NETMOD	L. Lhotka
Internet-Draft	CESNET
Intended status: Standards Track	March 03, 2011
Expires: September 04, 2011	

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

Abstract

This document contains a specification of a core YANG data model for IP routing configuration. 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 - routes and routing tables, routing protocol instances, route filters and route pipes.

Status of this Memo

Copyright Notice

Table of Contents

*1. Introduction

*2. Terminology and Notation

*3. Objectives

*4. Design of the Routing Data Model

*4.1. Route

*4.2. Routing Tables

*4.3. Routing Protocol Instances

*4.3.1. Defining New Routing Protocols

*4.4. Route Pipes

*4.5. Route Filters

*5. Core Routing YANG Module

*6. IANA Considerations

*7. Security Considerations

*8. Acknowledgments

*9. References

*9.1. Normative References

*9.2. Informative References

*Appendix A. Example Module for Routing Information Protocol

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

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

*Author's Address

1. Introduction

The NETCONF Data Modeling Language (NETMOD) Working Group has completed the essential specifications for the YANG data modeling language [RFC6020], common data types [RFC6021], and the corresponding data modeling environment and tools [RFC6087], [RFC6110]. The new NETMOD WG charter puts emphasis on the development of a set of core YANG data models for the following subsystems: [it.ifmodel]) was already published [YANG-IF].

1. core system data model,

2. core interface data model,

3. core routing data model.

The initial version of the core interface data model (item
This document contains an initial specification for item [it.rout],
namely the "ietf-routing" YANG module representing the core routing
data model. While the module can be directly used for simple devices
with static routing, its main purpose is to provide basic building
blocks for more complicated setups involving multiple routing protocols
and advanced functions, such as 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

*datastore

*message

*operation

*server

The following terms are defined in [RFC6020]:

*augment

*configuration data

*container

*data model

*data node

*data tree

*data type

*feature

*grouping

- *identity
- *instance identifier
- *leaf-list
- *list
- *mandatory node
- *module
- *operational state data
- *RPC operation
- *submodule

The following terms are defined in [XML-INFOSET]:

- *attribute
- *document
- *document element
- *element
- *information set
- *namespace

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.
- *Simple routing setups, such as static routing, should be configurable in a simple way, ideally without any need to define additional YANG modules.
- *On the other hand, the framework defined by the module 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. Design of the Routing Data Model

The "ietf-routing" YANG module presented in Section 5 defines a data modeling framework with several essential components: routes, routing tables, routing protocol instances, route filters and route pipes. By combining these components in various ways, and filling them with appropriate content models defined in other modules, a broad range of routing setups can be covered.

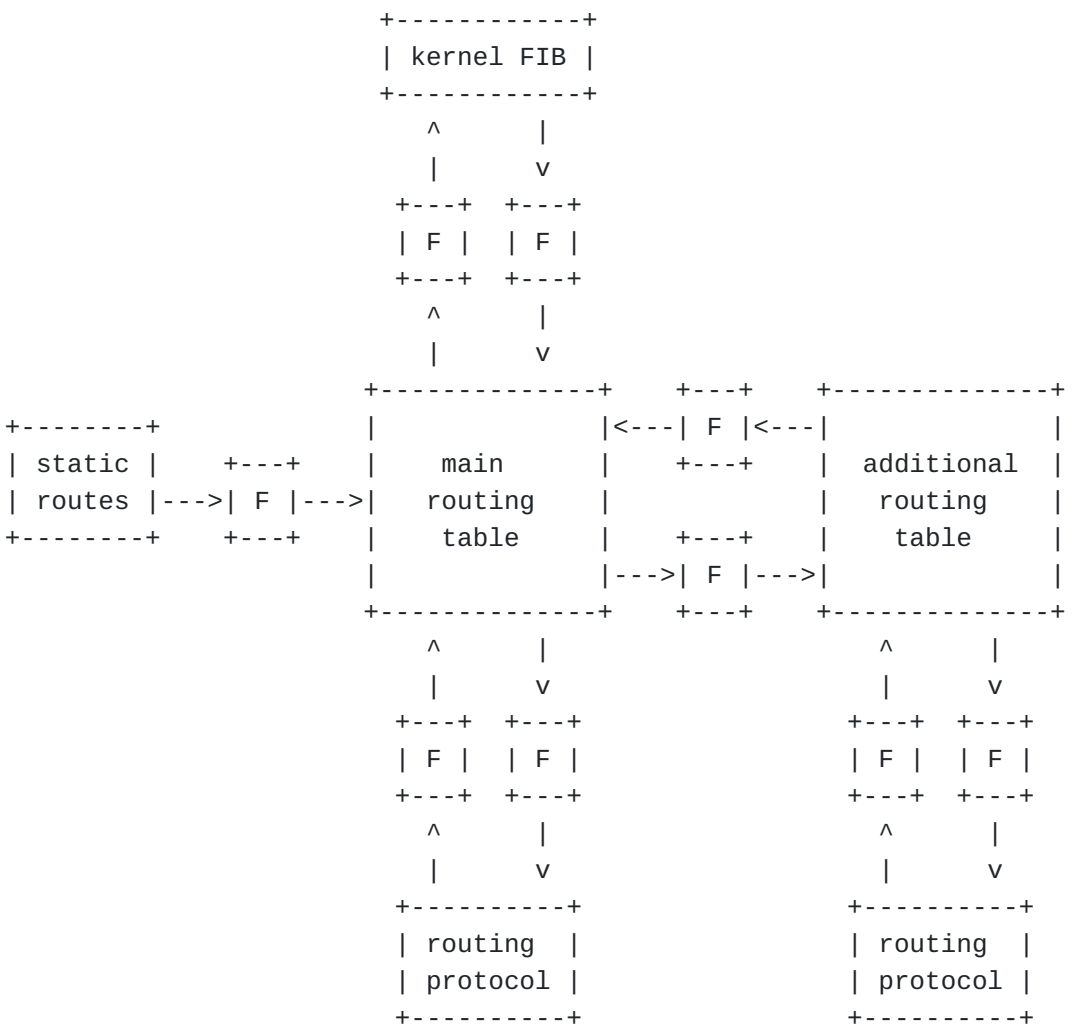

Figure 1 shows an example of a more complicated setup:

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

*Each routing protocol instance, including the static pseudo-protocol instance, is connected to exactly one routing table with

which it can exchange routes (in both directions, except for the static pseudo-protocol).

*Routing tables may also be connected to each other through route pipes and exchange routes in one or both directions.

*The main routing table exchanges routes with the forwarding information base (FIB) in the operating system kernel as follows: active routes from the main routing table are installed in the FIB and used for packet forwarding, and automatic routes set up by the kernel (for example, direct routes to connected networks) are passed to the main routing table.

*Route exchanges along all connections may be controlled by means of route filters denoted by "F" in the figure.

All configuration and operational state data defined by the "ietf-routing" module apear inside the "routing" container. The following subsections describe the individual components of the core routing framework.

4.1. Route

Routes are basic units of information in a routing system. The "ietf-routing" module defines only the following essential route parameters:

*route-type - permitted values are "unicast" (default), "multicast" and "anycast".

*destination-prefix - IP prefix specifying the set of destination addresses for which the route may be used. This parameter 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 parameters is sufficient for a simple static route configuration. It is expected that future modules defining routing protocols will add other route attributes such as metrics or preferences.
Routes 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 arrived.
- last-modified - date and time of last modification, or installation, of the route.

In the core routing data model, routing tables are 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 data model also defines an RPC operation, "delete-route" which allows the client to immediately delete a specified route from a routing table.
Every compliant implementation MUST automatically configure the main routing table. Additional routing tables MAY be configured by adding their unique names to the "configured-routing-tables" leaf-list.

4.3. Routing Protocol Instances

The "ietf-routing" module provides an open-ended framework for defining multiple routing protocol instances. Each of them is identified by a unique name 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 may be passed to any routing protocol connected to the table 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 server 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.

The "direct" pseudo-protocol MUST exist in exactly one instance in any server implementation. It is the source for 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 other 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 "routing-protocol", or to an identity derived from "routing-protocol".

- Additional route attributes MAY be defined. Their definitions have to be inserted as operational state data by augmenting the definition of "route" under "routing-table". Naturally, routes (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. In particular, the module first defines a new identity for the RIP protocol:

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

RIP-specific configuration data are then integrated into the "routing-protocol-instance" node by using the following 'augment' statement, which applies only for routing protocol instances whose type is "rip":

```
augment "/rt:routing/rt:routing-protocol-instances/" +
        "rt:routing-protocol-instance" {
  container rip-configuration {
    when "../rt:type='rip'";
    ...
  }
}
```

4.4. Route Pipes

Route pipes are unidirectional links connecting pairs of routing tables that allow for passing routes from one routing table to another. Each route pipe is identified by a unique name and has two mandatory parameters, "origin" and "recipient", that specify the two routing tables that are being connected.
The transport of routes from the "origin" table to the "recipient" table may be controlled by means of a route filter (see Section 4.5). If no filter is defined, all routes present in the "origin" table MUST also appear in the "recipient" table.

4.5. Route Filters

The "ietf-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 two routing tables connected through a route pipe. Route filters may also manipulate routes, i.e., add, delete, or modify their properties.
By itself, the route filtering framework defined in the "ietf-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 unique name. Its type may be specified by the "type" identity reference - this opens the space for multiple route filtering framework implementations. The only route filter type defined in the "ietf-routing" module carries the default

"route-filter" identity, and represents the "deny all" route filtering policy.

5. Core Routing YANG Module

```
<CODE BEGINS> file "ietf-routing@2011-03-03.yang"

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

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-interfaces {
    prefix if;
  }

  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 basic
     configuration of IP 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).

     Every compliant implementation MUST support IPv4 unicast routing
     and implement at least one (main) routing table, which is
     connected to the OS kernel forwarding information base.";

  revision 2011-03-03;

  /* Features */
```

```
feature ipv6-routing {
  description
    "This feature MUST be advertised by devices supporting IPv6
     routing. Such a device MUST implement at least one extra routing
     table to which all IPv6 unicast routing protocols are connected
     by default.";
}

feature ipv4-multicast-routing {
  description
    "This feature MUST be set by devices supporting IPv4
     multicast routing.  Such a device MUST implement at least one
     extra routing table to which all IPv6 multicast routing
     protocols are connected by default.";
}

feature ipv6-multicast-routing {
  description
    "This feature MUST be set by devices supporting IPv6
     multicast routing.  Such a device MUST implement at least one
     extra routing table to which all IPv6 multicast routing
     protocols are connected by default.";
}

/* Identities */

identity address-family {
  description
    "Base identity from which address family identities are
     derived.";
}

identity af-ipv4 {
  base address-family;
  description
    "IPv4 address family.";
}

identity af-ipv6 {
  base address-family;
  description
    "IPv6 address family.";
}

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
     an instance of this pseudo-protocol.";
}

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

identity route-filter {
  description
    "Base identity for route filters. It also represents the
     empty route filter that blocks everything.";
}

identity route-type {
  description
    "Base identity for route types.";
}

identity unicast {
  base route-type;
  description
    "Unicast route type.";
}

identity multicast {
  base route-type;
  description
    "Multicast route type.";
}

identity anycast {
  base route-type;
  description
    "Anycast route type.";
}

/* Type definitions */

typedef routing-table-ref {
  type leafref {
    path "/routing/configured-routing-tables/name";
  }
```

```
  description
    "This type represents a reference to a configured routing
     table.";
}

typedef routing-protocol-instance-ref {
  type leafref {
    path "/routing/routing-protocol-instances/" +
         "routing-protocol-instance/name";
  }
  description
    "This type represents a reference to a configured routing
     protocol instance.";
}

typedef route-filter-ref {
  type leafref {
    path "/routing/route-filters/route-filter/name";
  }
  description
    "This type represents a reference to a configured route
     filter.";
}

/* Groupings */

grouping ip-route-content {
  description
    "Components of an IP route.";
  leaf type {
    type identityref {
      base route-type;
    }
    default "unicast";
  }
  leaf destination-prefix {
    type inet:ip-prefix;
    mandatory true;
    description
      "The set of destination addresses for which the route may
       be used.";
  }
  leaf next-hop {
    type inet:ip-address;
    description
      "IP 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 parameter is mainly
       used in direct routes.";
  }
}

rpc delete-route {
  description
    "This operation deletes a route with given parameters from
     a specified routing table. <ok> is returned by the server
     upon successful completion.";
  input {
    container route {
      description
        "All routes that match this parameter MUST be deleted
         from the target routing table.";
      uses ip-route-content;
    }
    leaf routing-table {
      type routing-table-ref;
      description
        "This parameter specifies the target routing
         table.";
    }
  }
}

/* Data tree */

container routing {
  description
    "IP routing parameters.";
  container configured-routing-tables {
    description
      "Names of configured routing tables. Their contents are
       available as operational state data under 'routing-tables'. At
       least one (main) table MUST be configured for every supported
       address family. This default routing table is usually
       connected to the main kernel forwarding table.";
    leaf-list name {
      type string;
      min-elements 1;
    }
  }
  container routing-protocol-instances {
    description
      "Container for configured routing protocol instances.
```

```
   Every implementation MUST preconfigure the 'direct'
   pseudo-protocol instance providing the routes to directly
   connected networks.";
 list routing-protocol-instance {
   key "name";
   container static-routes {
     when "../type='static'";
     description
       "Configuration of a 'static' pseudo-protocol instance
        consists of a list of routes.";
     list static-route {
       key "id";
       leaf id {
         type string;
       }
       leaf description {
         type string;
       }
       uses ip-route-content;
     }
   }
   leaf name {
     type string;
   }
   leaf description {
     type string;
   }
   leaf address-family {
     type identityref {
       base address-family;
     }
     default "af-ipv4";
     description
       "Address family used by the routing protocol instance.";
   }
   leaf type {
     type identityref {
       base routing-protocol;
     }
     mandatory true;
     description
       "Type of the routing protocol.";
   }
   leaf routing-table {
     type routing-table-ref;
     description
       "The routing table to which the protocol instance is
        connected. By default it is the main routing table for the
        given address family.";
```

```
    }
    leaf import-filter {
      type route-filter-ref;
      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 route-filter-ref;
      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
         anywhere.";
    }
  }
}
container route-pipes {
  description
    "Route pipes facilitate transport of routes between pairs
     of routing tables.";
  list route-pipe {
    key "name";
    description
      "A route-pipe is a unidirectional connection between
       'origin' and 'recipient'.";
    leaf name {
      type string;
    }
    leaf description {
      type string;
    }
    leaf origin {
      type routing-table-ref;
      mandatory true;
      description
        "The originating routing-table.";
    }
    leaf recipient {
      type routing-table-ref;
      mandatory true;
```

```
      description
        "The receiving routing-table.";
    }
    leaf route-filter {
      type route-filter-ref;
      description
        "All routes passing through the route pipe are filtered by
         the route filter referred to by this leaf. If it is not
         present, all routes from 'origin' are passed to
         'destination'.";
    }
  }
}
container route-filters {
  description
    "Non-trivial route filters are expected to be defined in
     other modules.";
  list route-filter {
    key "name";
    description
      "A route filter is used for filtering routes between
       routing protocol instances and routing tables (import
       filter) and vice versa (export filter), and in route pipes
       that connect pairs of routing tables.";
    leaf name {
      type string;
    }
    leaf description {
      type string;
    }
    leaf type {
      type identityref {
        base route-filter;
      }
      default "route-filter";
      description
        "Type of the route-filter. The default value
         represents an all-blocking filter.";
    }
  }
}

/* Operational state data */

container routing-tables {
  config false;
  description
    "Routing tables and their contents.";
  list routing-table {
```

```
        min-elements 1;
        description
          "Exactly one routing table MUST be present for every 'name'
           entry appearing in '/routing/configured-routing-tables'.";
        leaf name {
          type routing-table-ref;
        }
        leaf address-family {
          type identityref {
            base address-family;
          }
          default "af-ipv4";
          description
            "Address family of all routes in the routing table.";
        }
        list route {
          leaf source-protocol {
            type routing-protocol-instance-ref;
            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.";
          }
          uses ip-route-content;
        }
      }
    }
  }
}
```

<CODE ENDS>

6. IANA Considerations

This document requests the following registration of a namespace URI 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.
---------------------------------------------------
```

7. Security Considerations

TBD.

8. Acknowledgments

The author wishes to thank the following individuals who provided helpful suggestions and/or comments on this document: TBD.

9. References

9.1. Normative References

[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.
[XML-INFOSET]	Tobin, R. and J. Cowan, "XML Information Set (Second Edition)", World Wide Web Consortium Recommendation REC-xml-infoset-20040204, February 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.
[YANG-IF]	Bjorklund, M, "A YANG Data Model for Interface Configuration", Internet-Draft draft-bjorklund-netmod-interfaces-cfg-00, December 2010.

9.2. Informative References

[RFC6087]	Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.
[RFC6110]	Lhotka, L., "Mapping YANG to Document Schema Definition Languages and Validating NETCONF Content", RFC 6110, February 2011.

Appendix A. Example Module for Routing Information Protocol

This appendix demonstrates how the "ietf-routing" module can be extended to support a new routing protocol. Appendix Appendix A.1 contains a YANG module that 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. This module also imports the "ietf-interfaces" module defined in [YANG-IF].
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. A route filter is also defined in order to prevent static routes to private networks from being redistributed to RIP. The instance document also uses data nodes from the two example YANG modules "ex-ethernet" and "ex-ip" defined in [YANG-IF].

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

```
<?xml version="1.0" encoding="utf-8"?>

<nc:rpc-reply
    message-id="101"
    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:rip="http://example.com/rip"
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <nc:data>
    <if:interfaces>
      <if:interface>
        <if:name>eth0</if:name>
        <if:type>eth:ethernet</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>eth:ethernet</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>
    <routing>
      <configured-routing-tables>
        <name>rt0</name>
      </configured-routing-tables>
      <routing-protocol-instances>
        <routing-protocol-instance>
          <name>direct</name>
          <type>direct</type>
        </routing-protocol-instance>
        <routing-protocol-instance>
          <name>st0</name>
          <description>
            Static routing is used for the internal network.
          </description>
```

```
    <type>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>
    <type>rip:rip</type>
    <export-filter>to-rip</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>to-rip</name>
    <description>
      Block redistribution of static routes.
    </description>
  </route-filter>
</route-filters>
<routing-tables>
  <routing-table>
    <name>rt0</name>
    <route>
      <destination-prefix>192.168.1.0/24</destination-prefix>
      <source-protocol>direct</source-protocol>
      <outgoing-interface>eth0</outgoing-interface>
      <last-modified>2010-02-24T17:11:23+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>
      <rip:tag>64500</rip:tag>
      <last-modified>2010-02-24T17:11:27+01:00</last-modified>
    </route>
    <route>
      <destination-prefix>0.0.0.0/0</destination-prefix>
      <next-hop>192.0.2.2</next-hop>
```

```
            <rip:metric>2</rip:metric>
            <rip:tag>64500</rip:tag>
            <source-protocol>rip0</source-protocol>
            <last-modified>2010-03-03T13:00:23+01:00</last-modified>
          </route>
        </routing-table>
      </routing-tables>
    </routing>
  </nc:data>
</nc:rpc-reply>
```

Author's Address

Ladislav Lhotka Lhotka CESNET EMail: lhotka@cesnet.cz