NETMOD	L. Lhotka
Internet-Draft	CESNET
Intended status: Standards Track	September 23, 2011
Expires: March 26, 2012	

A YANG Data Model for Routing Configuration
draft-ietf-netmod-routing-cfg-01

Abstract

This document contains a specification of three 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 common building blocks for such configurations - router instances, routes, routing tables, routing protocols and route filters.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This document contains an initial specification of three 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 "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.

ED. QUESTION: Would it be possible/useful to publish the "iana-afn-safi" module as a separate I-D, perhaps together with "iana-if-type"?
The first two 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 two modules can be directly used for simple IPv4-only devices with static routing, their main purpose is to provide essential building blocks for more complicated setups involving other address families such as IPv6, multicast routing, multiple routing protocols, and advanced functions such as route filtering or 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 [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" and "ietf-ipv4-unicast-routing-cfg" modules.

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]
inet	ietf-inet-types	[RFC6021]
ip	ietf-ip	[YANG-IP]
rip	example-rip	Appendix Appendix A
rt	ietf-routing	Section 6

Prefix	YANG module	Reference
v4ur	ietf-ipv4-unicast-routing	Section 7
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 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", defines the generic components of a routing system. The second module, "ietf-ipv4-unicast-routing", augments the "ietf-routing" module with new data nodes that are needed for IPv4 unicast routing.
The combined data hierarchy defined by both YANG modules is shown in Figure 1.

```
+--rw routing
   +--rw router [name]
      +--rw name
      +--rw description?
      +--rw enabled?
      +--rw routing-protocols
      |  +--rw routing-protocol [name]
      |     +--rw name
      |     +--rw description?
      |     +--rw type
      |     +--rw connected-routing-tables
      |     |  +--rw connected-routing-table [name]
      |     |     +--rw name
      |     |     +--rw import-filter?
      |     |     +--rw export-filter?
      |     +--rw v4ur:ipv4-unicast-static-routes
      |        +--rw v4ur:static-route [id]
      |           +--rw v4ur:id
      |           +--rw v4ur:description?
      |           +--rw v4ur:destination-prefix?
      |           +--rw v4ur:next-hop?
      |           +--rw v4ur:outgoing-interface?
      +--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:destination-prefix?
            |     +--ro v4ur:next-hop?
            |     +--ro v4ur:outgoing-interface?
            +--rw recipient-routing-tables [recipient-name]
               +--rw recipient-name
               +--rw filter?
```

As can be see 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 provide further details about these components.

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

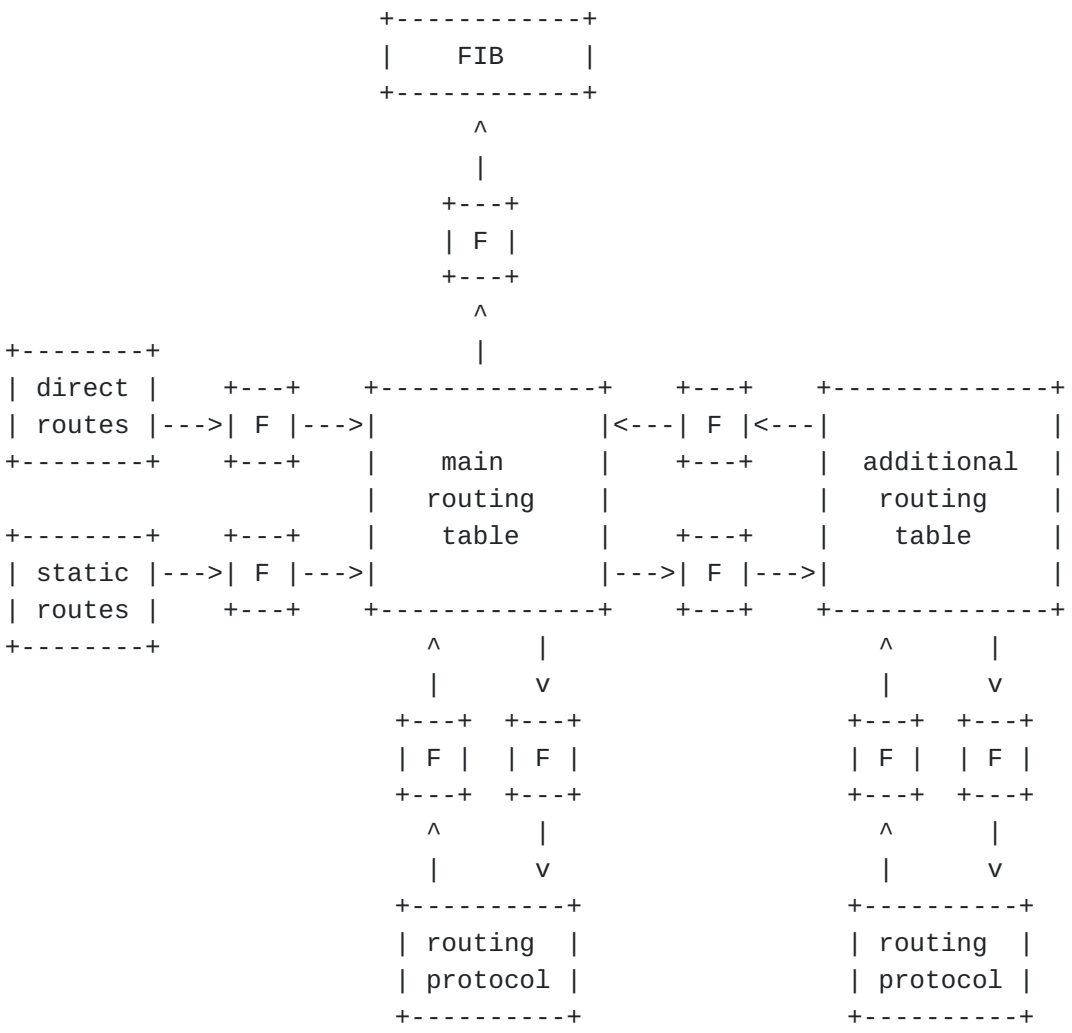


Figure 2 shows an example of a more complicated setup. Several of its features are worth mentioning:

- 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 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 Figure 2.

4.1. Router

Each router instance in the core routing data model represents a (virtual) 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.

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.

In the core routing data model, the contents of routing tables (list of routes) are defined as operational state data. Routing protocol operations 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:

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.

The main routing table SHOULD serve as the 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, and MUST be assigned a type from a selection which includes all routing protocol types supported by the server, such as static, 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 exists.
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 "ietf-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 router instance 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 MAY be configured in zero or multiple instances, although a typical implementation will have exactly one instance.

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 to 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 configuration data and RPC input or output.

- The recommended way of defining configuration data specific to a new protocol is to augment the "routing-protocol" 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 defined in a grouping and then added to route definitions appearing in "routing-table" and in the output part of "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;
}
```

The "when" substatement in the first "augment" guarantees that the new route attributes are only valid when the source protocol is RIP. Finally, RIP-specific configuration data are integrated into the "rt:routing-protocol" node by using the following "augment" statement, which applies only to 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-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";
        }
      }
    }
    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 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 framework(s) 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" module.
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@2011-09-23.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) 2011 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 2011-09-23 {
  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@2011-09-23.yang"

module ietf-routing {

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

  prefix "rt";

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

  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@cesnet.cz>
    ";

  description
    "This module contains YANG definitions of essential components
     that may be used for configuring a routing subsystem.

     Copyright (c) 2011 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 2011-09-23 {
  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.";
  leaf source-protocol {
    type string;
    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.";
  }
}

/* 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.";
      uses route-content;
    }
  }
}

/* Data Nodes */

container routing {
  description
    "Routing parameters.";
  list router {
    key "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 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 connected-routing-table {
        key "name";
        description
          "List of routing tables to which the routing protocol
           instance is connected. No more than one routing
           table may be configured for each AFN/SAFI pair.

           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
           "This must be the name of 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 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. It is expected that this data
                 node will be augmented with information specific for
                 routes of each address family.";
              uses route-content;
            }
          }
          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@2011-09-23.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";
  }

  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.

     Copyright (c) 2011 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 2011-09-23 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Configuration";
}

/* Groupings */

grouping route-content {
  description
    "Specific parameters of IPv4 unicast routes.";
  leaf destination-prefix {
    type inet:ipv4-prefix;
    description
      "IPv4 destination prefix.";
  }
  leaf next-hop {
    type inet:ipv4-address;
    description
      "IPv4 address of the next hop.";
  }
  leaf outgoing-interface {
    type if:interface-ref;
    description
      "Outgoing interface.";
  }
}

/* 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" {
  when "rt:type='rt:static'" {
    description
      "The augment is only valid for the 'static'
       pseudo-protocol.";
  }
  description
    "This augment defines the configuration of the static
     pseudo-protocol with data specific for IPv4 unicast.";
  container ipv4-unicast-static-routes {
    description
      "Configuration of a 'static' pseudo-protocol instance
       consists of a list of routes.";
    list static-route {
      key "id";
      ordered-by "user";
      description
        "A 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.";
      }
      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. 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: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:         iana-afn-safi
namespace:    urn:ietf:params:xml:ns:yang:iana-afn-safi
prefix:       ianaaf
reference:    RFC XXXX
------------------------------------------------------------------
```

9. 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: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.

10. Acknowledgments

The author wishes to thank Martin Bjorklund, Joel Halpern, Tom Petch and Juergen Schoenwaelder for their helpful comments and suggestions.

11. References

11.1. 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.
[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.
[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", Internet-Draft draft-ietf-netmod-interfaces-cfg-02, September 2011.
[YANG-IP]	Bjorklund, M., "A YANG Data Model for IP Configuration", Internet-Draft draft-ietf-netmod-ip-cfg-00, September 2011.

11.2. 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. Appendix Appendix A.1 contains the 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

```
<CODE BEGINS> file "example-rip@2011-09-23.yang"

module example-rip {

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

  prefix "rip";

  import ietf-routing {
    prefix "rt";
  }

  import ietf-interfaces {
    prefix "if";
  }

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

  augment "/rt:routing/rt:router/rt:routing-protocols/"
```

```
        + "rt:routing-protocol" {
    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";
          }
        }
      }
      leaf update-interval {
        type uint8 {
          range "10..60";
        }
        units "seconds";
        default "30";
        description
          "Time interval between periodic updates.";
      }
    }
  }

  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;
  }
}

<CODE ENDS>
```

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 advertising in the NETCONF <hello> message) the following YANG modules:

*ietf-interfaces [YANG-IF],

*ex-ethernet [YANG-IF],

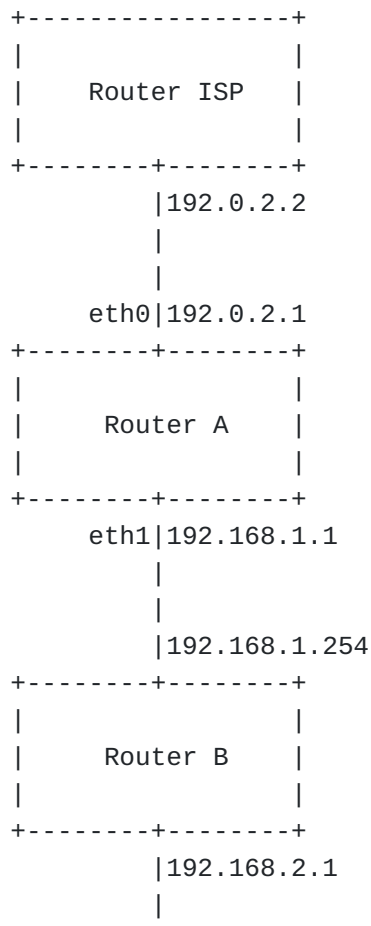
*ietf-ip [YANG-IP],

*ietf-routing (Section 6),

*ietf-ipv4-unicast-routing (Section 7),

*example-rip (Appendix Appendix A.1).

We assume a simple network setup as shown in Figure 12: 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="urn:ietf:params:xml:ns:yang:ietf-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:ipv4>
     <ip:address>
      <ip:ip>192.0.2.1</ip:ip>
      <ip:prefix-length>24</ip:prefix-length>
     </ip:address>
    </ip:ipv4>
   </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>192.168.1.1</ip:ip>
      <ip:prefix-length>24</ip:prefix-length>
     </ip:address>
    </ip:ipv4>
   </if:interface>
  </if:interfaces>
  <rt:routing>
   <rt:router>
    <rt:name>inet-0</rt:name>
    <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>
  <ipv4-unicast-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>
  </ipv4-unicast-static-routes>
 </rt:routing-protocol>
 <rt:routing-protocol>
  <rt:name>rip0</rt:name>
  <rt:description>
   RIP is used on the uplink.  Static routes to the
   internal networks are not advertized in RIP.
  </rt:description>
  <rt:type>rip:rip</rt:type>
  <rt:connected-routing-tables>
   <rt:connected-routing-table>
    <rt:name>ipv4-unicast-main</rt:name>
    <rt:export-filter>deny-all</rt:export-filter>
   </rt:connected-routing-table>
  </rt:connected-routing-tables>
  <rip:rip-configuration>
   <rip:rip-interfaces>
    <rip:rip-interface>
     <rip:name>eth0</rip:name>
    </rip:rip-interface>
   </rip:rip-interfaces>
  </rip:rip-configuration>
 </rt:routing-protocol>
</rt:routing-protocols>
<rt:route-filters>
 <rt:route-filter>
  <rt:name>deny-all</rt:name>
 </rt:route-filter>
</rt:route-filters>
<rt:routing-tables>
 <rt:routing-table>
  <rt:name>ipv4-unicast-fib</rt:name>
  <rt:routes>
   <rt:route>
    <destination-prefix>192.0.2.1/24</destination-prefix>
    <rt:source-protocol>direct</rt:source-protocol>
    <outgoing-interface>eth0</outgoing-interface>
    <rt:last-modified>2011-09-23T17:11:27+01:00</rt:last-modified>
   </rt:route>
   <rt:route>
    <destination-prefix>192.168.1.0/24</destination-prefix>
    <rt:source-protocol>direct</rt:source-protocol>
```

```
    <outgoing-interface>eth1</outgoing-interface>
    <rt:last-modified>2011-09-23T17:11:27+01:00</rt:last-modified>
   </rt:route>
   <rt:route>
    <destination-prefix>192.168.2.0/24</destination-prefix>
    <rt:source-protocol>st0</rt:source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <rt:last-modified>2011-09-23T17:11:32+01:00</rt:last-modified>
   </rt:route>
   <rt:route>
    <destination-prefix>0.0.0.0/0</destination-prefix>
    <rt:source-protocol>rip0</rt:source-protocol>
    <next-hop>192.0.2.2</next-hop>
    <rip:metric>2</rip:metric>
    <rip:tag>64500</rip:tag>
    <rt:last-modified>2011-09-23T18:02:45+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>
    <destination-prefix>192.0.2.1/24</destination-prefix>
    <rt:source-protocol>direct</rt:source-protocol>
    <outgoing-interface>eth0</outgoing-interface>
    <rt:last-modified>2011-09-23T17:11:27+01:00</rt:last-modified>
   </rt:route>
   <rt:route>
    <destination-prefix>192.168.1.0/24</destination-prefix>
    <rt:source-protocol>direct</rt:source-protocol>
    <outgoing-interface>eth1</outgoing-interface>
    <rt:last-modified>2011-09-23T17:11:27+01:00</rt:last-modified>
   </rt:route>
   <rt:route>
    <destination-prefix>192.168.2.0/24</destination-prefix>
    <rt:source-protocol>st0</rt:source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <rt:last-modified>2011-09-23T17:11:32+01:00</rt:last-modified>
   </rt:route>
   <rt:route>
    <destination-prefix>0.0.0.0/0</destination-prefix>
    <rt:source-protocol>rip0</rt:source-protocol>
    <next-hop>192.0.2.2</next-hop>
    <rip:metric>2</rip:metric>
    <rip:tag>64500</rip:tag>
```

```
        <rt:last-modified>2011-09-23T18:02:45+01:00</rt:last-modified>
       </rt:route>
      </rt:routes>
     </rt:routing-table>
    </rt:routing-tables>
   </rt:router>
  </rt:routing>
 </nc:data>
</nc:rpc-reply>
```

Appendix B. Change Log

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

Appendix B.1. 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 Lhotka CESNET EMail: lhotka@cesnet.cz