</source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <rip:metric>2</rip:metric>
    <rip:tag>64500</rip:tag>
    <last-modified>2010-04-01T18:02:45+01:00</last-modified>
```

```
              </route>
             </routes>
           </routing-table>
         </routing-tables>
       </ipv4-unicast-routing>
     </rt:routing-process>
   </rt:routing>
  </nc:data>
</nc:rpc-reply>
```

Author's Address

Ladislav Lhotka Lhotka CESNET EMail: lhotka@cesnet.cz