

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 31, 2013

A. Atlas, Ed.  
T. Nadeau  
Juniper Networks  
D. Ward  
Cisco Systems  
July 30, 2012

Interface to the Routing System Problem Statement  
draft-atlas-irs-problem-statement-00

Abstract

As modern networks grow in scale and complexity, the need for rapid and dynamic control increases. With scale, the need to automate even the simplest operations is important, but even more critical is the ability to quickly interact with more complex operations such as policy-based controls.

In order to enable applications to have access to and control over information in the Internet's routing system, we need a publically documented interface specification. The interface needs to support real-time, transaction-based interactions using efficient data models and encodings. Furthermore, the interface must support a variety of use cases including those where verified control feed-back loops are needed.

This document expands upon these statements of requirements to provide a problem statement for an interface to the Internet routing system.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 31, 2013.

## Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. IRS Model and Problem Area for The IETF . . . . .	3
3. Standard Data-Models of Routing State for Installation . . . . .	5
4. Learning Router Information . . . . .	5
5. Desired Aspects of a Protocol for IRS . . . . .	6
6. Existing Management Interfaces . . . . .	7
7. Acknowledgements . . . . .	8
8. IANA Considerations . . . . .	8
9. Security Considerations . . . . .	8
10. Informative References . . . . .	9
Appendix A. Gaps and Concerns for SNMP . . . . .	9
Appendix B. Gaps and Concerns with NetConf . . . . .	10
Authors' Addresses . . . . .	11

## 1. Introduction

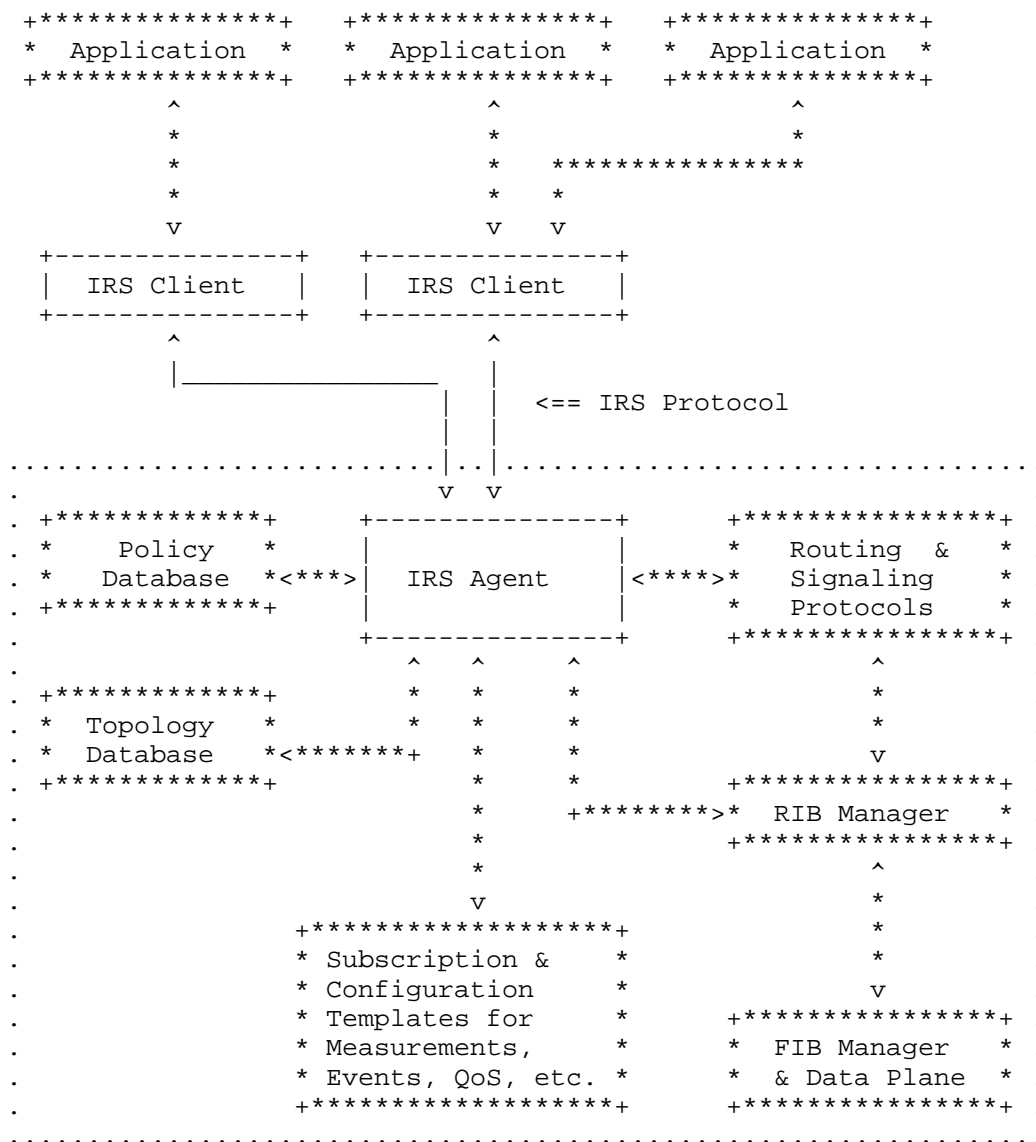
As modern networks grow in scale and complexity, the need for rapid and dynamic control increases. With scale, the need to automate even the simplest operations is important, but even more critical is the ability to quickly interact with more complex operations such as policy-based controls.

With complexity comes the need for more sophisticated automated applications and orchestration software that can process large quantities of data, run complex algorithms, and adjust the routing state as required in order to support the applications, their calculations and their policies. Changes made to the routing state of a network by external applications must be verifiable by those applications to ensure that the correct state has been installed in the right places.

Mechanisms to support the requirements outlined above have been developed piecemeal as proprietary solutions to specific situations and needs. A standard protocol, clear operations that an application can initiate with that protocol, and data-models to support such actions would facilitate wide-scale deployment of interoperable applications and routing systems. That a protocol designed to facilitate rapid, isolated, secure, and dynamic routing changes is needed motivates the creation of an Interface to The Routing System (IRS).

## 2. IRS Model and Problem Area for The IETF

Managing a network of deployed devices running a variety of routing protocols involves interactions among multiple different functions and components that exist within the network. Some of these components are virtual while some are physical; all should be made available to be managed and manipulated by applications, given that appropriate access, authentication, and policy hurdles have been crossed. The management of only some of these components requires standardization, as others have already been standardized. The IRS model is intended to incorporate existing mechanisms where appropriate, and to build extensions and new protocols where needed. The IRS model and problem area proposed for IETF work is illustrated in Figure 1.



```
<--> interfaces inside the scope of IRS
```

```

+--+  objects inside the scope of IRS

```

```
<**> interfaces NOT within the scope of IRS
```

\*\*\*+ objects NOT within the scope of IRS

```
.... boundary of a router participating in the IRS
```

Figure 1: IRS model and Problem Area

A critical aspect of IRS is defining a suitable protocol or protocols to carry messages between the IRS Clients and the IRS Agent, and defining the encapsulation of data within those messages. This should provide a clear transfer syntax that is straightforward for applications to use (e.g., a Web Services design paradigm), and should provide the key features specified in Section 5.

The second critical aspect is semantic-aware data-models for information in the routing system and in a topology database. The data-models should be separable across different features of the managed components, versioned, and combine to provide a network data-model.

### 3. Standard Data-Models of Routing State for Installation

There is a need to be able to precisely control routing and signaling state based upon policy or external measures. This can range from simple static routes to policy-based routing to static multicast replication and routing state. This means that the data model employed needs to handle indirection as well as different types of tunneling and encapsulation. The relevant MIB modules (for example [RFC4292]) lack the necessary generality and flexibility. In addition, by having IRS focus initially on interfaces to the RIB layer (e.g. RIB, LFIB, multicast RIB, policy-based routing), the ability to use routing indirection allows flexibility and functionality that can't be as easily obtained at the forwarding layer.

Efforts to provide this level of control have focused on standardizing data models that describe the forwarding plane (e.g. ForCES [RFC3746]). IRS recognizes that the routing system and a router's OS provide useful mechanisms that applications could usefully harness to accomplish application-level goals.

In addition to interfaces to the RIB layer, there is a need to configure the various routing and signaling protocols with differing dynamic state based upon application-level policy decisions. The range desired is not available via MIBs at the present time.

### 4. Learning Router Information

A router has information that applications may require so that they can understand the network, verify that programmed state is installed in the forwarding plane, measure the behavior of various flows, and

understand the existing configuration and state of the router. IRS provides a framework for applications to register for asynchronous notifications and for them to make specific requests for information.

Although there are efforts to extend the topological information available, even the best of these (e.g., BGP-LS [I-D.gredler-idr-ls-distribution]) still provides only the current active state as seen at the IGP layer and above. Detailed topological state that provides more information than the current functional status is needed by applications; only the active paths or links are known versus those desired or unknown to the routing topology.

For applications to have a feedback loop that includes awareness of the relevant traffic, an application must be able to request the measurement and timely, scalable reporting of data. While a mechanism such as IPFIX [RFC5470] may be the facilitator for delivering the data, the need for an application to be able to dynamically request that measurements be taken and data delivered is critical.

There are a wide range of events that applications could use for either verification of router state before other network state is changed (e.g. that a route has been installed), to act upon changes to relevant routes by others, or upon router events (e.g. link up/down). While a few of these (e.g. link up/down) may be available via MIB Notifications today, the full range is not - nor is there the ability to set up the router to trigger different actions upon an event's occurrence.

## 5. Desired Aspects of a Protocol for IRS

This section describes required aspects of a protocol that could support IRS. Whether such a protocol is built upon extending existing mechanisms or requires a new mechanism requires further investigation.

The key aspects needed in an interface to the routing system are:

**Multiple Simultaneous Asynchronous Operations:** A single application should be able to send multiple operations to IRS without needing to wait for each to complete before sending the next.

**Configuration Not Re-Processed:** When an IRS operation is processed, it does not require that any of the configuration be processed. I.e., the desired behavior is orthogonal to the static configuration.

**Duplex:** Communications can be established by either the router or the application. Similarly, events, acknowledgements, failures, operations, etc. can be sent at any time by both the router and the application. The IRS is not a pure pull-model where only the application queries to pull responses.

**High-Throughput:** At a minimum, the IRS Agent and associated router should be able to handle hundreds of simple operations per second.

**Responsive:** It should be possible to complete simple operations within a sub-second time-scale.

**Multi-Channel:** It should be possible for information to be communicated via the interface from different components in the router without requiring going through a single channel. For example, for scaling, some exported data or events may be better sent directly from the forwarding plane, while other interactions may come from the control-plane. Thus a single TCP session would not be a good match.

**Timing of State Installation and Expiration:** The ability to have state installed with different lifetimes and different start-times is very valuable. In particular, the ability of an IRS client to request that a pre-sent operation be started based upon a dynamic event would provide a powerful functionality.

To extract information in a scalable fashion that is more easily used by applications, the ability to specify filtering constructs in an operation requesting data or requesting an asynchronous notification is very valuable.

## 6. Existing Management Interfaces

This section discusses the combination of the abstract data models, their representation in a data language, and the transfer protocol commonly used with them as a single entity. While other combinations are possible, the combinations described are those that have significant deployment.

There are three basic ways that routers are managed. The most popular is the command line interface (CLI), which allows both configuration and learning of device state. This is a proprietary interface resembling a UNIX shell that allows for very customized control and observation of a device, and, specifically of interest in this case, its routing system. Some form of this interface exists on almost every device (virtual or otherwise). Processing of information returned to the CLI (called "screen scraping") is a

burdensome activity because the data is normally formatted for use by a human operator, and because the layout of the data can vary from device to device, and between different software versions. Despite its ubiquity, this interface has never been standardized and is unlikely to ever be standardized. IRS does not involve CLI standardization.

The second most popular interface for interrogation of a device's state, statistics, and configuration is The Simple Network Management Protocol (SNMP) and a set of relevant standards-based and proprietary Management Information Base (MIB) modules. SNMP has a strong history of being used by network managers to gather statistical and state information about devices, including their routing systems. However, SNMP is very rarely used to configure a device or any of its systems for reasons that vary depending upon the network operator. Some example reasons include complexity, the lack of desired configuration semantics (e.g., configuration "roll-back", "sandboxing" or configuration versioning), and the difficulty of using the semantics (or lack thereof) as defined in the MIB modules to configure device features. Therefore, SNMP is not considered as a candidate solution for the problems motivating IRS.

Finally, the IETF's Network Configuration (or NetConf) protocol has made many strides at overcoming most of the limitations around configuration that were just described. However, the lack of standard data models have hampered the adoption of NetConf.

## 7. Acknowledgements

The authors would like to thank Ken Gray for their suggestions and review.

## 8. IANA Considerations

This document includes no request to IANA.

## 9. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting of detailed router state. More investigation remains to fully define the security requirements, such as authorization and authentication levels.



## 10. Informative References

- [I-D.gredler-idr-ls-distribution]  
Gredler, H., Medved, J., Previdi, S., and A. Farrel,  
"North-Bound Distribution of Link-State and TE Information  
using BGP", draft-gredler-idr-ls-distribution-02 (work in  
progress), July 2012.
- [RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal,  
"Forwarding and Control Element Separation (ForCES)  
Framework", RFC 3746, April 2004.
- [RFC4292] Haberman, B., "IP Forwarding Table MIB", RFC 4292,  
April 2006.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek,  
"Architecture for IP Flow Information Export", RFC 5470,  
March 2009.

## Appendix A. Gaps and Concerns for SNMP

Though SNMP can allow state to be written, the overhead of the required infrastructure is quite high. Clients and servers that wish to use SNMP must build in and understand a large number of MIB modules, including many proprietary modules. Even when ignoring the overhead in building the SNMP processing and handling functions into an application, these properties lend themselves well to read-only operations. A critical lack in MIB modules for read-write (i.e., for configuration) operations is that there is no semantic understanding of the objects defined in the modules encoded in those modules. Any required semantic knowledge must be specifically hand-coded into applications or ignored. Further, many MIB modules do not allow the writing of state, and this limits coverage; owing to the cumbersome nature, there has not been interest in increasing coverage.

An additional deficiency in using SNMP MIB modules either for reading or writing comes in the inherent co-mingling of semantics and syntax in the form of indexing requirements. SNMP MIB modules contain tables that also define an index format. This format is then translated - often mapped onto - a device's actual implementation. Such a mapping means an implementation either matches the module's indexing during searches or not; if not, then an implementation is slowed down when it searches for objects. Even in not-so-extreme cases, such slow performance can result in the SNMP manager's request timing-out owing to the delay of the SNMP agent's response.

For example, if a search of N\*M objects is stipulated as (N, M) in

the standard MIB module, but the implementation happens to choose to index its tables internally as (M, N), this will result in search times of  $O(N^2)$ . When N or M become large, as they do in routing tables, this results in wasted processing cycles for the device, and either extremely long wait times for queries, or simply a lack of answers to queries. It is a clear requirement for the IRS to not suffer from this issue.

In addition to these difficulties, SNMP matches up to the key needed aspects as follows:

Multiple Simultaneous Asynchronous Operations: Supported, but difficult for configuration.

Configuration Not Re-Processed: Supported

Duplex: The manager must initiate communications with the SNMP agent on the router. With the limited exception of Notifications and InformRequests defined in a MIB module, this is a pull model.

High-Throughput: Possible

Responsive: Possible

Multi-Channel: Possible

The key gaps identified for SNMP are:

- a. Infrastructure Overhead
- b. Lack of Semantic Information in the Data-Model
- c. Required Indexing, from mingling of semantics and syntax, badly impacting performance or driving implementation decisions.
- d. Limited interest and use for configuration
- e. Communication model isn't naturally duplex.

#### Appendix B. Gaps and Concerns with NetConf

While NetConf has solved many of the deficiencies present in SNMP in terms of configuration, it still does not satisfy a number of requirements needed to manage today's routing information. First, the lack of standard data models have hampered the adoption of NetConf; a significant amount of per-vendor customization is still needed. The transport mechanisms that are currently defined (e.g.,

SOAP/BEEP) for NetConf are not those commonly used by modern applications (e.g., ReST or JSON).

NetConf primarily facilitates configuration rather than reading of state or handling asynchronous events.

NetConf matches up to the key needed aspects as follows:

Multiple Simultaneous Asynchronous Operations: Not Possible

Configuration Not Re-Processed: Not Possible

Duplex: Not Possible - strict pull model.

High-Throughput: Unlikely - Can depend on configuration size

Responsive: Unlikely - Can depend on configuration size

Multi-Channel: Not Possible

#### Authors' Addresses

Alia Atlas (editor)  
Juniper Networks  
10 Technology Park Drive  
Westford, MA 01886  
USA

Email: [akatlas@juniper.net](mailto:akatlas@juniper.net)

Thomas Nadeau  
Juniper Networks  
1194 N. Mathilda Ave.  
Sunnyvale, CA 94089  
USA

Email: [tnadeau@juniper.net](mailto:tnadeau@juniper.net)

Dave Ward  
Cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: wardd@cisco.com



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: December 7, 2012

M. Bjorklund  
Tail-f Systems  
June 5, 2012

IANA Interface Type and Address Family YANG Modules  
draft-ietf-netmod-iana-if-type-04

Abstract

This document defines the initial versions of the iana-if-type and iana-afn-safi YANG modules.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 7, 2012.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. IANA Maintained Interface Type YANG Module . . . . .	4
3. IANA Maintained AFN and SAFI YANG Module . . . . .	36
4. IANA Considerations . . . . .	45
5. Security Considerations . . . . .	47
6. Normative References . . . . .	48
Author's Address . . . . .	49

## 1. Introduction

This document defines the initial version of the iana-if-type and iana-afn-safi YANG modules, for interface type definitions, and Address Family Numbers (AFN) and Subsequent Address Family Identifiers (SAFI), respectively.

The iana-if-type module reflects IANA's existing "ifType definitions" registry. The latest revision of the module can be obtained from the IANA web site.

Whenever a new interface type is added to the "ifType definitions" registry, the IANAifType-MIB and the iana-if-type YANG module are updated by IANA.

The iana-afn-safi module reflects IANA's existing "Address Family Numbers" and "Subsequent Address Family Identifiers" registries.

Whenever a new address family number is added to the "Address Family Numbers" registry, the IANA-ADDRESS-FAMILY-NUMBERS-MIB and the iana-afn-safi YANG module are updated by IANA.

Whenever a new subsequent address family identifier is added to the "Subsequent Address Family Identifiers" registry, the iana-afn-safi YANG module is updated by IANA.



## 2. IANA Maintained Interface Type YANG Module

```
<CODE BEGINS> file "iana-if-type.yang"

module iana-if-type {
  namespace "urn:ietf:params:xml:ns:yang:iana-if-type";
  prefix ianaift;

  organization "IANA";
  contact
    "      Internet Assigned Numbers Authority

    Postal: ICANN
           4676 Admiralty Way, Suite 330
           Marina del Rey, CA 90292

    Tel:    +1 310 823 9358
    E-Mail: iana@iana.org";
  description
    "This YANG module defines the iana-if-type typedef, which
    contains YANG definitions for IANA-registered interface types.

    This YANG module is maintained by IANA, and reflects the
    'ifType definitions' registry.

    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.";
  // RFC Ed.: replace XXXX with actual RFC number and remove this
  // note.

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  revision 2012-06-05 {
    description
      "Initial revision.";
```

```
reference
  "RFC XXXX: TITLE";
}

typedef iana-if-type {
  type enumeration {
    enum "other" {
      value 1;
      description
        "None of the following";
    }
    enum "regular1822" {
      value 2;
    }
    enum "hdl1822" {
      value 3;
    }
    enum "ddnX25" {
      value 4;
    }
    enum "rfc877x25" {
      value 5;
      reference
        "RFC 1382 - SNMP MIB Extension for the X.25 Packet Layer";
    }
    enum "ethernetCsmacd" {
      value 6;
      description
        "For all ethernet-like interfaces, regardless of speed,
        as per RFC3635.";
      reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
    }
    enum "iso88023Csmacd" {
      value 7;
      status deprecated;
      description
        "Deprecated via RFC3635.
        Use ethernetCsmacd(6) instead.";
      reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
    }
    enum "iso88024TokenBus" {
      value 8;
    }
    enum "iso88025TokenRing" {
```

```
    value 9;
}
enum "iso88026Man" {
    value 10;
}
enum "starLan" {
    value 11;
    status deprecated;
    description
        "Deprecated via RFC3635.
        Use ethernetCsmacd(6) instead.";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
}
enum "proteon10Mbit" {
    value 12;
}
enum "proteon80Mbit" {
    value 13;
}
enum "hyperchannel" {
    value 14;
}
enum "fddi" {
    value 15;
    reference
        "RFC 1512 - FDDI Management Information Base";
}
enum "lapb" {
    value 16;
    reference
        "RFC 1381 - SNMP MIB Extension for X.25 LAPB";
}
enum "sdlc" {
    value 17;
}
enum "dsl" {
    value 18;
    description
        "DSL-MIB";
    reference
        "RFC 4805 - Definitions of Managed Objects for the
        DSL, J1, E1, DS2, and E2 Interface Types";
}
enum "e1" {
    value 19;
    status obsolete;
}
```

```
    description
        "Obsolete see DS1-MIB";
    reference
        "RFC 4805 - Definitions of Managed Objects for the
         DS1, J1, E1, DS2, and E2 Interface Types";
}
enum "basicISDN" {
    value 20;
    description
        "see also RFC2127";
}
enum "primaryISDN" {
    value 21;
}
enum "propPointToPointSerial" {
    value 22;
    description
        "proprietary serial";
}
enum "ppp" {
    value 23;
}
enum "softwareLoopback" {
    value 24;
}
enum "eon" {
    value 25;
    description
        "CLNP over IP";
}
enum "ethernet3Mbit" {
    value 26;
}
enum "nsip" {
    value 27;
    description
        "XNS over IP";
}
enum "slip" {
    value 28;
    description
        "generic SLIP";
}
enum "ultra" {
    value 29;
    description
        "ULTRA technologies";
}
```

```
enum "ds3" {
    value 30;
    description
        "DS3-MIB";
    reference
        "RFC 3896 - Definitions of Managed Objects for the
        DS3/E3 Interface Type";
}
enum "sip" {
    value 31;
    description
        "SMDS, coffee";
    reference
        "RFC 1694 - Definitions of Managed Objects for SMDS
        Interfaces using SMIV2";
}
enum "frameRelay" {
    value 32;
    description
        "DTE only.";
    reference
        "RFC 2115 - Management Information Base for Frame Relay
        DTEs Using SMIV2";
}
enum "rs232" {
    value 33;
    reference
        "RFC 1659 - Definitions of Managed Objects for RS-232-like
        Hardware Devices using SMIV2";
}
enum "para" {
    value 34;
    description
        "parallel-port";
    reference
        "RFC 1660 - Definitions of Managed Objects for
        Parallel-printer-like Hardware Devices using
        SMIV2";
}
enum "arcnet" {
    value 35;
    description
        "arcnet";
}
enum "arcnetPlus" {
    value 36;
    description
        "arcnet plus";
```

```
}
enum "atm" {
  value 37;
  description
    "ATM cells";
}
enum "miox25" {
  value 38;
  reference
    "RFC 1461 - SNMP MIB extension for Multiprotocol
      Interconnect over X.25";
}
enum "sonet" {
  value 39;
  description
    "SONET or SDH";
}
enum "x25ple" {
  value 40;
  reference
    "RFC 2127 - ISDN Management Information Base using SMIV2";
}
enum "iso88022llc" {
  value 41;
}
enum "localTalk" {
  value 42;
}
enum "smdsDxi" {
  value 43;
}
enum "frameRelayService" {
  value 44;
  description
    "FRNETSERV-MIB";
  reference
    "RFC 2954 - Definitions of Managed Objects for Frame
      Relay Service";
}
enum "v35" {
  value 45;
}
enum "hssi" {
  value 46;
}
enum "hippi" {
  value 47;
}
```

```
enum "modem" {  
    value 48;  
    description  
        "Generic modem";  
}  
enum "aal5" {  
    value 49;  
    description  
        "AAL5 over ATM";  
}  
enum "sonetPath" {  
    value 50;  
}  
enum "sonetVT" {  
    value 51;  
}  
enum "smdsIcip" {  
    value 52;  
    description  
        "SMDS InterCarrier Interface";  
}  
enum "propVirtual" {  
    value 53;  
    description  
        "proprietary virtual/internal";  
    reference  
        "RFC 2863 - The Interfaces Group MIB";  
}  
enum "propMultiplexor" {  
    value 54;  
    description  
        "proprietary multiplexing";  
    reference  
        "RFC 2863 - The Interfaces Group MIB";  
}  
enum "ieee80212" {  
    value 55;  
    description  
        "100BaseVG";  
}  
enum "fibreChannel" {  
    value 56;  
    description  
        "Fibre Channel";  
}  
enum "hippiInterface" {  
    value 57;  
    description
```

```
    "HIPPI interfaces";
}
enum "frameRelayInterconnect" {
    value 58;
    status obsolete;
    description
        "Obsolete use either
        frameRelay(32) or frameRelayService(44).";
}
enum "aflane8023" {
    value 59;
    description
        "ATM Emulated LAN for 802.3";
}
enum "aflane8025" {
    value 60;
    description
        "ATM Emulated LAN for 802.5";
}
enum "cctEmul" {
    value 61;
    description
        "ATM Emulated circuit";
}
enum "fastEther" {
    value 62;
    status deprecated;
    description
        "Obsoleted via RFC3635.
        ethernetCsmacd(6) should be used instead";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
}
enum "isdn" {
    value 63;
    description
        "ISDN and X.25";
    reference
        "RFC 1356 - Multiprotocol Interconnect on X.25 and ISDN
        in the Packet Mode";
}
enum "v11" {
    value 64;
    description
        "CCITT V.11/X.21";
}
enum "v36" {
```



```
    value 65;
    description
        "CCITT V.36";
}
enum "g703at64k" {
    value 66;
    description
        "CCITT G703 at 64Kbps";
}
enum "g703at2mb" {
    value 67;
    status obsolete;
    description
        "Obsolete see DS1-MIB";
}
enum "qllc" {
    value 68;
    description
        "SNA QLLC";
}
enum "fastEtherFX" {
    value 69;
    status deprecated;
    description
        "Obsoleted via RFC3635
        ethernetCsmacd(6) should be used instead";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
}
enum "channel" {
    value 70;
    description
        "channel";
}
enum "ieee80211" {
    value 71;
    description
        "radio spread spectrum";
}
enum "ibm370parChan" {
    value 72;
    description
        "IBM System 360/370 OEMI Channel";
}
enum "escon" {
    value 73;
    description
```

```
        "IBM Enterprise Systems Connection";
    }
    enum "dls" {
        value 74;
        description
            "Data Link Switching";
    }
    enum "isdns" {
        value 75;
        description
            "ISDN S/T interface";
    }
    enum "isdnu" {
        value 76;
        description
            "ISDN U interface";
    }
    enum "lapd" {
        value 77;
        description
            "Link Access Protocol D";
    }
    enum "ipSwitch" {
        value 78;
        description
            "IP Switching Objects";
    }
    enum "rsrb" {
        value 79;
        description
            "Remote Source Route Bridging";
    }
    enum "atmLogical" {
        value 80;
        description
            "ATM Logical Port";
        reference
            "RFC 3606 - Definitions of Supplemental Managed Objects
              for ATM Interface";
    }
    enum "ds0" {
        value 81;
        description
            "Digital Signal Level 0";
        reference
            "RFC 2494 - Definitions of Managed Objects for the DS0
              and DS0 Bundle Interface Type";
    }
}
```

```
enum "ds0Bundle" {
    value 82;
    description
        "group of ds0s on the same ds1";
    reference
        "RFC 2494 - Definitions of Managed Objects for the DS0
        and DS0 Bundle Interface Type";
}
enum "bsc" {
    value 83;
    description
        "Bisynchronous Protocol";
}
enum "async" {
    value 84;
    description
        "Asynchronous Protocol";
}
enum "cnr" {
    value 85;
    description
        "Combat Net Radio";
}
enum "iso88025Dtr" {
    value 86;
    description
        "ISO 802.5r DTR";
}
enum "eplrs" {
    value 87;
    description
        "Ext Pos Loc Report Sys";
}
enum "arap" {
    value 88;
    description
        "Appletalk Remote Access Protocol";
}
enum "propCnls" {
    value 89;
    description
        "Proprietary Connectionless Protocol";
}
enum "hostPad" {
    value 90;
    description
        "CCITT-ITU X.29 PAD Protocol";
}
```

```
enum "termPad" {  
    value 91;  
    description  
        "CCITT-ITU X.3 PAD Facility";  
}  
enum "frameRelayMPI" {  
    value 92;  
    description  
        "Multiproto Interconnect over FR";  
}  
enum "x213" {  
    value 93;  
    description  
        "CCITT-ITU X213";  
}  
enum "adsl" {  
    value 94;  
    description  
        "Asymmetric Digital Subscriber Loop";  
}  
enum "radsl" {  
    value 95;  
    description  
        "Rate-Adapt. Digital Subscriber Loop";  
}  
enum "sdsl" {  
    value 96;  
    description  
        "Symmetric Digital Subscriber Loop";  
}  
enum "vdsl" {  
    value 97;  
    description  
        "Very H-Speed Digital Subscrib. Loop";  
}  
enum "iso88025CRFPInt" {  
    value 98;  
    description  
        "ISO 802.5 CRFP";  
}  
enum "myrinet" {  
    value 99;  
    description  
        "Myricom Myrinet";  
}  
enum "voiceEM" {  
    value 100;  
    description
```

```
        "voice recEive and transMit";
    }
    enum "voiceFXO" {
        value 101;
        description
            "voice Foreign Exchange Office";
    }
    enum "voiceFXS" {
        value 102;
        description
            "voice Foreign Exchange Station";
    }
    enum "voiceEncap" {
        value 103;
        description
            "voice encapsulation";
    }
    enum "voiceOverIp" {
        value 104;
        description
            "voice over IP encapsulation";
    }
    enum "atmDxi" {
        value 105;
        description
            "ATM DXI";
    }
    enum "atmFuni" {
        value 106;
        description
            "ATM FUNI";
    }
    enum "atmIma" {
        value 107;
        description
            "ATM IMA";
    }
    enum "pppMultilinkBundle" {
        value 108;
        description
            "PPP Multilink Bundle";
    }
    enum "ipOverCdlc" {
        value 109;
        description
            "IBM ipOverCdlc";
    }
    enum "ipOverClaw" {
```

```
    value 110;
    description
        "IBM Common Link Access to Workstn";
}
enum "stackToStack" {
    value 111;
    description
        "IBM stackToStack";
}
enum "virtualIpAddress" {
    value 112;
    description
        "IBM VIPA";
}
enum "mpc" {
    value 113;
    description
        "IBM multi-protocol channel support";
}
enum "ipOverAtm" {
    value 114;
    description
        "IBM ipOverAtm";
    reference
        "RFC 2320 - Definitions of Managed Objects for Classical IP
        and ARP Over ATM Using SMIPv2 (IPOA-MIB)";
}
enum "iso88025Fiber" {
    value 115;
    description
        "ISO 802.5j Fiber Token Ring";
}
enum "tdlc" {
    value 116;
    description
        "IBM twinaxial data link control";
}
enum "gigabitEthernet" {
    value 117;
    status deprecated;
    description
        "Obsoleted via RFC3635
        ethernetCsmacd(6) should be used instead";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
}
enum "hdlc" {
```

```
        value 118;
        description
            "HDLC";
    }
    enum "lapf" {
        value 119;
        description
            "LAP F";
    }
    enum "v37" {
        value 120;
        description
            "V.37";
    }
    enum "x25mlp" {
        value 121;
        description
            "Multi-Link Protocol";
    }
    enum "x25huntGroup" {
        value 122;
        description
            "X25 Hunt Group";
    }
    enum "transpHdlc" {
        value 123;
        description
            "Transp HDLC";
    }
    enum "interleave" {
        value 124;
        description
            "Interleave channel";
    }
    enum "fast" {
        value 125;
        description
            "Fast channel";
    }
    enum "ip" {
        value 126;
        description
            "IP (for APPN HPR in IP networks)";
    }
    enum "docsCableMacLayer" {
        value 127;
        description
            "CATV Mac Layer";
    }
```

```
}
enum "docsCableDownstream" {
    value 128;
    description
        "CATV Downstream interface";
}
enum "docsCableUpstream" {
    value 129;
    description
        "CATV Upstream interface";
}
enum "al2MppSwitch" {
    value 130;
    description
        "Avalon Parallel Processor";
}
enum "tunnel" {
    value 131;
    description
        "Encapsulation interface";
}
enum "coffee" {
    value 132;
    description
        "coffee pot";
    reference
        "RFC 2325 - Coffee MIB";
}
enum "ces" {
    value 133;
    description
        "Circuit Emulation Service";
}
enum "atmSubInterface" {
    value 134;
    description
        "ATM Sub Interface";
}
enum "l2vlan" {
    value 135;
    description
        "Layer 2 Virtual LAN using 802.1Q";
}
enum "l3ipvlan" {
    value 136;
    description
        "Layer 3 Virtual LAN using IP";
}
```



```
enum "l3ipxvlan" {
    value 137;
    description
        "Layer 3 Virtual LAN using IPX";
}
enum "digitalPowerline" {
    value 138;
    description
        "IP over Power Lines";
}
enum "mediaMailOverIp" {
    value 139;
    description
        "Multimedia Mail over IP";
}
enum "dtm" {
    value 140;
    description
        "Dynamic synchronous Transfer Mode";
}
enum "dcn" {
    value 141;
    description
        "Data Communications Network";
}
enum "ipForward" {
    value 142;
    description
        "IP Forwarding Interface";
}
enum "msdsl" {
    value 143;
    description
        "Multi-rate Symmetric DSL";
}
enum "ieee1394" {
    value 144;
    description
        "IEEE1394 High Performance Serial Bus";
}
enum "if-gsn" {
    value 145;
    description
        "HIPPI-6400";
}
enum "dvbRccMacLayer" {
    value 146;
    description
```

```
        "DVB-RCC MAC Layer";
    }
    enum "dvbRccDownstream" {
        value 147;
        description
            "DVB-RCC Downstream Channel";
    }
    enum "dvbRccUpstream" {
        value 148;
        description
            "DVB-RCC Upstream Channel";
    }
    enum "atmVirtual" {
        value 149;
        description
            "ATM Virtual Interface";
    }
    enum "mplsTunnel" {
        value 150;
        description
            "MPLS Tunnel Virtual Interface";
    }
    enum "srp" {
        value 151;
        description
            "Spatial Reuse Protocol          ";
    }
    enum "voiceOverAtm" {
        value 152;
        description
            "Voice Over ATM";
    }
    enum "voiceOverFrameRelay" {
        value 153;
        description
            "Voice Over Frame Relay";
    }
    enum "ids1" {
        value 154;
        description
            "Digital Subscriber Loop over ISDN";
    }
    enum "compositeLink" {
        value 155;
        description
            "Avici Composite Link Interface";
    }
    enum "ss7SigLink" {
```

```
        value 156;
        description
            "SS7 Signaling Link";
    }
    enum "propWirelessP2P" {
        value 157;
        description
            "Prop. P2P wireless interface";
    }
    enum "frForward" {
        value 158;
        description
            "Frame Forward Interface";
    }
    enum "rfc1483" {
        value 159;
        description
            "Multiprotocol over ATM AAL5";
        reference
            "RFC 1483 - Multiprotocol Encapsulation over ATM
              Adaptation Layer 5";
    }
    enum "usb" {
        value 160;
        description
            "USB Interface";
    }
    enum "ieee8023adLag" {
        value 161;
        description
            "IEEE 802.3ad Link Aggregate";
    }
    enum "bgppolicyaccounting" {
        value 162;
        description
            "BGP Policy Accounting";
    }
    enum "frf16MfrBundle" {
        value 163;
        description
            "FRF .16 Multilink Frame Relay";
    }
    enum "h323Gatekeeper" {
        value 164;
        description
            "H323 Gatekeeper";
    }
    enum "h323Proxy" {
```

```
        value 165;
        description
            "H323 Voice and Video Proxy";
    }
    enum "mpls" {
        value 166;
        description
            "MPLS";
    }
    enum "mfSigLink" {
        value 167;
        description
            "Multi-frequency signaling link";
    }
    enum "hdsl2" {
        value 168;
        description
            "High Bit-Rate DSL - 2nd generation";
    }
    enum "shdsl" {
        value 169;
        description
            "Multirate HDSL2";
    }
    enum "dslFDL" {
        value 170;
        description
            "Facility Data Link 4Kbps on a DS1";
    }
    enum "pos" {
        value 171;
        description
            "Packet over SONET/SDH Interface";
    }
    enum "dvbAsiIn" {
        value 172;
        description
            "DVB-ASI Input";
    }
    enum "dvbAsiOut" {
        value 173;
        description
            "DVB-ASI Output";
    }
    enum "plc" {
        value 174;
        description
            "Power Line Communications";
    }
```

```
}
enum "nfas" {
  value 175;
  description
    "Non Facility Associated Signaling";
}
enum "tr008" {
  value 176;
  description
    "TR008";
}
enum "gr303RDT" {
  value 177;
  description
    "Remote Digital Terminal";
}
enum "gr303IDT" {
  value 178;
  description
    "Integrated Digital Terminal";
}
enum "isup" {
  value 179;
  description
    "ISUP";
}
enum "propDocsWirelessMaclayer" {
  value 180;
  description
    "Cisco proprietary Maclayer";
}
enum "propDocsWirelessDownstream" {
  value 181;
  description
    "Cisco proprietary Downstream";
}
enum "propDocsWirelessUpstream" {
  value 182;
  description
    "Cisco proprietary Upstream";
}
enum "hiperlan2" {
  value 183;
  description
    "HIPERLAN Type 2 Radio Interface";
}
enum "propBWAp2Mp" {
  value 184;
```

```
description
  "PropBroadbandWirelessAccesspt2multipt use of this value
   for IEEE 802.16 WMAN interfaces as per IEEE Std 802.16f
   is deprecated and ieee80216WMAN(237) should be used
   instead.";
}
enum "sonetOverheadChannel" {
  value 185;
  description
    "SONET Overhead Channel";
}
enum "digitalWrapperOverheadChannel" {
  value 186;
  description
    "Digital Wrapper";
}
enum "aal2" {
  value 187;
  description
    "ATM adaptation layer 2";
}
enum "radioMAC" {
  value 188;
  description
    "MAC layer over radio links";
}
enum "atmRadio" {
  value 189;
  description
    "ATM over radio links";
}
enum "imt" {
  value 190;
  description
    "Inter Machine Trunks";
}
enum "mvl" {
  value 191;
  description
    "Multiple Virtual Lines DSL";
}
enum "reachDSL" {
  value 192;
  description
    "Long Reach DSL";
}
enum "frDlciEndPt" {
  value 193;
```

```
        description
        "Frame Relay DLCI End Point";
    }
    enum "atmVciEndPt" {
        value 194;
        description
        "ATM VCI End Point";
    }
    enum "opticalChannel" {
        value 195;
        description
        "Optical Channel";
    }
    enum "opticalTransport" {
        value 196;
        description
        "Optical Transport";
    }
    enum "propAtm" {
        value 197;
        description
        "Proprietary ATM";
    }
    enum "voiceOverCable" {
        value 198;
        description
        "Voice Over Cable Interface";
    }
    enum "infiniband" {
        value 199;
        description
        "Infiniband";
    }
    enum "teLink" {
        value 200;
        description
        "TE Link";
    }
    enum "q2931" {
        value 201;
        description
        "Q.2931";
    }
    enum "virtualTg" {
        value 202;
        description
        "Virtual Trunk Group";
    }
}
```

```
enum "sipTg" {
    value 203;
    description
        "SIP Trunk Group";
}
enum "sipSig" {
    value 204;
    description
        "SIP Signaling";
}
enum "docsCableUpstreamChannel" {
    value 205;
    description
        "CATV Upstream Channel";
}
enum "econet" {
    value 206;
    description
        "Acorn Econet";
}
enum "pon155" {
    value 207;
    description
        "FSAN 155Mb Symmetrical PON interface";
}
enum "pon622" {
    value 208;
    description
        "FSAN622Mb Symmetrical PON interface";
}
enum "bridge" {
    value 209;
    description
        "Transparent bridge interface";
}
enum "linegroup" {
    value 210;
    description
        "Interface common to multiple lines";
}
enum "voiceEMFGD" {
    value 211;
    description
        "voice E&M Feature Group D";
}
enum "voiceFGDEANA" {
    value 212;
    description
```



```
        "voice FGD Exchange Access North American";
    }
    enum "voiceDID" {
        value 213;
        description
            "voice Direct Inward Dialing";
    }
    enum "mpegTransport" {
        value 214;
        description
            "MPEG transport interface";
    }
    enum "sixToFour" {
        value 215;
        status deprecated;
        description
            "6to4 interface (DEPRECATED)";
        reference
            "RFC 4087 - IP Tunnel MIB";
    }
    enum "gtp" {
        value 216;
        description
            "GTP (GPRS Tunneling Protocol)";
    }
    enum "pdnEtherLoop1" {
        value 217;
        description
            "Paradyne EtherLoop 1";
    }
    enum "pdnEtherLoop2" {
        value 218;
        description
            "Paradyne EtherLoop 2";
    }
    enum "opticalChannelGroup" {
        value 219;
        description
            "Optical Channel Group";
    }
    enum "homepna" {
        value 220;
        description
            "HomePNA ITU-T G.989";
    }
    enum "gfp" {
        value 221;
        description
```

```
    "Generic Framing Procedure (GFP)";
}
enum "ciscoISLvlan" {
    value 222;
    description
        "Layer 2 Virtual LAN using Cisco ISL";
}
enum "actelisMetaLOOP" {
    value 223;
    description
        "Acteleis proprietary MetaLOOP High Speed Link";
}
enum "fcipLink" {
    value 224;
    description
        "FCIP Link";
}
enum "rpr" {
    value 225;
    description
        "Resilient Packet Ring Interface Type";
}
enum "qam" {
    value 226;
    description
        "RF Qam Interface";
}
enum "lmp" {
    value 227;
    description
        "Link Management Protocol";
    reference
        "RFC 4327 - Link Management Protocol (LMP) Management
        Information Base (MIB)";
}
enum "cblVectaStar" {
    value 228;
    description
        "Cambridge Broadband Networks Limited VectaStar";
}
enum "docsCableMCmtsDownstream" {
    value 229;
    description
        "CATV Modular CMTS Downstream Interface";
}
enum "adsl2" {
    value 230;
    status deprecated;
}
```

```
description
  "Asymmetric Digital Subscriber Loop Version 2
  (DEPRECATED/OBSOLETE - please use adsl2plus(238)
  instead)";
reference
  "RFC 4706 - Definitions of Managed Objects for Asymmetric
  Digital Subscriber Line 2 (ADSL2)";
}
enum "macSecControlledIF" {
  value 231;
  description
    "MACSecControlled";
}
enum "macSecUncontrolledIF" {
  value 232;
  description
    "MACSecUncontrolled";
}
enum "aviciOpticalEther" {
  value 233;
  description
    "Avici Optical Ethernet Aggregate";
}
enum "atmbond" {
  value 234;
  description
    "atmbond";
}
enum "voiceFGDOS" {
  value 235;
  description
    "voice FGD Operator Services";
}
enum "mocaVersion1" {
  value 236;
  description
    "MultiMedia over Coax Alliance (MoCA) Interface
    as documented in information provided privately to IANA";
}
enum "ieee80216WMAN" {
  value 237;
  description
    "IEEE 802.16 WMAN interface";
}
enum "adsl2plus" {
  value 238;
  description
    "Asymmetric Digital Subscriber Loop Version 2,
```

```
        Version 2 Plus and all variants";
    }
    enum "dvbRcsMacLayer" {
        value 239;
        description
            "DVB-RCS MAC Layer";
        reference
            "RFC 5728 - The SatLabs Group DVB-RCS MIB";
    }
    enum "dvbTdm" {
        value 240;
        description
            "DVB Satellite TDM";
        reference
            "RFC 5728 - The SatLabs Group DVB-RCS MIB";
    }
    enum "dvbRcsTdma" {
        value 241;
        description
            "DVB-RCS TDMA";
        reference
            "RFC 5728 - The SatLabs Group DVB-RCS MIB";
    }
    enum "x86Laps" {
        value 242;
        description
            "LAPS based on ITU-T X.86/Y.1323";
    }
    enum "wwanPP" {
        value 243;
        description
            "3GPP WWAN";
    }
    enum "wwanPP2" {
        value 244;
        description
            "3GPP2 WWAN";
    }
    enum "voiceEBS" {
        value 245;
        description
            "voice P-phone EBS physical interface";
    }
    enum "ifPwType" {
        value 246;
        description
            "Pseudowire interface type";
        reference
```

```
    "RFC 5601 - Pseudowire (PW) Management Information Base";
}
enum "ilan" {
    value 247;
    description
        "Internal LAN on a bridge per IEEE 802.1ap";
}
enum "pip" {
    value 248;
    description
        "Provider Instance Port on a bridge per IEEE 802.1ah PBB";
}
enum "aluELP" {
    value 249;
    description
        "Alcatel-Lucent Ethernet Link Protection";
}
enum "gpon" {
    value 250;
    description
        "Gigabit-capable passive optical networks (G-PON) as per
        ITU-T G.948";
}
enum "vdsl2" {
    value 251;
    description
        "Very high speed digital subscriber line Version 2
        (as per ITU-T Recommendation G.993.2)";
    reference
        "RFC 5650 - Definitions of Managed Objects for Very High
        Speed Digital Subscriber Line 2 (VDSL2)";
}
enum "capwapDot11Profile" {
    value 252;
    description
        "WLAN Profile Interface";
    reference
        "RFC 5834 - Control and Provisioning of Wireless Access
        Points (CAPWAP) Protocol Binding MIB for
        IEEE 802.11";
}
enum "capwapDot11Bss" {
    value 253;
    description
        "WLAN BSS Interface";
    reference
        "RFC 5834 - Control and Provisioning of Wireless Access
        Points (CAPWAP) Protocol Binding MIB for
```

```
        IEEE 802.11";
    }
    enum "capwapWtpVirtualRadio" {
        value 254;
        description
            "WTP Virtual Radio Interface";
        reference
            "RFC 5833 - Control and Provisioning of Wireless Access
            Points (CAPWAP) Protocol Base MIB";
    }
    enum "bits" {
        value 255;
        description
            "bitsport";
    }
    enum "docsCableUpstreamRfPort" {
        value 256;
        description
            "DOCSIS CATV Upstream RF Port";
    }
    enum "cableDownstreamRfPort" {
        value 257;
        description
            "CATV downstream RF port";
    }
    enum "vmwareVirtualNic" {
        value 258;
        description
            "VMware Virtual Network Interface";
    }
    enum "ieee802154" {
        value 259;
        description
            "IEEE 802.15.4 WPAN interface";
        reference
            "IEEE 802.15.4-2006";
    }
    enum "otnOdu" {
        value 260;
        description
            "OTN Optical Data Unit";
    }
    enum "otnOtu" {
        value 261;
        description
            "OTN Optical channel Transport Unit";
    }
    enum "ifVfiType" {
```

```
        value 262;
        description
            "VPLS Forwarding Instance Interface Type";
    }
    enum "g9981" {
        value 263;
        description
            "G.998.1 bonded interface";
    }
    enum "g9982" {
        value 264;
        description
            "G.998.2 bonded interface";
    }
    enum "g9983" {
        value 265;
        description
            "G.998.3 bonded interface";
    }
    enum "aluEpon" {
        value 266;
        description
            "Ethernet Passive Optical Networks (E-PON)";
    }
    enum "aluEponOnu" {
        value 267;
        description
            "EPON Optical Network Unit";
    }
    enum "aluEponPhysicalUni" {
        value 268;
        description
            "EPON physical User to Network interface";
    }
    enum "aluEponLogicalLink" {
        value 269;
        description
            "The emulation of a point-to-point link over the EPON
            layer";
    }
    enum "aluGponOnu" {
        value 270;
        description
            "GPON Optical Network Unit";
        reference
            "ITU-T G.984.2";
    }
    enum "aluGponPhysicalUni" {
```

```
        value 271;
        description
            "GPON physical User to Network interface";
        reference
            "ITU-T G.984.2";
    }
    enum "vmwareNicTeam" {
        value 272;
        description
            "VMware NIC Team";
    }
}
description
    "This data type is used as the syntax of the 'type'
    leaf in the 'interface' list in the YANG module
    ietf-interface.

    The definition of this typedef with the
    addition of newly assigned values is published
    periodically by the IANA, in either the Assigned
    Numbers RFC, or some derivative of it specific to
    Internet Network Management number assignments. (The
    latest arrangements can be obtained by contacting the
    IANA.)

    Requests for new values should be made to IANA via
    email (iana@iana.org).";
reference
    "ifType definitions registry.
    <http://www.iana.org/assignments/smi-numbers>";
}
}
```

<CODE ENDS>



## 3. IANA Maintained AFN and SAFI YANG Module

```
<CODE BEGINS> file "iana-afn-safi.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

      Tel:    +1 310 823 9358
      E-Mail: iana@iana.org";
  description
    "This YANG module provides two typedefs containing YANG
    definitions for the following IANA-registered enumerations:

    - Address Family Numbers (AFN)

    - Subsequent Address Family Identifiers (SAFI)

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

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

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

    This version of this YANG module is part of RFC XXXX; see the
    RFC itself for full legal notices.";
  // RFC Ed.: replace XXXX with actual RFC number and remove this
  // note.

  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  revision 2012-06-04 {
```

```
description
  "Initial revision.";
reference
  "RFC XXXX: TITLE";
}

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
        "HDLCL (8-bit multidrop)";
    }
    enum bbn1822 {
      value "5";
      description
        "BBN 1822";
    }
    enum all802 {
      value "6";
      description
        "802 (includes all 802 media plus Ethernet 'canonical
        format')";
    }
    enum e163 {
      value "7";
      description
        "E.163";
    }
  }
}
```

```
}
enum e164 {
  value "8";
  description
    "E.164 (SMDS, FrameRelay, ATM)";
}
enum f69 {
  value "9";
  description
    "F.69 (Telex)";
}
enum x121 {
  value "10";
  description
    "X.121 (X.25, Frame Relay)";
}
enum ipx {
  value "11";
  description
    "IPX (Internetwork Packet Exchange)";
}
enum appletalk {
  value "12";
  description
    "Appletalk";
}
enum decnetIV {
  value "13";
  description
    "DECnet IV";
}
enum banyanVines {
  value "14";
  description
    "Banyan Vines";
}
enum e164withNsap {
  value "15";
  description
    "E.164 with NSAP format subaddress";
  reference
    "ATM Forum UNI 3.1";
}
enum dns {
  value "16";
  description
    "DNS (Domain Name System)";
}
```

```
enum distinguishedName {
    value "17";
    description
        "Distinguished Name (per X.500)";
}
enum asNumber {
    value "18";
    description
        "Autonomous System Number";
}
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 l2vpn {
    value "25";
    description
        "AFI for L2VPN information";
    reference
        "RFC 4761: Virtual Private LAN Service (VPLS): Using BGP
        for Auto-Discovery and Signaling";
}
```

```

        RFC 6074: Provisioning, Auto-Discovery, and Signaling in
        Layer 2 Virtual Private Networks (L2VPNs)
        ";
    }
    enum eigrpCommon {
        value "16384";
        description
            "EIGRP Common Service Family";
    }
    enum eigrpIPv4 {
        value "16385";
        description
            "EIGRP IPv4 Service Family";
    }
    enum eigrpIPv6 {
        value "16386";
        description
            "EIGRP IPv6 Service Family";
    }
    enum lcaf {
        value "16387";
        description
            "LISP Canonical Address Format";
    }
}
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>
    ";
}

typedef subsequent-address-family {
    type enumeration {
        enum nlri-unicast {
            value "1";
            description
                "Network Layer Reachability Information used for unicast
                forwarding";
            reference
                "RFC 4760: Multiprotocol Extensions for BGP-4";
        }
        enum nlri-multicast {
            value "2";
            description

```

```
        "Network Layer Reachability Information used for multicast
        forwarding";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum nlri-mpls {
    value "4";
    description
        "Network Layer Reachability Information (NLRI) with MPLS
        Labels";
    reference
        "RFC 3107: Carrying Label Information in BGP-4";
}
enum mcast-vpn {
    value "5";
    description
        "MCAST-VPN";
    reference
        "RFC 6514: BGP Encodings and Procedures for Multicast in
        MPLS/BGP IP VPNs";
}
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: Dynamic Placement of Multi
        Segment Pseudowires";
}
enum encapsulation {
    value "7";
    description
        "Encapsulation SAFI";
    reference
        "RFC 5512: The BGP Encapsulation Subsequent Address Family
        Identifier (SAFI) and the BGP Tunnel Encapsulation
        Attribute";
}
enum tunnel-safi {
    value "64";
    status "obsolete";
    description
        "Tunnel SAFI";
    reference
        "draft-nalawade-kapoor-tunnel-safi: BGP Tunnel SAFI";
}
```

```
}
enum vpls {
  value "65";
  description
    "Virtual Private LAN Service (VPLS)";
  reference
    "RFC 4761: Virtual Private LAN Service (VPLS): Using BGP
    for Auto-Discovery and Signaling

    RFC 6074: Provisioning, Auto-Discovery, and Signaling in
    Layer 2 Virtual Private Networks (L2VPNs)
    ";
}
enum bgp-mdt {
  value "66";
  description
    "BGP MDT SAFI";
  reference
    "RFC 6037: Cisco Systems' Solution for Multicast in
    BGP/MPLS IP VPNs";
}
enum bgp-4over6 {
  value "67";
  description
    "BGP 4over6 SAFI";
  reference
    "RFC 5747: 4over6 Transit Solution Using IP Encapsulation
    and MP-BGP Extensions";
}
enum bgp-6over4 {
  value "68";
  description
    "BGP 6over4 SAFI";
}
enum llvpn-auto-discovery {
  value "69";
  description
    "Layer-1 VPN auto-discovery information";
  reference
    "RFC 5195: BGP-Based Auto-Discovery for Layer-1 VPNs";
}
enum mpls-vpn {
  value "128";
  description
    "MPLS-labeled VPN address";
  reference
    "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)";
}
```

```
enum multicast-bgp-mpls-vpn {
  value "129";
  description
    "Multicast for BGP/MPLS IP Virtual Private Networks
    (VPNs)";
  reference
    "RFC 6513: Multicast in MPLS/BGP IP VPNs

    RFC 6514: BGP Encodings and Procedures for Multicast in
    MPLS/BGP IP VPNs
    ";
}
enum route-target-constraints {
  value "132";
  description
    "Route Target constraints";
  reference
    "RFC 4684: Constrained Route Distribution for Border
    Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS)
    Internet Protocol (IP) Virtual Private Networks (VPNs)";
}
enum ipv4-diss-flow {
  value "133";
  description
    "IPv4 dissemination of flow specification rules";
  reference
    "RFC 5575: Dissemination of Flow Specification Rules";
}
enum vpnv4-diss-flow {
  value "134";
  description
    "IPv4 dissemination of flow specification rules";
  reference
    "RFC 5575: Dissemination of Flow Specification Rules";
}
enum vpn-auto-discovery {
  value "140";
  status "obsolete";
  description
    "VPN auto-discovery";
  reference
    "draft-ietf-l3vpn-bgpvpn-auto: Using BGP as an
    Auto-Discovery Mechanism for VR-based Layer-3 VPNs";
}
}
description
  "This typedef is a YANG enumeration of IANA-registered
  subsequent address family identifiers (SAFI).";
```



```
reference
  "Subsequent Address Family Identifiers (SAFI) Parameters. IANA,
    2012-02-22. <http://www.iana.org/assignments/safi-namespace/
    safi-namespace.xml>
    ";
  }
}
```

<CODE ENDS>

#### 4. IANA Considerations

This document defines the initial version of the IANA-maintained `iana-if-type` and `iana-afn-safi` YANG modules.

The `iana-if-type` module is intended to reflect the "ifType definitions" registry. When an interface type is added to this registry, a new "enum" statement must be added to the "iana-if-type" typedef, with the same name and value as the corresponding enumeration in IANAifType-MIB. If the new interface type has a reference, a new "reference" statement should be added to the new "enum" statement. If an interface type is deprecated in the "ifType definitions" registry, the corresponding "enum" statement must be updated with a "status" statement with the value "deprecated".

When the `iana-if-type` YANG module is updated, a new "revision" statement must be added.

The `iana-afn-safi` module is intended to reflect the "Address Family Numbers" and "Subsequent Address Family Identifiers" registries. When an AFN or SAFI is added to these registries, a new "enum" statement must be added to the "address-family" or "subsequent-address-family" typedefs. If the new parameter has a reference, a new "reference" statement should be added to the new "enum" statement. If a parameter gets deprecated in the registry, the corresponding "enum" statement must be updated with a "status" statement with the value "deprecated".

When the `iana-afn-safi` YANG module is updated, a new "revision" statement must be added.

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registrations are requested to be made.

URI: urn:ietf:params:xml:ns:yang:iana-if-types

Registrant Contact: IANA.

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 two YANG modules in the YANG Module Names registry [RFC6020].

```
name:      iana-if-type
namespace: urn:ietf:params:xml:ns:yang:iana-if-type
prefix:    ianaift
reference:  RFC XXXX

name:      iana-afn-safi
namespace: urn:ietf:params:xml:ns:yang:iana-afn-safi
prefix:    ianaaf
reference:  RFC XXXX
```

## 5. Security Considerations

Since this document does not introduce any technology or protocol, there are no security issues to be considered for this document itself.

## 6. Normative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Author's Address

Martin Bjorklund  
Tail-f Systems

Email: [mbj@tail-f.com](mailto:mbj@tail-f.com)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 15, 2013

M. Bjorklund  
Tail-f Systems  
July 14, 2012

A YANG Data Model for Interface Configuration  
draft-ietf-netmod-interfaces-cfg-05

Abstract

This document defines a YANG data model for the configuration of network interfaces. It is expected that interface type specific configuration data models augment the generic interfaces data model defined in this document.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 15, 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.



## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
2. Objectives . . . . .	4
3. Interfaces Data Model . . . . .	5
3.1. The interface List . . . . .	5
3.2. Interface References . . . . .	6
3.3. Interface Layering . . . . .	6
4. Relationship to the IF-MIB . . . . .	8
5. Interfaces YANG Module . . . . .	9
6. IANA Considerations . . . . .	14
7. Security Considerations . . . . .	15
8. Acknowledgments . . . . .	16
9. References . . . . .	17
9.1. Normative References . . . . .	17
9.2. Informative References . . . . .	17
Appendix A. Example: Ethernet Interface Module . . . . .	18
Appendix B. Example: Ethernet Bonding Interface Module . . . . .	20
Appendix C. Example: VLAN Interface Module . . . . .	21
Appendix D. Example: NETCONF <get> reply . . . . .	22
Appendix E. ChangeLog . . . . .	23
E.1. Version -05 . . . . .	23
E.2. Version -04 . . . . .	23
E.3. Version -03 . . . . .	23
E.4. Version -02 . . . . .	23
E.5. Version -01 . . . . .	23
Author's Address . . . . .	24

## 1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of network interfaces. It is expected that interface type specific configuration data models augment the generic interfaces data model defined in this document.

Network interfaces are central to the configuration of many Internet protocols. Thus, it is important to establish a common data model for how interfaces are identified and configured.

### 1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

The following terms are defined in [RFC6241] and are not redefined here:

- o client
- o server

The following terms are defined in [RFC6020] and are not redefined here:

- o augment
- o data model
- o data node

## 2. Objectives

This section describes some of the design objectives for the model presented in Section 5.

- o It is recognized that existing implementations will have to map the interface data model defined in this memo to their proprietary native data model. The new data model should be simple to facilitate such mappings.
- o The data model should be suitable for new implementations to use as-is, without requiring a mapping to a different native model.
- o References to interfaces should be as simple as possible, preferably by using a single leafref.
- o The mapping to ifIndex [RFC2863] used by SNMP to identify interfaces must be clear.
- o The model must support interface layering, both simple layering where one interface is layered on top of exactly one other interface, and more complex scenarios where one interface is aggregated over N other interfaces, or when N interfaces are multiplexed over one other interface.
- o The data model should support the pre-provisioning of interface configuration, i.e., it should be possible to configure an interface whose physical interface hardware is not present on the device. It is recommended that devices that support dynamic addition and removal of physical interfaces also support pre-provisioning.

### 3. Interfaces Data Model

The data model in the module "ietf-interfaces" has the following structure, where square brackets are used to enclose a list's keys, and "?" means that the leaf is optional:

```

+--rw interfaces
  +--rw interface [name]
    +--rw name                string
    +--rw description?        string
    +--rw type                 ianaift:iana-if-type
    +--rw location?           string
    +--rw enabled?            boolean
    +--ro if-index             int32
    +--rw mtu?                uint32
    +--rw link-up-down-trap-enable? enumeration
```

This module defines one YANG feature:

snmp-if-mib: Indicates that the server implements IF-MIB [RFC2863].

#### 3.1. The interface List

The data model for interface configuration presented in this document uses a flat list of interfaces. Each interface in the list is identified by its name. Furthermore, each interface has a mandatory "type" leaf, and a "location" leaf. The combination of "type" and "location" is unique within the interface list.

It is expected that interface type specific data models augment the interface list, and use the "type" leaf to make the augmentation conditional.

As an example of such an interface type specific augmentation, consider this YANG snippet. For a more complete example, see Appendix A.

```
import interfaces {
  prefix "if";
}

augment "/if:interfaces/if:interface" {
  when "if:type = 'ethernetCsmacd'";

  container ethernet {
    leaf duplex {
      ...
    }
  }
}
```

The "location" leaf is a string. It is optional in the data model, but if the type represents a physical interface, it is mandatory. The format of this string is device- and type-dependent. The device uses the location string to identify the physical or logical entity that the configuration applies to. For example, if a device has a single array of 8 ethernet ports, the location can be one of the strings "1" to "8". As another example, if a device has N cards of M ports, the location can be on the form "n/m", such as "1/0".

How a client can learn which types and locations are present on a certain device is outside the scope of this document.

### 3.2. Interface References

An interface is identified by its name, which is unique within the server. This property is captured in the "interface-ref" typedef, which other YANG modules SHOULD use when they need to reference an existing interface.

### 3.3. Interface Layering

There is no generic mechanism for how an interface is configured to be layered on top of some other interface. It is expected that interface type specific models define their own data nodes for interface layering, by using "interface-ref" types to reference lower layers.

Below is an example of a model with such nodes. For a more complete example, see Appendix B.

```
augment "/if:interfaces/if:interface" {
  when "if:type = 'ieee8023adLag'";

  leaf-list slave-if {
    type if:interface-ref;
    must "/if:interfaces/if:interface[if:name = current()]"
      + "/if:type = 'ethernetCsmacd'" {
      description
        "The type of a slave interface must be ethernet";
    }
  }
  // other bonding config params, failover times etc.
}
```

#### 4. Relationship to the IF-MIB

If the device implements IF-MIB [RFC2863], each entry in the "interface" list is typically mapped to one ifEntry. The "if-index" leaf contains the value of the corresponding ifEntry's ifIndex.

In most cases, the "name" of an "interface" entry is mapped to ifName. ifName is defined as an DisplayString [RFC2579] which uses a 7-bit ASCII character set. An implementation MAY restrict the allowed values for "name" to match the restrictions of ifName.

The IF-MIB allows two different ifEntries to have the same ifName. Devices that support this feature, and also support the configuration of these interfaces using the "interface" list, cannot have a 1-1 mapping between the "name" leaf and ifName.

The IF-MIB also defines the writable object ifPromiscuousMode. Since this object typically is not a configuration object, it is not mapped to the "ietf-interfaces" module.

The following table lists the YANG data nodes with corresponding objects in the IF-MIB.

YANG data node	IF-MIB object
interface	ifEntry
name	ifName
description	ifAlias
type	ifType
enabled	ifAdminStatus
if-index	ifIndex
mtu	ifMtu
link-up-down-trap-enable	ifLinkUpDownTrapEnable

Mapping of YANG data nodes to IF-MIB objects

## 5. Interfaces YANG Module

This YANG module imports a typedef from [I-D.ietf-netmod-iana-if-type].

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-interfaces@2012-07-14.yang"

```
module ietf-interfaces {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";  
    prefix if;  
  
    import iana-if-type {  
        prefix ianaift;  
    }  
  
    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:    Martin Bjorklund  
                  <mailto:mbj@tail-f.com>";  
  
    description  
        "This module contains a collection of YANG definitions for  
        configuring network interfaces.  
  
        Copyright (c) 2012 IETF Trust and the persons identified as  
        authors of the code. All rights reserved.  
  
        Redistribution and use in source and binary forms, with or  
        without modification, is permitted pursuant to, and subject  
        to the license terms contained in, the Simplified BSD License  
        set forth in Section 4.c of the IETF Trust's Legal Provisions  
        Relating to IETF Documents  
        (http://trustee.ietf.org/license-info)."
```



```

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

// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
revision 2012-07-14 {
    description
        "Initial revision.";
    reference
        "RFC XXXX: A YANG Data Model for Interface Configuration";
}

/* Typedefs */

typedef interface-ref {
    type leafref {
        path "/if:interfaces/if:interface/if:name";
    }
    description
        "This type is used by data models that need to reference
        interfaces.";
}

/* Features */

feature snmp-if-mib {
    description
        "This feature indicates that the server implements IF-MIB.";
    reference
        "RFC 2863: The Interfaces Group MIB";
}

/* Data nodes */

container interfaces {
    description
        "Interface parameters.";

    list interface {
        key "name";
        unique "type location";

        description
            "The list of configured interfaces on the device.";
    }
}
```

```
leaf name {
  type string;
  description
    "An arbitrary name for the interface.

    A device MAY restrict the allowed values for this leaf,
    possibly depending on the type and location.

    For example, if a device has a single array of 8 ethernet
    ports, the name might be restricted to be on the form
    'ethN', where N is an integer between '1' and '8'.

    This leaf MAY be mapped to ifName by an implementation.
    Such an implementation MAY restrict the allowed values for
    this leaf so that it matches the restrictions of ifName.
    If a NETCONF server that implements this restriction is
    sent a value that doesn't match the restriction, it MUST
    reply with an rpc-error with the error-tag
    'invalid-value'.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifName";
}

leaf description {
  type string;
  description
    "A textual description of the interface.

    This leaf MAY be mapped to ifAlias by an implementation.
    Such an implementation MAY restrict the allowed values for
    this leaf so that it matches the restrictions of ifAlias.
    If a NETCONF server that implements this restriction is
    sent a value that doesn't match the restriction, it MUST
    reply with an rpc-error with the error-tag
    'invalid-value'.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifAlias";
}

leaf type {
  type ianaift:iana-if-type;
  mandatory true;
  description
    "The type of the interface.

    When an interface entry is created, a server MAY
    initialize the type leaf with a valid value, e.g., if it
    is possible to derive the type from the name of the
```

```
        interface.";
    }

    leaf location {
        type string;
        description
            "The device-specific location of the interface of a
            particular type.  The format of the location string
            depends on the interface type and the device.

            If the interface's type represents a physical interface,
            this leaf MUST be set.

            For example, if a device has a single array of 8 ethernet
            ports, the location can be one of '1' to '8'.  As another
            example, if a device has N cards of M ports, the location
            can be on the form 'n/m'.

            When an interface entry is created, a server MAY
            initialize the location leaf with a valid value, e.g., if
            it is possible to derive the location from the name of
            the interface.";
    }

    leaf enabled {
        type boolean;
        default "true";
        description
            "The desired state of the interface.

            This leaf contains the configured, desired state of the
            interface.  Systems that implement the IF-MIB use the
            value of this leaf to set IF-MIB.ifAdminStatus to 'up' or
            'down' after an ifEntry has been initialized, as described
            in RFC 2863.";
        reference
            "RFC 2863: The Interfaces Group MIB - ifAdminStatus";
    }

    leaf if-index {
        if-feature snmp-if-mib;
        type int32 {
            range "1..2147483647";
        }
        config false;
        description
            "The ifIndex value for the ifEntry represented by this
            interface.
```

```
        Media-specific modules must specify how the type is
        mapped to entries in the ifTable.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifIndex";
}

leaf mtu {
    type uint32;
    description
        "The size, in octets, of the largest packet that the
        interface can send and receive. This node might not be
        valid for all interface types.

        Media-specific modules must specify any restrictions on
        the mtu for their interface type.";
}

leaf link-up-down-trap-enable {
    if-feature snmp-if-mib;
    type enumeration {
        enum enabled {
            value 1;
        }
        enum disabled {
            value 2;
        }
    }
    description
        "Indicates whether linkUp/linkDown SNMP notifications
        should be generated for this interface.

        If this node is not configured, the value 'enabled' is
        operationally used by the server for interfaces which do
        not operate on top of any other interface (as defined in
        the ifStackTable), and 'disabled' otherwise.";
    reference
        "RFC 2863: The Interfaces Group MIB -
        ifLinkUpDownTrapEnable";
}
}
}

<CODE ENDS>
```

## 6. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-interfaces

Registrant Contact: The IESG.

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

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name:	ietf-interfaces
namespace:	urn:ietf:params:xml:ns:yang:ietf-interfaces
prefix:	if
reference:	RFC XXXX

## 7. Security Considerations

The YANG module defined in this memo is 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].

There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., <edit-config>) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/interfaces/interface: This list specifies the configured interfaces on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

/interfaces/interface/enabled: This leaf controls if an interface is enabled or not. Unauthorized access to this leaf could cause the device to ignore packets it should receive and process.

/interfaces/interface/mtu: Setting this leaf to a very small value can be used to slow down interfaces.

## 8. Acknowledgments

The author wishes to thank Alexander Clemm, Per Hedeland, Ladislav Lhotka, and Juergen Schoenwaelder for their helpful comments.

## 9. References

### 9.1. Normative References

- [I-D.ietf-netmod-iana-if-type]  
Bjorklund, M., "IANA Interface Type and Address Family YANG Modules", draft-ietf-netmod-iana-if-type-02 (work in progress), April 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

### 9.2. Informative References

- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIV2", STD 58, RFC 2579, April 1999.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.



#### Appendix A. Example: Ethernet Interface Module

This section gives a simple example of how an Ethernet interface module could be defined. It demonstrates how media-specific configuration parameters can be conditionally augmented to the generic interface list. It is not intended as a complete module for ethernet configuration.

```
module ex-ethernet {
  namespace "http://example.com/ethernet";
  prefix "eth";

  import ietf-interfaces {
    prefix if;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ethernetCsmacd'";

    container ethernet {
      must "../if:location" {
        description
          "An ethernet interface must specify the physical location
           of the ethernet hardware.";
      }
      choice transmission-params {
        case auto {
          leaf auto-negotiate {
            type empty;
          }
        }
        case manual {
          leaf duplex {
            type enumeration {
              enum "half";
              enum "full";
            }
          }
          leaf speed {
            type enumeration {
              enum "10Mb";
              enum "100Mb";
              enum "1Gb";
              enum "10Gb";
            }
          }
        }
      }
    }
  }
  // other ethernet specific params...
}
```

## Appendix B. Example: Ethernet Bonding Interface Module

This section gives an example of how interface layering can be defined. An ethernet bonding interface is defined, which bonds several ethernet interfaces into one logical interface.

```
module ex-ethernet-bonding {
  namespace "http://example.com/ethernet-bonding";
  prefix "bond";

  import ietf-interfaces {
    prefix if;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ieee8023adLag'";

    leaf-list slave-if {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]"
        + "/if:type = 'ethernetCsmacd'" {
        description
          "The type of a slave interface must be ethernet.";
      }
    }
    leaf bonding-mode {
      type enumeration {
        enum round-robin;
        enum active-backup;
        enum broadcast;
      }
    }
    // other bonding config params, failover times etc.
  }
}
```

## Appendix C. Example: VLAN Interface Module

This section gives an example of how a vlan interface module can be defined.

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

  import ietf-interfaces {
    prefix if;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ethernetCsmacd' or
         if:type = 'ieee8023adLag'";
    leaf vlan-tagging {
      type boolean;
      default false;
    }
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'l2vlan'";

    leaf base-interface {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]"
        + "/vlan:vlan-tagging = true" {
        description
          "The base interface must have vlan tagging enabled.";
      }
    }
    leaf vlan-id {
      type uint16 {
        range "1..4094";
      }
      must "../base-interface";
    }
  }
}
```

## Appendix D. Example: NETCONF &lt;get&gt; reply

This section gives an example of a reply to the NETCONF <get> request for a device that implements the example data models above.

```
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="101">
  <data>
    <interfaces
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type>ethernetCsmacd</type>
        <location>0</location>
        <enabled>true</enabled>
        <if-index>2</if-index>
      </interface>
      <interface>
        <name>eth1</name>
        <type>ethernetCsmacd</type>
        <location>1</location>
        <enabled>true</enabled>
        <if-index>7</if-index>
        <vlan-tagging
          xmlns="http://example.com/vlan">true</vlan-tagging>
        </interface>
      </interfaces>
    </data>
  </rpc-reply>
```

## Appendix E. ChangeLog

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

## E.1. Version -05

- o Added an Informative References section.
- o Updated the Security Considerations section.
- o Clarified the behavior of an NETCONF server when invalid values are received.

## E.2. Version -04

- o Clarified why ifPromiscuousMode is not part of this data model.
- o Added a table that shows the mapping between this YANG data model and IF-MIB.

## E.3. Version -03

- o Added the section Relationship to the IF-MIB.
- o Changed if-index to be a leaf instead of leaf-list.
- o Explained the notation used in the data model tree picture.

## E.4. Version -02

- o Editorial fixes

## E.5. Version -01

- o Changed leaf "if-admin-status" to leaf "enabled".
- o Added Security Considerations

Author's Address

Martin Bjorklund  
Tail-f Systems

Email: [mbj@tail-f.com](mailto:mbj@tail-f.com)





Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 17, 2013

M. Bjorklund  
Tail-f Systems  
July 16, 2012

A YANG Data Model for IP Configuration  
draft-ietf-netmod-ip-cfg-05

Abstract

This document defines a YANG data model for configuration of IP implementations.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
2. IP Data Model . . . . .	4
3. Relationship to IP-MIB . . . . .	6
4. IP configuration YANG Module . . . . .	7
5. IANA Considerations . . . . .	14
6. Security Considerations . . . . .	15
7. Acknowledgments . . . . .	16
8. References . . . . .	17
8.1. Normative References . . . . .	17
8.2. Informative References . . . . .	17
Appendix A. Example: NETCONF <get> reply . . . . .	19
Author's Address . . . . .	20

## 1. Introduction

This document defines a YANG [RFC6020] data model for configuration of IP implementations.

The initial version of this data model focuses on configuration parameters for interfaces. Future revisions of this data model might add other kinds of IP configuration parameters.

Configuration parameters to control IP routing are defined in [I-D.ietf-netmod-routing-cfg].

### 1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

The following terms are defined in [RFC6241] and are not redefined here:

- o client
- o server

The following terms are defined in [RFC6020] and are not redefined here:

- o augment
- o data model
- o data node

## 2. IP Data Model

The module "ietf-ip" augments the "interface" list defined in the "ietf-interfaces" module [I-D.ietf-netmod-interfaces-cfg] with the following data nodes, where square brackets are used to enclose a list's keys, and "?" means that the node is optional. Choice and case nodes are enclosed in parenthesis, and a case node is marked with a colon (":").

```

+--rw if:interfaces
  +--rw if:interface [name]
    ...
    +--rw ipv4
      +--rw enabled?          boolean
      +--rw ip-forwarding?    boolean
      +--rw address [ip]
        +--rw ip              inet:ipv4-address
        +--rw (subnet)?
          +--:(prefix-length)
            +--rw ip:prefix-length?  uint8
          +--:(netmask)
            +--rw ip:netmask?        inet:ipv4-address
      +--rw neighbor [ip]
        +--rw ip              inet:ipv4-address
        +--rw phys-address?    yang:phys-address
    +--rw ipv6
      +--rw enabled?          boolean
      +--rw ip-forwarding?    boolean
      +--rw address [ip]
        +--rw ip              inet:ipv6-address
        +--rw prefix-length?  uint8
      +--rw neighbor [ip]
        +--rw ip              inet:ipv6-address
        +--rw phys-address?    yang:phys-address
      +--rw dup-addr-detect-transmits?  uint32
      +--rw autoconf
        +--rw create-global-addresses?    boolean
        +--rw create-temporary-addresses?  boolean
        +--rw temporary-valid-lifetime?    uint32
        +--rw temporary-preferred-lifetime? uint32

```

The data model defines two containers, "ipv4" and "ipv6", representing the IPv4 and IPv6 address families. In each container, there is a leaf "enabled" that controls if the address family is enabled on that interface, and a leaf "ip-forwarding" that controls if ip packet forwarding for the address family is enabled on the interface. In each container, there is also a list of manually configured addresses, and a list of manually configured mappings from

ip addresses to physical addresses.

### 3. Relationship to IP-MIB

If the device implements IP-MIB [RFC4293], each entry in the "ipv4/address" and "ipv6/address" lists is mapped to one `ipAddressEntry`, where the `ipAddressIfIndex` refers to the interface where the "address" entry is configured.

The IP-MIB defines objects to control IPv6 Router Advertisement. The corresponding YANG data nodes are defined in [I-D.ietf-netmod-routing-cfg].

The entries in "ipv4/neighbor" and "ipv6/neighbor" are mapped to `ipNetToPhysicalTable`.

The object `ipAddressStatus` is writable in the IP-MIB but does not represent configuration, and is thus not mapped to the YANG module.

The following table lists the YANG data nodes with corresponding objects in the IP-MIB.

YANG data node	IP-MIB object
ipv4/enabled	ipv4InterfaceEnableStatus
ipv4/address	ipAddressEntry
ipv4/address/ip	ipAddressAddrType / ipAddressAddr
ipv4/neighbor	ipNetToPhysicalTable
ipv6/enabled	ipv6InterfaceEnableStatus
ipv6/ip-forwarding	ipv6InterfaceForwarding
ipv6/address	ipAddressEntry
ipv6/address/ip	ipAddressAddrType / ipAddressAddr
ipv6/neighbor	ipNetToPhysicalTable

Mapping of YANG data nodes to IP-MIB objects

#### 4. IP configuration YANG Module

This module imports typedefs from [RFC6021] and [I-D.ietf-netmod-interfaces-cfg], and references [RFC0826], [RFC4861] and [RFC4862].

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-ip@2012-07-16.yang"

```
module ietf-ip {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-ip";  
    prefix ip;  
  
    import ietf-interfaces {  
        prefix if;  
    }  
    import ietf-inet-types {  
        prefix inet;  
    }  
    import ietf-yang-types {  
        prefix yang;  
    }  
  
    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:   Martin Bjorklund  
                  <mailto:mbj@tail-f.com>";  
  
    description  
        "This module contains a collection of YANG definitions for  
        configuring IP implementations.  
  
        Copyright (c) 2012 IETF Trust and the persons identified as  
        authors of the code. All rights reserved."
```

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

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

```
// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
revision 2012-07-16 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for IP Configuration";
}

/* Features */

feature non-contiguous-netmasks {
  description
    "Indicates support for configuring non-contiguous
    subnet masks.";
}

/* Data nodes */

augment "/if:interfaces/if:interface" {
  description
    "Parameters for configuring IP on interfaces.

    If an interface is not capable of running IP, the server
    must not allow the client to configure these parameters.";

  container ipv4 {
    description
      "Parameters for the IPv4 address family.";
    leaf enabled {
      type boolean;
      default true;
    }
    description
      "Controls if IPv4 is enabled or disabled on this
      interface.";
  }
}
```



```
}
leaf ip-forwarding {
  type boolean;
  default false;
  description
    "Controls if IPv4 packet forwarding is enabled or disabled
    on this interface.";
}
list address {
  key "ip";
  description
    "The list of manually configured IPv4 addresses
    on the interface.";

  leaf ip {
    type inet:ipv4-address;
    description
      "The IPv4 address on the interface.";
  }
  choice subnet {
    default prefix-length;
    description
      "The subnet can be specified as a prefix-length, or,
      if the server supports non-contiguous netmasks, as
      a netmask.

      The default subnet is a prefix-length of 32.";
    leaf prefix-length {
      type uint8 {
        range "0..32";
      }
      default 32;
      description
        "The length of the subnet prefix.";
    }
    leaf netmask {
      if-feature non-contiguous-netmasks;
      type inet:ipv4-address;
      description
        "The subnet specified as a netmask.";
    }
  }
}
list neighbor {
  key "ip";
  description
    "A list of manually configured mappings from IPv4
    addresses to physical addresses.
```

```
        Entries in this list are used as static entries in the
        ARP cache.";
reference
    "RFC 826: An Ethernet Address Resolution Protocol";

    leaf ip {
        type inet:ipv4-address;
        description
            "The IPv4 address of a neighbor node.";
    }
    leaf phys-address {
        type yang:phys-address;
        description
            "The physical level address of the neihgbor node.";
    }
}

container ipv6 {
    description
        "Parameters for the IPv6 address family.";

    leaf enabled {
        type boolean;
        default true;
        description
            "Controls if IPv6 is enabled or disabled on this
            interface.";
    }
    leaf ip-forwarding {
        type boolean;
        default false;
        description
            "Controls if IPv6 packet forwarding is enabled or disabled
            on this interface.";
        reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
            Section 6.2.1, IsRouter";
    }
    list address {
        key "ip";
        description
            "The list of manually configured IPv6 addresses
            on the interface.";

        leaf ip {
            type inet:ipv6-address;
            description
```

```
        "The IPv6 address on the interface.";
    }
    leaf prefix-length {
        type uint8 {
            range "0..128";
        }
        default 128;
        description
            "The length of the subnet prefix.";
    }
}
list neighbor {
    key "ip";
    description
        "A list of manually configured mappings from IPv6
        addresses to physical addresses.

        Entries in this list are used as static entries in the
        Neighbor Cache.";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)";

    leaf ip {
        type inet:ipv6-address;
        description
            "The IPv6 address of a neighbor node.";
    }
    leaf phys-address {
        type yang:phys-address;
        description
            "The physical level address of the neihgbor node.";
    }
}
leaf dup-addr-detect-transmits {
    type uint32;
    default 1;
    description
        "The number of consecutive Neighbor Solicitation messages
        sent while performing Duplicate Address Detection on a
        tentative address. A value of zero indicates that
        Duplicate Address Detection is not performed on
        tentative addresses. A value of one indicates a single
        transmission with no follow-up retransmissions.";
    reference
        "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}
container autoconf {
    description
```

```
        "Parameters to control the autoconfiguration of IPv6
        addresses, as described in RFC 4862.";
reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";

leaf create-global-addresses {
    type boolean;
    default true;
    description
        "If enabled, the host creates global addresses as
        described in section 5.5 of RFC 4862.";
    reference
        "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}
leaf create-temporary-addresses {
    type boolean;
    default false;
    description
        "If enabled, the host creates temporary addresses as
        described in RFC 4941.";
    reference
        "RFC 4941: Privacy Extensions for Stateless Address
        Autoconfiguration in IPv6";
}
leaf temporary-valid-lifetime {
    type uint32;
    units "seconds";
    default 604800;
    description
        "The time the temporary address is valid.";
    reference
        "RFC 4941: Privacy Extensions for Stateless Address
        Autoconfiguration in IPv6
        - TEMP_VALID_LIFETIME";
}
leaf temporary-preferred-lifetime {
    type uint32;
    units "seconds";
    default 86400;
    description
        "The time the temporary address is preferred.";
    reference
        "RFC 4941: Privacy Extensions for Stateless Address
        Autoconfiguration in IPv6
        - TEMP_PREFERED_LIFETIME";
}
}
```

```
}  
}
```

```
<CODE ENDS>
```

## 5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:ietf-ip

Registrant Contact: The NETMOD WG of the IETF.

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

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name:	ietf-ip
namespace:	urn:ietf:params:xml:ns:yang:ietf-ip
prefix:	ip
reference:	RFC XXXX

## 6. Security Considerations

The YANG module defined in this memo is 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].

There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

ipv4/enabled and ipv6/enabled: These leafs are used to enable or disable IPv4 and IPv6 on a specific interface. By enabling a protocol on an interface, an attacker might be able to create an unsecured path into a node (or through it if routing is also enabled). By disabling a protocol on an interface, an attacker might be able to force packets to be routed through some other interface or deny access to some or all of the network via that protocol.

ipv4/address and ipv6/address: These lists specify the configured IP addresses on an interface. By modifying this information, an attacker can cause a node to either ignore messages destined to it or accept (at least at the IP layer) messages it would otherwise ignore. The use of filtering or security associations may reduce the potential damage in the latter case.

ipv4/ip-forwarding and ipv6/ip-forwarding: These leafs allow a client to enable or disable the routing functions on the entity. By disabling the routing functions, an attacker would possibly be able to deny service to users. By enabling the routing functions, an attacker could open a conduit into an area. This might result in the area providing transit for packets it shouldn't or might allow the attacker access to the area bypassing security safeguards. =ipv6/autoconf: The leafs in this branch control the autoconfiguration of IPv6 addresses and in particular whether temporary addresses are used or not. By modifying the corresponding leafs, an attacker might impact the addresses used by a node and thus indirectly the privacy of the users using the node.

## 7. Acknowledgments

The author wishes to thank Ladislav Lhotka, Juergen Schoenwaelder, and Dave Thaler for their helpful comments.



## 8. References

### 8.1. Normative References

- [I-D.ietf-netmod-interfaces-cfg]  
Bjorklund, M., "A YANG Data Model for Interface Configuration", draft-ietf-netmod-interfaces-cfg-05 (work in progress), July 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

### 8.2. Informative References

- [I-D.ietf-netmod-routing-cfg]  
Lhotka, L., "A YANG Data Model for Routing Configuration", draft-ietf-netmod-routing-cfg-04 (work in progress), July 2012.
- [RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.
- [RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

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

## Appendix A. Example: NETCONF &lt;get&gt; reply

This section gives an example of a reply to the NETCONF <get> request for a device that implements the data model defined in this document.

```
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="101">
  <data>
    <interfaces
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type>ethernetCsmacd</type>
        <location>0</location>
        <if-index>2</if-index>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>2001:DB8::1</ip>
            <prefix-length>32</prefix-length>
          </address>
          <dup-addr-detect-transmits>0</dup-addr-detect-transmits>
        </ipv6>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

Author's Address

Martin Bjorklund  
Tail-f Systems

Email: [mbj@tail-f.com](mailto:mbj@tail-f.com)



NETMOD  
Internet-Draft  
Intended status: Standards Track  
Expires: January 10, 2013

L. Lhotka  
CZ.NIC  
July 9, 2012

A YANG Data Model for Routing Configuration  
draft-ietf-netmod-routing-cfg-04

Abstract

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

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 10, 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology and Notation . . . . .	4
2.1. Glossary of New Terms . . . . .	4
2.2. Prefixes in Data Node Names . . . . .	5
3. Objectives . . . . .	6
4. The Design of the Core Routing Data Model . . . . .	7
4.1. Router . . . . .	10
4.1.1. Configuration of IPv6 Router Interfaces . . . . .	10
4.2. Route . . . . .	11
4.3. Routing Tables . . . . .	12
4.4. Routing Protocols . . . . .	13
4.4.1. Routing Pseudo-Protocols . . . . .	14
4.4.2. Defining New Routing Protocols . . . . .	14
4.5. Route Filters . . . . .	17
4.6. RPC Operations . . . . .	18
4.6.1. Operation "active-route" . . . . .	18
4.6.2. Operation "route-count" . . . . .	19
5. Interactions with Other YANG Modules . . . . .	20
5.1. Module "ietf-interfaces" . . . . .	20
5.2. Module "ietf-ip" . . . . .	20
6. Routing YANG Module . . . . .	22
7. IPv4 Unicast Routing YANG Module . . . . .	34
8. IPv6 Unicast Routing YANG Module . . . . .	38
9. IANA Considerations . . . . .	47
10. Security Considerations . . . . .	49
11. Acknowledgments . . . . .	50
12. References . . . . .	51
12.1. Normative References . . . . .	51
12.2. Informative References . . . . .	51
Appendix A. Example: Adding a New Routing Protocol . . . . .	52
Appendix B. Example: Reply to the NETCONF <get> Message . . . . .	55
Appendix C. Change Log . . . . .	60
C.1. Changes Between Versions -03 and -04 . . . . .	60
C.2. Changes Between Versions -02 and -03 . . . . .	60
C.3. Changes Between Versions -01 and -02 . . . . .	61
C.4. Changes Between Versions -00 and -01 . . . . .	61
Author's Address . . . . .	62

## 1. Introduction

This document contains a specification of the following YANG modules:

- o Module "ietf-routing" provides generic components of a routing data model.
- o Module "ietf-ipv4-unicast-routing" augments the "ietf-routing" module with additional data specific to IPv4 unicast.
- o Module "ietf-ipv6-unicast-routing" augments the "ietf-routing" module with additional data specific to IPv6 unicast, including the router configuration variables required by [RFC4861].

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



## 2. Terminology and Notation

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

The following terms are defined in [RFC6241]:

- o client
- o message
- o protocol operation
- o server

The following terms are defined in [RFC6020]:

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

### 2.1. Glossary of New Terms

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

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

**direct route:** a route to a directly connected network.

## 2.2. Prefixes in Data Node Names

In this document, names of data nodes, RPC methods and other data model objects are used mostly without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
ianaaf	iana-afn-safi	[IANA-IF-AF]
if	ietf-interfaces	[YANG-IF]
ip	ietf-ip	[YANG-IP]
rip	example-rip	Appendix A
rt	ietf-routing	Section 6
v4ur	ietf-ipv4-unicast-routing	Section 7
v6ur	ietf-ipv6-unicast-routing	Section 8
yang	ietf-yang-types	[RFC6021]
inet	ietf-inet-types	[RFC6021]

Table 1: Prefixes and corresponding YANG modules

### 3. Objectives

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

- o 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).
- o Simple routing setups, such as static routing, should be configurable in a simple way, ideally without any need to develop additional YANG modules.
- o 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.
- o Device vendors will want to map the data models built on this generic framework to their proprietary data models and configuration interfaces. Therefore, the framework should be flexible enough to facilitate such a mapping and accommodate data models with different logic.

#### 4. The Design of the Core Routing Data Model

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

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

```

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

```

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

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

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

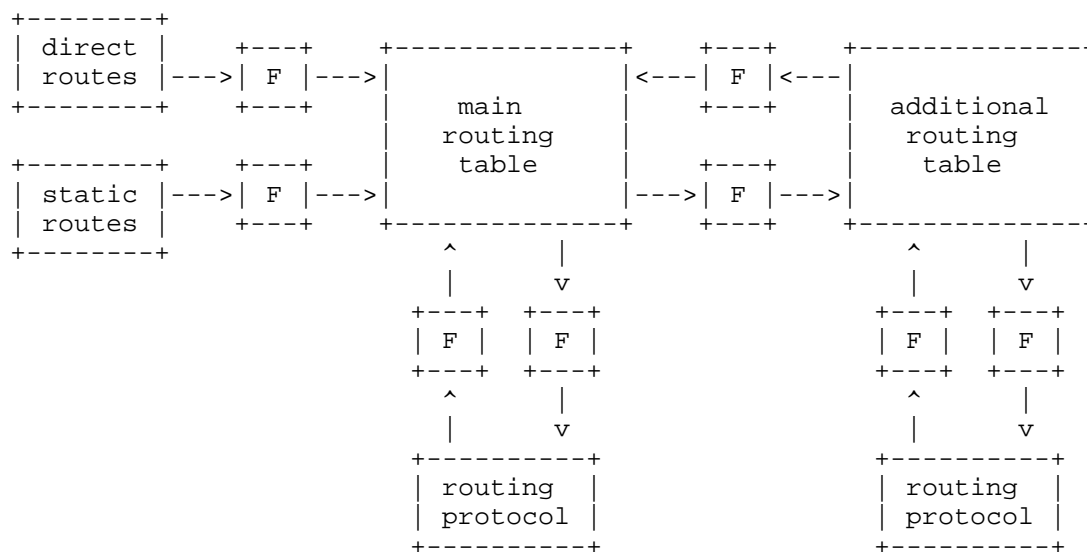


Figure 2: Example setup of the routing subsystem

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

- o Along with the main routing table, which must always be present, an additional routing table is configured.
- o Each routing protocol instance, including the "static" and "direct" pseudo-protocols, is connected to one routing table with which it can exchange routes (in both directions, except for the "static" and "direct" pseudo-protocols).
- o Routing tables may also be connected to each other and exchange routes in either direction (or both).
- o Route exchanges along all connections may be controlled by means of route filters, denoted by "F" in Figure 2.

#### 4.1. Router

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

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

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

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

##### 4.1.1. Configuration of IPv6 Router Interfaces

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

- o send-advertisements,
- o max-rtr-adv-interval,
- o min-rtr-adv-interval,
- o managed-flag,
- o other-config-flag,
- o link-mtu,
- o reachable-time,
- o retrans-timer,

- o cur-hop-limit,
- o default-lifetime,
- o prefix-list: a list of prefixes to be advertised. The following parameters are associated with each prefix in the list:
  - \* valid-lifetime,
  - \* on-link-flag,
  - \* preferred-lifetime,
  - \* autonomous-flag.

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

NOTES:

1. The "IsRouter" flag, which is also required by [RFC4861], is implemented in the "ietf-ip" module [YANG-IP] (leaf "ip:ip-forwarding").
2. The original specification [RFC4861] allows the implementations to decide whether the "valid-lifetime" and "preferred-lifetime" parameters remain the same in consecutive advertisements, or decrement in real time. However, the latter behavior seems problematic because the values might be reset again to the (higher) configured values after a configuration is reloaded. Moreover, no implementation is known to use the decrementing behavior. The "ietf-ipv6-unicast-routing" module therefore assumes the former behavior with constant values.

#### 4.2. 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:

- o "destination-prefix": IP prefix specifying the set of destination addresses for which the route may be used. This attribute is mandatory.
- o "next-hop": IP address of an adjacent router or host to which packets with destination addresses belonging to "destination-prefix" should be sent.



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

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

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

#### 4.3. Routing Tables

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

- o "source-protocol": name of the routing protocol from which the route was originally obtained.
- o "age": number of seconds since the route was created or last updated.

Each routing table may contain only routes of the same address family. Address family information consists of two parameters - "address-family" and "safi" (Subsequent Address Family Identifier, SAFI). The permitted values for these two parameters are defined by IANA and represented using YANG enumeration types "ianaaf:address-family" and "ianaaf:subsequent-address-family" [IANA-IF-AF].

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

One routing table MUST be present for each router instance and each address family supported by that router instance. It is the so-called main routing table to which all routing protocol instances supporting the given address family SHOULD be connected by default. For the two address families that are part of the core routing data model, the names of the main routing tables SHOULD be as follows:

- o "main-ipv4-unicast" for IPv4 unicast,

- o "main-ipv6-unicast" for IPv6 unicast.

Additional routing tables MAY be configured by creating new entries in the "routing-table" list, either as a part of factory-default configuration, or by a client's action.

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

The way how the routing system uses information from routing tables is outside the scope of this document. Typically, implementations will either use a forwarding table, or perform a direct look-up in the main routing table in conjunction with a route cache.

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

#### 4.4. Routing Protocols

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

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

Routes learned from the network by a routing protocol are passed to the connected routing table(s) and vice versa - routes appearing in a routing table are passed to all routing protocols connected to the table (except "direct" and "static" pseudo-protocols) and may be 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:

- o import filter controls which routes are passed from a routing protocol instance to the routing table,
- o export filter controls which routes the routing protocol instance may receive from the connected routing table.

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

#### 4.4.1. Routing Pseudo-Protocols

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

Every router instance MUST contain exactly one instance of the "direct" pseudo-protocol type. The name of this instance MUST also be "direct". It is the source of direct routes for all configured address families. Direct routes are normally supplied by the operating system kernel, based on the configuration of network interface addresses, see Section 5.2. Direct routes SHOULD by default appear in the main routing table for each configured address family. However, using the framework defined in this document, the target routing table for direct routes MAY be changed by connecting the "direct" protocol instance to a non-default routing table. Direct routes can also be filtered before they appear in the routing table.

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

#### 4.4.2. Defining New Routing Protocols

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

- o A new identity MUST be defined for the routing protocol and its base identity MUST be set to "rt:routing-protocol", or to an identity derived from "rt:routing-protocol".
- o Additional route attributes MAY be defined, preferably in one place by means of defining a YANG grouping. The new attributes have to be inserted as operational state data by augmenting the definition of "rt:route" inside "rt:routing-table", and possibly to other places in the configuration, operational state data and RPC input or output.
- o Per-interface configuration parameters can be added by augmenting the data node "rt:interface" (the list of router interfaces).
- o Other configuration parameters and operational state data can be defined by augmenting the "routing-protocol" data node. By using the "when" statement, this augment SHOULD be made conditional and valid only if the value of the "rt:type" child leaf equals to the new protocol's identity.

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

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

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

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

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

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

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

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

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

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

#### 4.5. Route Filters

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

Route filters are global, which means that a configured route filter

may be used by any or all router instances.

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

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

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

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

Each route filter is identified by a name which MUST be unique within the entire configuration. Its type MUST be specified by the "type" identity reference - this opens the space for multiple route filtering framework implementations. The default value for the route filter type is the identity "deny-all-route-filter".

#### 4.6. RPC Operations

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

- o active-route,
- o route-count.

Their parameters and semantics are described in the following subsections.

##### 4.6.1. Operation "active-route"

Description: Retrieve one or more active routes from the forwarding information base (FIB) of a router instance, i.e., the route(s) that are currently used by that router instance for sending datagrams to the destination whose address is provided as an input parameter.

Parameters:

router-name: Name of the router instance whose FIB is to be queried.

destination-address: Network layer destination address for which the active routes are requested.

Positive Response: One or more "route" elements containing the active route(s).

Negative Response:

If the logical router is not found, the server sends an "rpc-error" message with "error-tag" set to "data-missing", and "error-app-tag" set to "router-not-found".

If no route exists for the given destination address, the server sends an "rpc-error" message with "error-tag" set to "data-missing" and "error-app-tag" set to "no-route".

#### 4.6.2. Operation "route-count"

Description: Retrieve the total number of routes in a routing table.

Parameters:

router-name: Name of the logical router containing the routing table.

routing-table: Name of the routing table.

Positive Response: Element "number-of-routes" containing the requested nonnegative number.

Negative Response: If the logical router or the routing table is not found, the server sends an "rpc-error" message with "error-tag" set to "data-missing", and "error-app-tag" set to "router-not-found" or "routing-table-not-found", respectively.



## 5. Interactions with Other YANG Modules

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

### 5.1. Module "ietf-interfaces"

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

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

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

### 5.2. Module "ietf-ip"

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

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

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

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

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

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

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

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

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

In addition, the "ietf-ip" module allows for configuring IPv4 and IPv6 addresses and subnet masks. Configuration of these parameters

on an enabled interface MUST result in an immediate creation of the corresponding direct route (usually in the main routing table). Its destination prefix is set according to the configured IP address and subnet mask, and the interface is set as the outgoing interface for that route.

## 6. Routing YANG Module

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

<CODE BEGINS> file "ietf-routing@2012-07-09.yang"

```
module ietf-routing {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-routing";  
  
    prefix "rt";  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import ietf-interfaces {  
        prefix "if";  
    }  
  
    import iana-afn-safi {  
        prefix "ianaaf";  
    }  
  
    organization  
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
  
    contact  
        "WG Web: <http://tools.ietf.org/wg/netmod/>  
        WG List: <mailto:netmod@ietf.org>  
  
        WG Chair: David Kessens  
        <mailto:david.kessens@nsn.com>  
  
        WG Chair: Juergen Schoenwaelder  
        <mailto:j.schoenwaelder@jacobs-university.de>  
  
        Editor: Ladislav Lhotka  
        <mailto:lhotka@nic.cz>  
    ";  
  
    description  
        "This YANG module defines essential components that may be used  
        for configuring a routing subsystem.  
  
        Copyright (c) 2012 IETF Trust and the persons identified as
```

authors of the code. All rights reserved.

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

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

```
";

revision 2012-07-09 {
  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.";
  }

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

/* Type Definitions */

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

/* Groupings */

grouping afn-safi {
  leaf address-family {
    type ianaaf:address-family;
    default "ipv4";
    description
      "Address family of routes in the routing table.";
  }
  leaf safi {
    type ianaaf:subsequent-address-family;
    default "nlri-unicast";
    description
      "Subsequent address family identifier of routes in the
      routing table.";
  }
  description
    "This grouping provides two parameters specifying address
    family and subsequent address family.";
}

grouping route-content {
  description
    "Generic parameters of routes.

    A module for an address family should define a specific
```

```
        version of this grouping containing 'uses rt:route-content'.
    ";
    leaf outgoing-interface {
        type if:interface-ref;
        description
            "Outgoing interface.";
    }
}

/* RPC Methods */

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

        Parameters

        1. 'router-name',

        2. 'destination-address'.

        If the logical router with 'router-name' doesn't exist, then
        this operation will fail with error-tag 'missing-element' and
        error-app-tag 'router-not-found'.

        If there is no active route for 'destination-address', then
        this operation will fail with error-tag 'data-missing' and
        error-app-tag 'no-route'.
    ";
    input {
        leaf router-name {
            type router-ref;
            mandatory "true";
            description
                "Name of the router instance whose forwarding information
                base is being queried.";
        }
        container destination-address {
            uses afn-safi;
            description
                "Network layer destination address.

                AFN/SAFI-specific modules must augment this container with
                a leaf named 'address'.
            ";
        }
    }
}
```

```
output {
  list route {
    min-elements "1";
    uses afn-safi;
    uses route-content;
    description
      "Route contents specific for each address family should be
       defined through augmenting.";
  }
}

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

    Parameters:

    1. 'router-name',

    2. 'routing-table-name'.

    If the logical router with 'router-name' doesn't exist, then
    this operation will fail with error-tag 'missing-element' and
    error-app-tag 'router-not-found'.

    If the routing table with 'routing-table-name' doesn't exist,
    then this operation will fail with error-tag 'missing-element'
    and error-app-tag 'routing-table-not-found'.

    ";
  input {
    leaf router-name {
      type router-ref;
      mandatory "true";
      description
        "Name of the router instance containing the routing
         table.";
    }
    leaf routing-table {
      type leafref {
        path "/routing/router/routing-tables/routing-table/name";
      }
      mandatory "true";
      description
        "Name of the routing table.";
    }
  }
  output {
```

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

/* Data Nodes */

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

            Network layer interfaces assigned to the router must have
            their entries in the "interfaces" list.
            ';
        leaf name {
            type string;
            description
                "The unique router name.";
        }
        leaf router-id {
            type inet:ipv4-address;
            description
                "Global router ID in the form of an IPv4 address.

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

                Routing protocols may override this global parameter
                inside their configuration.
                ";
        }
        leaf description {
            type string;
            description
                "Textual description of the router.";
        }
        leaf enabled {
            type boolean;
        }
    }
}
```



```
    default "true";
    description
        "Enable the router. The default value is 'true'."

        If this parameter is false, the parent router instance is
        disabled, despite any other configuration that might be
        present.
    ";
}
container interfaces {
    description
        "Router interface parameters.";
    list interface {
        key "name";
        description
            "List of network layer interfaces assigned to the router
            instance.";
        leaf name {
            type if:interface-ref;
            description
                "A reference to the name of a configured network layer
                interface.";
        }
    }
}
container routing-protocols {
    description
        "Container for the list of configured routing protocol
        instances.";
    list routing-protocol {
        key "name";
        description
            "An instance of a routing protocol.";
        leaf name {
            type string;
            description
                "The name of the routing protocol instance.";
        }
        leaf description {
            type string;
            description
                "Textual description of the routing protocol
                instance.";
        }
        leaf type {
            type identityref {
                base routing-protocol;
            }
        }
    }
}
```

```

    mandatory "true";
    description
        "Type of the routing protocol - an identity derived
        from the 'routing-protocol' base identity.";
}
container connected-routing-tables {
    description
        "Container for connected routing tables.";
    list routing-table {
        must "not(..../..../..../routing-tables/"
            + "routing-table[rt:name=current()/"
            + "preceding-sibling::routing-table/name]/"
            + "address-family=..../..../..../routing-tables/"
            + "routing-table[rt:name=current()/name]/"
            + "address-family and ..../..../..../routing-tables/"
            + "routing-table[rt:name=current()/"
            + "preceding-sibling::routing-table/name]/safi=../"
            + "...../routing-tables/"
            + "routing-table[rt:name=current()/name]/safi)" {
            error-message "Each routing protocol may have no "
                + "more than one connected routing "
                + "table for each AFN and SAFI.";
        }
        description
            "For each AFN/SAFI pair there may be at most one
            connected routing table.";
    }
    key "name";
    description
        "List of routing tables to which the routing protocol
        instance is connected.

        If no connected routing table is defined for an
        address family, the routing protocol should be
        connected by default to the main routing table for
        that address family.

        ";
    leaf name {
        type leafref {
            path "...../routing-tables/routing-table/"
                + "name";
        }
        description
            "Reference to an existing routing table.";
    }
    leaf import-filter {
        type leafref {
            path "/routing/route-filters/route-filter/name";
        }
    }
}

```

```
        description
            "Reference to a route filter that is used for
            filtering routes passed from this routing protocol
            instance to the routing table specified by the
            'name' sibling node. If this leaf is not present,
            the behavior is protocol-specific, but typically
            it means that all routes are accepted.";
    }
    leaf export-filter {
        type leafref {
            path "/routing/route-filters/route-filter/name";
        }
        description
            "Reference to a route filter that is used for
            filtering routes passed from the routing table
            specified by the 'name' sibling node to this
            routing protocol instance. If this leaf is not
            present, the behavior is protocol-specific -
            typically it means that all routes are accepted,
            except for the 'direct' and 'static'
            pseudo-protocols which accept no routes from any
            routing table.";
    }
}
}
container static-routes {
    must "../type='rt:static'" {
        error-message "Static routes may be configured only "
            + "for 'static' routing protocol.";
        description
            "This container is only valid for the 'static'
            routing protocol.";
    }
    description
        "Configuration of 'static' pseudo-protocol.";
}
}
}
container routing-tables {
    description
        "Container for configured routing tables.";
    list routing-table {
        key "name";
        description
            "Each entry represents a routing table identified by the
            'name' key. All routes in a routing table must have the
            same AFN and SAFI.";
        leaf name {
```

```
    type string;
    description
      "The name of the routing table.";
  }
  uses afn-safi;
  leaf description {
    type string;
    description
      "Textual description of the routing table.";
  }
  container routes {
    config "false";
    description
      "Current contents of the routing table (operational
      state data).";
    list route {
      description
        "A routing table entry. This data node must augmented
        with information specific for routes of each address
        family.";
      uses route-content;
      leaf source-protocol {
        type leafref {
          path "/routing/router/routing-protocols/"
            + "routing-protocol/name";
        }
        mandatory "true";
        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 age {
        type uint32;
        units "seconds";
        mandatory "true";
        description
          "The number of seconds since the parent route was
          created or last updated.";
      }
    }
  }
  container recipient-routing-tables {
    description
      "Container for recipient routing tables.";
    list recipient-routing-table {
      key "name";
    }
  }
```

```

        description
            "A list of routing tables that receive routes from
            this routing table.";
        leaf name {
            type leafref {
                path "/routing/router/routing-tables/"
                    + "routing-table/name";
            }
            description
                "The name of the recipient routing table.";
        }
        leaf filter {
            type leafref {
                path "/routing/route-filters/route-filter/name";
            }
            description
                "A route filter which is applied to the routes
                passed on to the recipient routing table.";
        }
    }
}

container route-filters {
    description
        "Container for configured route filters.";
    list route-filter {
        key "name";
        description
            "Route filters are used for filtering and/or manipulating
            routes that are passed between a routing protocol and a
            routing table or vice versa, or between two routing
            tables. It is expected that other modules augment this
            list with contents specific for a particular route filter
            type.";
        leaf name {
            type string;
            description
                "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 "rt: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.";
    }
  }
}
```

<CODE ENDS>

## 7. IPv4 Unicast Routing YANG Module

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

<CODE BEGINS> file "ietf-ipv4-unicast-routing@2012-07-09.yang"

```
module ietf-ipv4-unicast-routing {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing";  
  
    prefix "v4ur";  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    organization  
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
  
    contact  
        "WG Web: <http://tools.ietf.org/wg/netmod/>  
        WG List: <mailto:netmod@ietf.org>  
  
        WG Chair: David Kessens  
        <mailto:david.kessens@nsn.com>  
  
        WG Chair: Juergen Schoenwaelder  
        <mailto:j.schoenwaelder@jacobs-university.de>  
  
        Editor: Ladislav Lhotka  
        <mailto:lhotka@nic.cz>  
        ";  
  
    description  
        "This YANG module augments the 'ietf-routing' module with basic  
        configuration and operational state data for IPv4 unicast  
        routing.  
  
        Every implementation must preconfigure a routing table with the  
        name 'main-ipv4-unicast', which is the main routing table for  
        IPv4 unicast."
```

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

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

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

```
";

revision 2012-07-09 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Configuration";
}

/* Groupings */

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

/* RPC Methods */

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



```
    leaf address {
      type inet:ipv4-address;
      description
        "IPv4 destination address.";
    }
  }

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

/* Data nodes */

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

          It is not required that the routes be sorted according
          to their "id".
          ';
      }
      leaf description {
        type string;
        description
          "Textual description of the route.";
      }
    }
  }
}
```

```
    }
    uses rt:route-content;
    uses route-content {
      refine "dest-prefix" {
        mandatory "true";
      }
    }
  }
}

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

<CODE ENDS>

## 8. IPv6 Unicast Routing YANG Module

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

<CODE BEGINS> file "ietf-ipv6-unicast-routing@2012-07-09.yang"

```
module ietf-ipv6-unicast-routing {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing";  
  
    prefix "v6ur";  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import ietf-interfaces {  
        prefix "if";  
    }  
  
    import ietf-ip {  
        prefix "ip";  
    }  
  
    organization  
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
  
    contact  
        "WG Web: <http://tools.ietf.org/wg/netmod/>  
        WG List: <mailto:netmod@ietf.org>  
  
        WG Chair: David Kessens  
        <mailto:david.kessens@nsn.com>  
  
        WG Chair: Juergen Schoenwaelder  
        <mailto:j.schoenwaelder@jacobs-university.de>  
  
        Editor: Ladislav Lhotka  
        <mailto:lhotka@nic.cz>  
        ";  
  
    description
```

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

Every implementation must preconfigure a routing table with the name 'main-ipv6-unicast', which is the main routing table for IPv6 unicast.

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

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

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

";

```
revision 2012-07-09 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Configuration";
}
```

/\* Groupings \*/

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

/\* RPC Methods \*/

```
augment "/rt:active-route/rt:input/rt:destination-address" {
  when "address-family='ipv6' and safi='nlri-unicast'" {
    description
      "This augment is valid only for IPv6 unicast.";
  }
  description
    "The 'address' leaf augments the 'rt:destination-address'
    parameter of the 'rt:active-route' operation.";
  leaf address {
    type inet:ipv6-address;
    description
      "IPv6 destination address.";
  }
}

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

/* Data nodes */

augment "/rt:routing/rt:router/rt:interfaces/rt:interface" {
  when "/if:interfaces/if:interface[name=current()/name]/ip:ipv6/"
    + "ip:enabled='true'" {
    description
      "This augment is only valid for router interfaces with
      enabled IPv6.

      NOTE: Parameter 'is-router' is not included, it is expected
      that it will be implemented by the 'ietf-ip' module.
      ";
  }
  description
    "IPv6-specific parameters of router interfaces.";
  container ipv6-router-advertisements {
    description
      "Parameters of IPv6 Router Advertisements.";
    reference
      "RFC 4861: Neighbor Discovery for IP version 6 (IPv6).

      RFC 4862: IPv6 Stateless Address Autoconfiguration.
      ";
  }
}
```

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

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

    Its default value is dynamic:

    - if max-rtr-adv-interval >= 9 seconds, the default value
      is 0.33 * max-rtr-adv-interval;

    - otherwise it is 0.75 * max-rtr-adv-interval.

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

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

            The default should be set to the value specified in IANA
            Assigned Numbers that was in effect at the time of
            implementation.
            ";
        reference
            "IANA: IP Parameters,
            http://www.iana.org/assignments/ip-parameters";
    }
}
```

```
leaf default-lifetime {
  type uint16 {
    range "0..9000";
  }
  units "seconds";
  description
    "The value to be placed in the Router Lifetime field of
    Router Advertisements sent from the interface, in seconds.
    MUST be either zero or between max-rtr-adv-interval and
    9000 seconds. A value of zero indicates that the router is
    not to be used as a default router. These limits may be
    overridden by specific documents that describe how IPv6
    operates over different link layers.

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

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



```
        This may be used for removing the prefix from the
        default set of advertised prefixes.
    ";
}
case advertise {
    leaf valid-lifetime {
        type uint32;
        units "seconds";
        default "2592000";
        description
            "The value to be placed in the Valid Lifetime in
            the Prefix Information option, in seconds. The
            designated value of all 1's (0xffffffff)
            represents infinity.
        ";
    }
    leaf on-link-flag {
        type boolean;
        default "true";
        description
            "The value to be placed in the on-link flag
            ('L-bit') field in the Prefix Information
            option.";
    }
    leaf preferred-lifetime {
        type uint32;
        units "seconds";
        must ". <= ../valid-lifetime" {
            description
                "This value must not be larger than
                valid-lifetime.";
        }
        default "604800";
        description
            "The value to be placed in the Preferred Lifetime
            in the Prefix Information option, in seconds. The
            designated value of all 1's (0xffffffff)
            represents infinity.
        ";
    }
    leaf autonomous-flag {
        type boolean;
        default "true";
        description
            "The value to be placed in the Autonomous Flag
            field in the Prefix Information option.";
    }
}
}
```

```

    }
  }
}

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

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

augment "/rt:routing/rt:router/rt:routing-tables/rt:routing-table/"
+ "rt:routes/rt:route" {

```

```
    when "../../../rt:address-family='ipv6' and "  
      + "../../../rt:safi='nlri-unicast'" {  
        description  
          "This augment is valid only for IPv6 unicast.";  
      }  
      description  
        "This augment defines the content of IPv6 unicast routes.";  
      uses route-content;  
    }  
  }  
<CODE ENDS>
```

## 9. IANA Considerations

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

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

Registrant Contact: The IESG.

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

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

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

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

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

## 10. Security Considerations

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

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

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

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

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

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

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

## 11. Acknowledgments

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

## 12. References

### 12.1. Normative References

- [IANA-IF-AF] Bjorklund, M., "IANA Interface Type and Address Family YANG Modules", draft-ietf-netmod-iana-if-type-02 (work in progress), April 2012.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, September 2010.
- [RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, September 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "NETCONF Configuration Protocol", RFC 6241, June 2011.
- [YANG-IF] Bjorklund, M., "A YANG Data Model for Interface Configuration", draft-ietf-netmod-interfaces-cfg-04 (work in progress), April 2012.
- [YANG-IP] Bjorklund, M., "A YANG Data Model for IP Configuration", draft-ietf-netmod-ip-cfg-03 (work in progress), April 2012.

### 12.2. Informative References

- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.



## Appendix A. Example: Adding a New Routing Protocol

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

```
<CODE BEGINS> file "example-rip@2012-07-09.yang"
```

```
module example-rip {  
    namespace "http://example.com/rip";  
  
    prefix "rip";  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    identity rip {  
        base rt:routing-protocol;  
        description  
            "Identity for the RIP routing protocol.";  
    }  
  
    typedef rip-metric {  
        type uint8 {  
            range "0..16";  
        }  
    }  
  
    grouping route-content {  
        description  
            "RIP-specific route content.";  
        leaf metric {  
            type rip-metric;  
        }  
        leaf tag {  
            type uint16;  
            default "0";  
            description  
                "This leaf may be used to carry additional info, e.g. AS  
                number.";  
        }  
    }  
}
```

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

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

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

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

```
        "Time interval between periodic updates.";
    }
}
}
<CODE ENDS>
```

## Appendix B. Example: Reply to the NETCONF &lt;get&gt; Message

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

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

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

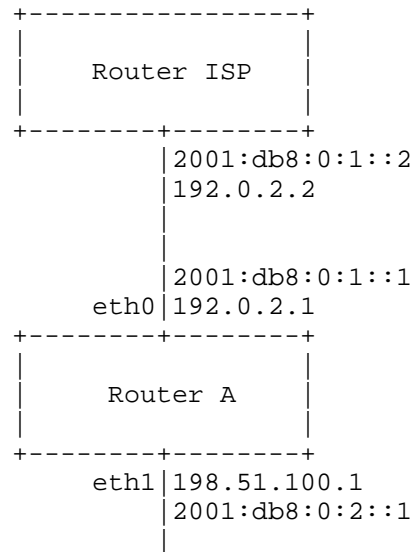


Figure 3: Example network configuration

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

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

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

```
</if:interface>
</if:interfaces>
<rt:routing>
  <rt:router>
    <rt:name>rtr0</rt:name>
    <rt:interfaces>
      <rt:interface>
        <rt:name>eth0</rt:name>
      </rt:interface>
      <rt:interface>
        <rt:name>eth1</rt:name>
        <v6ur:ipv6-router-advertisements>
          <v6ur:send-advertisements>true</v6ur:send-advertisements>
          <v6ur:prefix-list>
            <v6ur:prefix>
              <v6ur:prefix-spec>2001:db8:0:2::/64</v6ur:prefix-spec>
            </v6ur:prefix>
          </v6ur:prefix-list>
        </v6ur:ipv6-router-advertisements>
      </rt:interface>
    </rt:interfaces>
    <rt:routing-protocols>
      <rt:routing-protocol>
        <rt:name>direct</rt:name>
        <rt:type>rt:direct</rt:type>
      </rt:routing-protocol>
      <rt:routing-protocol>
        <rt:name>st0</rt:name>
        <rt:description>
          Static routing is used for the internal network.
        </rt:description>
        <rt:type>rt:static</rt:type>
        <rt:static-routes>
          <v4ur:ipv4>
            <v4ur:route>
              <v4ur:id>1</v4ur:id>
              <v4ur:dest-prefix>0.0.0.0/0</v4ur:dest-prefix>
              <v4ur:next-hop>192.0.2.2</v4ur:next-hop>
            </v4ur:route>
          </v4ur:ipv4>
          <v6ur:ipv6>
            <v6ur:route>
              <v6ur:id>1</v6ur:id>
              <v6ur:dest-prefix>::/0</v6ur:dest-prefix>
              <v6ur:next-hop>2001:db8:0:1::2</v6ur:next-hop>
            </v6ur:route>
          </v6ur:ipv6>
        </rt:static-routes>
      </rt:routing-protocol>
    </rt:routing-protocols>
  </rt:router>
</rt:routing>
```

```
<rt:connected-routing-tables>
  <rt:routing-table>
    <rt:name>main-ipv4-unicast</rt:name>
  </rt:routing-table>
  <rt:routing-table>
    <rt:name>main-ipv6-unicast</rt:name>
  </rt:routing-table>
</rt:connected-routing-tables>
</rt:routing-protocol>
</rt:routing-protocols>
<rt:routing-tables>
  <rt:routing-table>
    <rt:name>main-ipv4-unicast</rt:name>
    <rt:routes>
      <rt:route>
        <v4ur:dest-prefix>192.0.2.1/24</v4ur:dest-prefix>
        <rt:outgoing-interface>eth0</rt:outgoing-interface>
        <rt:source-protocol>direct</rt:source-protocol>
        <rt:age>3512</rt:age>
      </rt:route>
      <rt:route>
        <v4ur:dest-prefix>198.51.100.0/24</v4ur:dest-prefix>
        <rt:outgoing-interface>eth1</rt:outgoing-interface>
        <rt:source-protocol>direct</rt:source-protocol>
        <rt:age>3512</rt:age>
      </rt:route>
      <rt:route>
        <v4ur:dest-prefix>0.0.0.0/0</v4ur:dest-prefix>
        <rt:source-protocol>st0</rt:source-protocol>
        <v4ur:next-hop>192.0.2.2</v4ur:next-hop>
        <rt:age>2551</rt:age>
      </rt:route>
    </rt:routes>
  </rt:routing-table>
  <rt:routing-table>
    <rt:name>main-ipv6-unicast</rt:name>
    <rt:address-family>ipv6</rt:address-family>
    <rt:safi>nlri-unicast</rt:safi>
    <rt:routes>
      <rt:route>
        <v6ur:dest-prefix>2001:db8:0:1::/64</v6ur:dest-prefix>
        <rt:outgoing-interface>eth0</rt:outgoing-interface>
        <rt:source-protocol>direct</rt:source-protocol>
        <rt:age>3513</rt:age>
      </rt:route>
      <rt:route>
        <v6ur:dest-prefix>2001:db8:0:2::/64</v6ur:dest-prefix>
        <rt:outgoing-interface>eth1</rt:outgoing-interface>
```

```
    <rt:source-protocol>direct</rt:source-protocol>
    <rt:age>3513</rt:age>
  </rt:route>
  <rt:route>
    <v6ur:dest-prefix>::/0</v6ur:dest-prefix>
    <v6ur:next-hop>2001:db8:0:1::2</v6ur:next-hop>
    <rt:source-protocol>st0</rt:source-protocol>
    <rt:age>2550</rt:age>
  </rt:route>
</rt:routes>
</rt:routing-table>
</rt:routing-tables>
</rt:router>
</rt:routing>
</data>
</rpc-reply>
```



## Appendix C. Change Log

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

## C.1. Changes Between Versions -03 and -04

- o Changed "error-tag" for both RPC methods from "missing element" to "data-missing".
- o Removed the decrementing behavior for advertised IPv6 prefix parameters "valid-lifetime" and "preferred-lifetime".
- o Changed the key of the static route lists from "seqno" to "id" because the routes needn't be sorted.
- o Added 'must' constraint saying that "preferred-lifetime" must not be greater than "valid-lifetime".

## C.2. Changes Between Versions -02 and -03

- o Module "iana-afn-safi" moved to I-D "iana-if-type".
- o Removed forwarding table.
- o RPC "get-route" changed to "active-route". Its output is a list of routes (for multi-path routing).
- o New RPC "route-count".
- o For both RPCs, specification of negative responses was added.
- o Relaxed separation of router instances.
- o Assignment of interfaces to router instances needn't be disjoint.
- o Route filters are now global.
- o Added "allow-all-route-filter" for symmetry.
- o Added Section 5 about interactions with "ietf-interfaces" and "ietf-ip".
- o Added "router-id" leaf.
- o Specified the names for IPv4/IPv6 unicast main routing tables.
- o Route parameter "last-modified" changed to "age".

- o Added container "recipient-routing-tables".

#### C.3. Changes Between Versions -01 and -02

- o Added module "ietf-ipv6-unicast-routing".
- o The example in Appendix B now uses IP addresses from blocks reserved for documentation.
- o Direct routes appear by default in the FIB table.
- o Network layer interfaces must be assigned to a router instance. Additional interface configuration may be present.
- o The "when" statement is only used with "augment", "must" is used elsewhere.
- o Additional "must" statements were added.
- o The "route-content" grouping for IPv4 and IPv6 unicast now includes the material from the "ietf-routing" version via "uses rt:route-content".
- o Explanation of symbols in the tree representation of data model hierarchy.

#### C.4. Changes Between Versions -00 and -01

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

Author's Address

Ladislav Lhotka  
CZ.NIC

Email: [lhotka@nic.cz](mailto:lhotka@nic.cz)



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: January 31, 2013

A. Atlas, Ed.  
T. Nadeau  
Juniper Networks  
D. Ward  
Cisco Systems  
July 30, 2012

Interface to the Routing System Framework  
draft-ward-irs-framework-00

Abstract

This document describes a framework for a standard, programmatic interface for full-duplex, streaming state transfer in and out of the Internet's routing system. It lists the information that might be exchanged over the interface, and describes the uses of an interface to the Internet routing system.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 31, 2013.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Functional Overview . . . . .	3
1.2. Example Use-Cases . . . . .	5
2. Programmatic Interfaces . . . . .	6
3. Common Interface Considerations . . . . .	7
3.1. Capabilities . . . . .	7
3.2. Identity, Authorization, Authentication, and Security . . . . .	8
3.3. Speed and Frequency of State Installation . . . . .	8
3.4. Lifetime of IRS-Installed Routing System State . . . . .	9
3.5. Start-Time of IRS-Installed Routing System State . . . . .	10
4. Bidirectional Interfaces to the Routing System . . . . .	10
4.1. Static Routing . . . . .	11
4.1.1. Routing Information Base Interface . . . . .	11
4.1.2. Label Forwarding Information Base Interface . . . . .	12
4.1.3. Multicast Routing Information Base Interface . . . . .	13
4.2. Beyond Destination-based Routing . . . . .	13
4.2.1. Policy-Based Routing Interface . . . . .	13
4.2.2. QoS State . . . . .	14
4.3. Protocol Interactions . . . . .	14
4.3.1. IGP Interfaces . . . . .	14
4.3.2. BGP Interface . . . . .	15
4.3.3. PIM and mLDP Interfaces . . . . .	15
4.4. Triggered Sessions and Signaling . . . . .	16
4.4.1. OAM-related Sessions Interface . . . . .	16
4.4.2. Dynamic Session Creation . . . . .	16
4.4.3. Triggered Signaling . . . . .	16
5. Interfaces for Learned Information from the Routing System . . . . .	16
5.1. Efforts to Obtain Topological Data . . . . .	17
5.2. Measurements . . . . .	18
5.3. Events . . . . .	18
6. Manageability Considerations . . . . .	19
7. IANA Considerations . . . . .	19
8. Security Considerations . . . . .	19
9. Acknowledgements . . . . .	20
10. Informative References . . . . .	20
Authors' Addresses . . . . .	21

## 1. Introduction

Routers that form the Internet's routing infrastructure maintain state at various layers of detail and function. For example, each router has a Routing Information Base (RIB), and the routing protocols (OSPF, ISIS, BGP, etc.) each maintain protocol state and information about the state of the network.

A router also has information that may be required for applications to understand the network, verify that programmed state is installed in the forwarding plane, measure the behavior of various flows, and understand the existing configuration and state of the router. Furthermore, routers are configured or implemented with procedural or policy-based instructions for how to convert all of this information into the forwarding operations that are installed in the forwarding plane, and this is also state information that describes the behaviour of the router.

This document sets out a framework for a common, standard interface to allow access to all of this information. This Interface to the Routing System (IRS) would facilitate control and diagnosis of the routing infrastructure, as well as enabling sophisticated applications to be built on top of today's routed networks. The IRS is a programmatic, streaming interface for transferring state into and out of the Internet's routing system, and recognizes that the routing system and a router's OS provide useful mechanisms that applications could harness to accomplish application-level goals.

Fundamental to the IRS is a clear data model that defines the semantics of the information that can be written and read. The IRS provides a framework for registering for and requesting the appropriate information for each particular application. The IRS provides a way for applications to customize network behaviour while leveraging the existing routing system.

The IRS, and therefore this document, is specifically focused on an interface for routing and forwarding data.

### 1.1. Functional Overview

There are three key aspects to the IRS. First, the interface is a programmatic streaming interface meaning that it is asynchronous and offers fast, interactive access. Second, the IRS gives access to information and state that is not usually configurable or modeled in existing implementations or configuration protocols. Third, the IRS gives applications the ability to learn additional, structured, filterable information and events from the router.

IRS is described as a streaming programmatic interface; the key properties that are intended are:

**Multiple Simultaneous Asynchronous Operations:** A single application should be able to send multiple operations to IRS without needing to wait for each to complete before sending the next.

**Configuration Not Re-Processed:** When an IRS operation is processed, it does not require that any of the configuration be processed. I.e. the desired behavior with regard to static configuration is the same as learning a new BGP route - completely orthogonal.

**Duplex:** Communications can be established by either the router or the application. Similarly, events, acknowledgements, failures, operations, etc. can be sent at any time by both the router and the application. This is not a pure pull-model where only the application queries to pull responses.

**High-Throughput:** At a minimum, the IRS should be able to handle hundreds of operations per second.

**Responsive:** It should be possible to complete simple operations within a sub-second time-scale.

**Multi-Channel:** It should be possible for information to be communicated via the interface from different components in the router without requiring going through a single channel. For example, for scaling, some exported data or events may be better sent directly from the forwarding plane, while other interactions may come from the control-plane. Thus a single TCP session per application would not be a good match.

Such an interface facilitates the specification of non-permanent state into the routing system as well as the extraction of that information and additional dynamic information from the routing system. A non-routing protocol or application could inject state into a networking node's OS via the state-insertion aspects of the interface, that could then be distributed in a routing or signaling protocol.

Where existing mechanisms can provide part of the desired functionality, the coverage and gaps are briefly discussed in this document.

The existing mechanisms, such as SNMP and NetConf, that allow state to be written and read do not meet all of the above key properties needed for IRS. The overhead of infrastructure is also quite high and many MIBs do not, in definition or practice, allow writing of



state. There is also very limited capability to add new application-specific state to be distributed via the routing system. Conversely, NetConf is challenging for reading state from a router.

ForCES is another method for writing state into a router, but its focus is on the forwarding plane. By focusing on the forwarding plane, it requires that the forwarding plane be modeled and programmable and ignores the existence and intelligence of the router OS and routing system. ForCES provides a lower-level interface than IRS is intended to address.

### 1.2. Example Use-Cases

A few brief examples of ways an application could use the IRS are presented here. These are intended to give a sense of what could be done rather than to be primary and detailed motivational use-cases.

**Route Control via Indirection:** By enabling an application to install routes in the RIB, it is possible that when, for example, BGP resolves its IGP next-hop via the RIB, that could be to an application-installed route. In general, when a route is redistributed from one protocol to another, this is done via the RIB and such a route could have been installed via the IRS interface.

**Policy-Based Routing of Unknown Traffic:** A static route, installed into the RIB, could direct otherwise unrecognized traffic towards an application, through whatever appropriate tunnel was required, for further handling. Such a static route could be programmed with indirection, so that its outgoing path is whatever is used by another particular route (e.g. to a particular server).

**Services with Fixed Hours:** If an application were to provide services only during fixed time-periods, the application could install both a specific route on the local router in the RIB and advertise the associated prefix as being attached to the local router via the IGP. If the application knew the fixed hours, the state so installed could be time-based and automatically removed at approximately the correct time.

**Traffic Mirroring:** The interface to the multicast RIB could be used to mirror a particular traffic flow to both its original destination and a data collector.

**Static Multicast Trees:** An application could set up static (or partially static) multicast flows via entries in the multicast RIB without requiring an associated multicast protocol. This could be useful in networks with a fixed topology and well-planned

distribution tree that provides redundancy.

## 2. Programmatic Interfaces

A number of management interfaces exist today that allow for the indirect programming of the routing system. These include proprietary CLI, Netconf, and SNMP. However, none of these mechanisms allows for the direct programming of the routing system. Such streaming interfaces are needed to support dynamic time-based applications.

These interfaces should cater to how applications typically interact with other applications and network services rather than forcing them to use older mechanisms that are more complex to understand and implement, as well as operate.

The most critical component of the IRS is developing standard data models with their associated semantics. While many routing protocols are standardized, associated data models for IRS are not yet available. Instead, each router uses different information, mechanisms, and CLI which makes a standard interface for use by applications extremely cumbersome to develop and maintain. Well-known data modeling languages, such as YANG [RFC6020], exist and might be used for defining the necessary data models; more investigation into alternatives is required. It is understood that some portion (hopefully a small subset) will remain as proprietary extensions; the data models must support future extensions and proprietary extensions.

Since the IRS will need to support remote access between applications running on a host or server and routers in the network, at least one standard mechanism must be identified and defined to provide the transfer syntax, as defined by a protocol, used to communicate between the application and the routing system. Common functionality that IRS needs to support includes acknowledgements, dependencies, request-reserve-commit.

Appropriate candidate protocols must be identified that reduce the effort required by applications and, preferably, are familiar to application developers. Ideally, this should not require that applications understand and implement existing routing protocols to interact with IRS. These interfaces should instead be based on light-weight, rapidly deployable approaches; technology approaches must be evaluated but examples could include ReSTful web services, JSON, XMPP, and XML. These interfaces should possess self-describing attributes (e.g. a web services interface) so that applications can quickly query and learn about the active capabilities of a device.

It may be desirable to also define the local syntax (e.g. programming language APIs) that applications running local to a router can use.

Since evolution is anticipated in IRS over time, it is important that versioning and backwards compatibility are basic supported functionality. Similarly, common consistent error-handling and acknowledgement mechanisms are required that do not severely limit the scalability and responsiveness of these interfaces.

### 3. Common Interface Considerations

#### 3.1. Capabilities

Capability negotiation is a critical requirement because different implementations and software versions will have different abilities. Similarly, applications may have different capabilities for receiving exported information.

The IRS will have multiple interfaces, each with their own set of capabilities. Such capabilities may include the particular data model and what operations can be performed at what scale.

The capabilities negotiated may be filtered based upon different information, such as the application's authorization, application's capabilities, and the desired granularity for abstraction which the application understands. Different types of authorization may require the router to advertise different capabilities and restrictions.

The capability negotiation may take place at different levels of detail based upon the application and the specific functions in the IRS that the application is negotiating. The router and application must use the IRS to agree upon the proper level of abstraction for the interaction. For example, when an application describes a route between two topological items, these items may vary in detail from a network domain's name at a high level, or down to the port forwarding specifics of a particular device.

The data-model and capabilities available for an element may depend upon whether the element is physical or virtual; the virtual/physical distinction does not matter to IRS. Similarly, the location of the element may influence how an application converses with the associated router.

### 3.2. Identity, Authorization, Authentication, and Security

Applications that wish to manipulate or interrogate the state of the routing system must be appropriately authorized. This means that at least one means of determining the unique identity of an application and its associated access privileges must be available; this implies that the identity and associated access privileges must be verifiable from the router being programmed.

Furthermore, being able to associate a state and the modifications to a state with a specific application would aid in troubleshooting and auditing of the routing system. By associating identity and authorization with installed state, other applications with appropriate authority can clean up state abandoned by failed applications, if necessary.

Security of communication between the application and the router is also critical and must be considered in the design of the mechanisms to support these programmatic interfaces.

### 3.3. Speed and Frequency of State Installation

A programmatic interface does not by itself imply the frequency of state updates nor the speed at which the state installation is required. These are critical aspects of an interface and govern what an application can use the interface for. The difference between sub-second responsiveness to millions of updates and a day delay per update is, obviously, drastic. The key attributes of the programmatic interface are described in Section 1 and include that the interface must be asynchronous.

For each interface in IRS, it will be necessary to specify expected scaling, responsiveness, and performance so that applications can understand the uses to which the IRS can be used.

IRS must support asynchronous streaming real-time interactions between the applications and router. IRS must assume that there are many unrelated applications that may be simultaneously using IRS. This implies that applications must be able to subscribe to change events that notify them about changes done to state by other applications or configuration.

Furthermore, IRS should construct interfaces that cater to different scaling and frequency of update parameters. For example, slow, but detailed queries of the system, or fast yet higher level (less detailed) queries or modifications.

### 3.4. Lifetime of IRS-Installed Routing System State

In routers today, the lifetime of different routing state depends upon how that state was learned and committed. If the state is configuration state, then it is ephemeral when just in the running configuration or persistent when written to the startup configuration. If the state is learned via a routing protocol or SNMP, it is ephemeral, lasting only until the router reboots or the state is withdrawn.

Unlike previous injection mechanisms that implied the state lifetime, IRS requires that multiple models be supported for the lifetime of state it installs. This is because the lifetime or persistence of state of the routing system can vary based on the application that programmed it, policies or security authorization of the application.

There are four basic models to be supported.

**Ephemeral:** State installed by the application remains on the router in its active memory until such time as it is either removed by a routing or signaling protocol, removed by a configuration initiated by an application, or the router reboots. In the case of the latter, past state is forgotten when the router reboots.

**Persistent:** State installed by the application remains on the router across reboots or restarts of the system. It can be dynamically removed or manipulated by an application, by configuration, or by the routing system itself. This state does not appear in the router's configuration; it is processed after all the configuration upon a reboot.

**Time-Based:** When state is installed by the application, it has an expiration time specified. When that time has passed, the state is removed from the router. It can also be dynamically removed or manipulated by an application, by configuration or the routing system itself. State that hasn't expired will remain on a router through reboots.

**Time-Based Ephemeral:** When state is installed by the application, it has an expiration time specified. When that time has passed, the state is removed from the router. It can also be dynamically removed or manipulated by an application, by configuration, by the routing system itself, or by the router rebooting. Past state is forgotten after the router reboots.

### 3.5. Start-Time of IRS-Installed Routing System State

To provide flexibility, pre-programming, and handle dependencies, it is necessary to have multiple models of when a operation is to be handled. There are the following basic models to be supported.

**Immediate:** When the operation is received, it should be acted upon as quickly as reasonable (e.g. queued with other outstanding requests if necessary).

**Time-Based:** An application may provide an operation that is to be initiated at a particular time. When the specified time is reached, the operation should be acted upon as quickly as reasonable. Implementations may, of course, strive to improve the time-accuracy at which the operation is initiated.

**Triggered:** The operation should be initiated when the specified triggering event has happened. A triggering event could be the successful or failed completion of another operation. A triggering event could be a system event, such as an interface up or down, or another event such as a particular route changing its next-hops.

Because it is possible to request operations in models other than "Immediate" and some of the start-times will be at an unknown future point (e.g. "Triggered"), it is not feasible to guarantee that the resources required by an operation will always be available without reserving them from the time the operation is received. While that type of resource reservation should be possible, applications must also be able to handle an operation failing or being preempted due to resources or due to a higher priority or better authorized application taking ownership of the associated state or resource.

## 4. Bidirectional Interfaces to the Routing System

IRS is a bidirectional programmatic interface that allows both routing and non-routing applications to install, remove, read, and otherwise manipulate the state of the routing system.

Just as the Internet routing system is not a single protocol or implementation layer, neither does it make sense for the IRS to be at a single layer or reside within a single protocol. For each protocol or layer, there are different data models, abstractions and interface syntaxes and semantics required. However, with this in mind, it is ideal that a minimal set of mechanism(s) to define, transfer and manipulate this state will be specified with as few optional characteristics as possible. This will foster better

interoperability between different vendor implementations.

Since IRS is focused on the routing system, the layers of interest start with the RIB and continue up through the IGPs, BGP, RSVP-TE, LDP, etc. The intent is neither to provide interfaces to the forwarding plane nor to provide interfaces to application layers.

It is critical that these interfaces provide the ability to learn state, filtered by request, as well as install state. IRS assumes that there will be multiple applications using IRS and therefore the ability to read state is necessary to fully know the router's state. In general, if an interface allows the setting of state, the ability to read and modify that state is also necessary.

#### 4.1. Static Routing

The ability to specify static routes exists via CLI and MIBs but these mechanisms do not provide a streaming programmatic interface. IRS solves this problem by proposing interfaces to the RIB, LFIB, and Multicast RIBs.

By installing static routes into the RIB layer, IRS is able to utilize the existing router OS and its mechanisms for distributing the selected routes into the FIB and LIB. This avoids the need to model or standardize the forwarding plane.

##### 4.1.1. Routing Information Base Interface

The RIB is populated with routes and next-hops as supplied by configuration, management, or routing protocols. A route has a preference based upon the specific source from which the route was derived. Static routes, specified via CLI, can be installed with an appropriate preference. The FIB is populated by selecting from the RIB based on policy and tie-breaking criteria.

The IRS interface should allow dynamic reading and writing of routes into the RIB. There are several important attributes associated with doing so, as follows:

**Preference Value:** This allows decisions between conflicting routes, whether IRS-installed or otherwise. IRS-installed routes can each be installed with a different preference value.

**Route Table Context:** There can be different route table contexts in the RIB. Examples include multiple protocols (e.g. IPv4, IPv6), multiple topologies, different uses, and multiple networks (e.g. VRF tables for VPNs). Appropriate application-level abstractions are required to describe the desired route table context.

Route or Traffic Identification The specific IP prefix or even interface must be specified.

Outgoing Path and Encapsulation: It is necessary to specify the outgoing path and associated encapsulation. This may be done directly or indirectly. This is one of the more complex aspects with the following considerations.

Primary Next-Hops: To support multi-path forwarding, multiple primary next-hops can be specified and the traffic flows split among them.

Indirection: Instead of specifying particular primary next-hops, it is critical to be able to provide the ability for indirection, such as is used between BGP routes and IGP routes. Thus, the outgoing path might be specified via indirection to be the same as another route's.

Encapsulation: Associated with each primary next-hop can be details on the type of encapsulation for the packet. Such encapsulation could be MPLS, GRE, etc. as supported by the router.

Protection: For fast-reroute protection, each primary next-hop may have one or more alternate next-hops specified. Those are to be used when the primary next-hop fails.

DSCP: For QoS, the desired DSCP to be used for the outgoing traffic can be specified.

It is useful for an application to be able to read out the RIB state associated with particular traffic and be able to learn both the preferred route and its source as well as other candidates with lower preference.

Although there is no standardized model or specification of a RIB, it may be possible to build an interoperable bi-directional interface without one.

#### 4.1.2. Label Forwarding Information Base Interface

The LFIB has a similar role to the RIB for MPLS labeled packets. Each entry has slightly different information to accommodate MPLS forwarding and semantics. Although static MPLS can be used to configure specific state into the LFIB, there is no bidirectional programmatic interface to program, modify, or read the associated state.



Each entry in the LFIB requires a MPLS label context (e.g. platform, per-interface, or other context), incoming label, label operation, and next-hops with associated encapsulation, label operation, and so on. Via the IRS LFIB interface, an application could supply the information for an entry using either a pre-allocated MPLS label or a newly allocated MPLS label that is returned to the application.

#### 4.1.3. Multicast Routing Information Base Interface

There is no bidirectional programmatic interface to add, modify, remove or read state from the multicast RIB. This IRS interface would add those capabilities.

Multicast forwarding state can be set up by a variety of protocols. As with the unicast RIB, an application may wish to install a new route for multicast. The state to add might be the full multicast route information - including the incoming interface, the particular multicast traffic (e.g. (source, group) or MPLS label), and the outgoing interfaces and associated encapsulations to replicate the traffic too.

The multicast state added need not match to well-known protocol installed state. For instance, traffic received on an specified set, or all, interfaces that is destined to a particular prefix from all sources or a particular prefix could be subject to the specified replication.

#### 4.2. Beyond Destination-based Routing

Routing decisions and traffic treatment is not merely expressible via destination-based routing or even (S, G) routing, such as in multicast. Capturing these aspects into appropriate interfaces for the IRS provides the ability for applications to control them as well.

##### 4.2.1. Policy-Based Routing Interface

A common feature of routers is the ability to specify policy-based routing (PBR) rules for accepting, dropping, or differently forwarding particular traffic. This is a very useful functionality for an application to be able to rapidly add and remove state into. Such state would indicate the particular traffic to be affected and its subsequent behavior (e.g. drop, accept, forward on specified outgoing path and encapsulation, QoS, DSCP marking, policing, etc.). Such state is made more complex by the potential importance of ordering among the PBR rules.

While PBR rules can be specified via CLI, this mechanism is not a

streaming programmatic interface nor is there generally the ability to specify particular time-based lifetimes for each rule.

#### 4.2.2. QoS State

While per-hop behaviors are defined as well as standard DSCP meanings, the details of QoS configuration are not standardized and can be highly variable depending upon platform. It is NOT a goal of this work to standardize QoS configurations. Instead, a data object model can define push/pull configurations. More investigation is needed to better describe the details.

#### 4.3. Protocol Interactions

Providing IRS interfaces to the various routing protocols allows applications to specify policy, local topology changes, and availability to influence the routing protocols in a way that the detailed addition or modification of routes in the RIB does not.

The decision to distribute the routing state via a routing or signaling protocol depends upon the protocol-layer at which this state is injected into the routing system. It may also depend upon which routing domain or domains this information is injected as well.

In addition it is necessary to have the ability to pull state regarding various protocols from the router, a mechanism to register for asynchronous events, and the means to obtain those asynchronous events. An example of such state might be peer up/down.

##### 4.3.1. IGP Interfaces

The lack of a streaming programmatic interface to the IGPs limits the ability of applications to influence and modify the desired behavior of the IGP.

An application may need to indicate that a router is overloaded (via ISIS or the method described in [RFC3137]) because that router does not yet have sufficient state synchronized or installed into it. When critical state is provided not merely by routers but also from applications via the IRS, a synchronization mechanism can be needed.

The ability for an application to modify the local topology can be part of this interface. One possibility is to allow modification of local interface metrics to generally influence selected routes. A more extensive interface might include the ability to create a OSPF or ISIS adjacency across a specified interface (virtual or real) with the appropriate associated encapsulation.

The ability to attach a prefix to the local router would provide a straightforward method for an application to program a single router and have the proper routes computed and installed by all other routers in the relevant domains. Additional aspects to the prefix attachment, such as the metric with which to attach the prefix and fast-reroute characteristics, would be part of the interface.

Beyond such pure routing information, the need for an application to be able to install state to be flooded via an IGP has already been recognized. [I-D.ietf-isis-genapp] specifies a mechanism for flooding generalized application information via ISIS, but does not describe how an application can generate or consume this information. Similarly, [RFC5250] specifies Opaque LSAs for OSPF to provide for application-specific information to be flooded. An IRS interface and associated data object model would provide such a mechanism.

Additional investigation will identify other state that applications may wish to install.

From the IGP, applications via IRS can extract significant topological information about the routers, links, and associated attributes.

#### 4.3.2. BGP Interface

BGP carries significant policy and per-application specific information as well as internet routes. A significant interface into BGP is expected, with different data object models for different applications. For example, the IRS interface to BGP could provide the ability to specify the policy on which paths BGP chooses to advertise. Additionally, the ability to specify information with an application-specified AFI/SAFI could provide substantial flexibility and control.

An existing example of application information carried in BGP is BGP Flowspec [RFC5575] which can be used to provide traffic filtering and aid in handling denial-of-service attacks.

The ability to extract information from BGP is also quite critical. A useful example of this is the information available from BGP via [I-D.gredler-idr-ls-distribution], which allows link-state topology information to be carried in BGP.

#### 4.3.3. PIM and mLDP Interfaces

For PIM and mLDP, there are at least two types of state that an application might wish to install. First, an application might add an interface to join a particular multicast group. Second, an

application might provide an upstream route for traffic to be received from - rather than having PIM or mLDP need to consult the unicast RIB.

Additional investigation will identify other state that applications may wish to install.

#### 4.4. Triggered Sessions and Signaling

##### 4.4.1. OAM-related Sessions Interface

An application may need to trigger new OAM sessions (e.g. BFD, VCCP, etc.) using an appropriate template. For example, there may be applications that need to create a new tunnel, verify its functionality via new triggered OAM sessions, and then bring it into service if that OAM indicates successful functionality. More investigation is needed to better describe the details.

##### 4.4.2. Dynamic Session Creation

An application may wish to trigger a peering relationship for a protocol. For instance, a targeted LDP session may be required to exchange state installed locally with a remote router. More investigation is needed to better describe the different cases and details.

##### 4.4.3. Triggered Signaling

To easily create dynamic state throughout the network, an application may need to trigger signaling via protocols such as RSVP-TE. An example of such an application can be a Stateful Path Computation Element (PCE)[I-D.ietf-pce-stateful-pce], which has control of various LSPs that need to be signaled.

More investigation is needed to better describe the different cases and details.

#### 5. Interfaces for Learned Information from the Routing System

Just as applications need to inject state into the routing system to meet various application-specific and policy-based requirements, it is critical that applications be able to also extract necessary state from the routing system.

A part of each of these interfaces is the ability to specify the generation of the desired information (e.g., collecting specific per-flow measurements) and the ability to specify appropriate filters to

indicate the specifics and abstraction level of the information to be provided

The types of information to extract can be generally grouped into the following different categories.

**Topological:** The need to understand the network topology, at a suitable abstraction layer, is critical to applications. Connectivity is not sufficient - the associated costs, bandwidths, latencies, etc. are all important aspects of the network topology that strongly influence the decision-making and behavior of applications.

**Measurements:** Applications require measurements of traffic and network behavior in order to have a more meaningful feedback control loop. Such information may be per-interface, per-flow, per-firewall rule, per-queue, etc.

**Events:** There are a variety of asynchronous events that an application may require or use as triggering conditions for starting other operations. An obvious example is interface state events.

**Configuration:** For some aspects, it may be necessary for applications to be able to learn about the routing configuration on a box. This is partially available via various MIBs and NetConf. What additional information needs to be exported and the appropriate mechanisms needs further examination.

The need to extract information from the network is not new; there is on-going work in the IETF in this area. This framework describes those efforts in the context of the above categories and starts the discussion of the aspects still required.

### 5.1. Efforts to Obtain Topological Data

Topological data can be defined and presented at different layers (e.g. Layer-2, Layer-3) and with different characteristics exposed or hidden (e.g. physical or virtual, SRLGs, bandwidth, latency, etc.). It can also have different states, such as configured but unavailable, configurable, active, broken, administratively disabled, etc.

To solve the problem of only being able to obtain topological data via listening to the IGP in each area, BGP-LS [I-D.gredler-idr-ls-distribution] defines extensions to BGP so that link-state topology information can be carried in BGP and a single BGP listener in the AS can therefore learn and distribute the entire

AS's current link-state topology. BGP-LS solves the problem of distributing topological information throughout the network. While IRS may expand the information to be distributed, IRS addresses the API aspect of BGP-LS and not the network-wide distribution.

At another level, ALTO [RFC5693] provides topological information at a higher abstraction layer, which can be based upon network policy, and with application-relevant services located in it. The mechanism for ALTO obtaining the topology can vary and policy can apply to what is provided or abstracted.

Neither of these fully meet the need to obtain detailed, layered topological state that provides more information than the current functional status. While there are currently no sufficiently complete standards, the need for such functionality can be deduced by the number of proprietary systems that have been developed to obtain and manage topology; even Element Management Systems start with the need for learning and manipulating the topology. Similarly, orchestration layers for applications start with the need to manage topology and the associated database.

Detailed topology includes aspects such as physical nodes, physical links, virtual links, port to interface mapping, etc. The details should include the operational and administrative state as well as relevant parameters ranging from link bandwidth to SRLG membership. Layering is critical to provide the topology at the level of abstraction where it can be easily used by the application.

A key aspect of this interface is the ability to easily rate-limit, filter and specify the desired information to be extracted. This will help in allowing the interface to scale when queries are done.

## 5.2. Measurements

IPFIX [RFC5470] provides a way to measure and export per-traffic flow statistics. Applications that need to collect information about particular flows thus have a clear need to be able to install state to configure IPFIX to measure and export the relevant flows to the appropriate collectors.

## 5.3. Events

A programmatic interface for application to subscribe to asynchronous events is necessary. In addition to the interface state events already mentioned, an application may wish to subscribe to certain OAM-triggered events that aren't otherwise exported.

A RIB-based event could be reporting when the next-hops associated

with a route have changed. Other events could be used to verify that forwarding state has been programmed. For example, an application could request an event whenever a particular route in the RIB has its forwarding plane installation completed.

When an application registers for events, the application may request to get only the first such event, all such events, or all events until a certain time.

The full set of such events, that are not specifically related to other interfaces, needs to be investigated and defined.

## 6. Manageability Considerations

Manageability plays a key aspect in IRS. Some initial examples include:

**Data Authorization Levels:** The data-models used for IRS need the ability to indicate the required authorization level for installing or reading a particular subset of data. This allows control of what interactions each application can have.

**Identity Authorization Levels:** Associated with an application's identity should be an identity authorization level that is in a hierarchy so that higher authorized applications can manage and remove the state and resources used by other applications. The top of such a hierarchy would be the router configuration itself.

**Resource Limitations:** Using IRS, applications can consume resources, whether those be operations in a time-frame, entries in the RIB, stored operations to be triggered, etc. The ability to set resource limits based upon authorization is critical.

**Configuration Interactions:** The interaction of state installed via the IRS and via a router's configuration needs to be clearly defined.

## 7. IANA Considerations

This document includes no request to IANA.

## 8. Security Considerations

This framework describes interfaces that clearly require serious consideration of security. The ability to identify, authenticate and

authorize applications that wish to install state is necessary and briefly described in Section 3.2. Security of communications from the applications is also required.

More specifics on the security requirements requires further investigation.

## 9. Acknowledgements

The authors would like to thank Ken Gray, Adrian Farrel, Bruno Rijsman, Rex Fernando, Jan Medved, John Scudder, and Hannes Gredler for their suggestions and review.

## 10. Informative References

- [I-D.gredler-idr-ls-distribution]  
Gredler, H., Medved, J., Previdi, S., and A. Farrel,  
"North-Bound Distribution of Link-State and TE Information  
using BGP", draft-gredler-idr-ls-distribution-02 (work in  
progress), July 2012.
- [I-D.ietf-isis-genapp]  
Ginsberg, L., Previdi, S., and M. Shand, "Advertising  
Generic Information in IS-IS", draft-ietf-isis-genapp-04  
(work in progress), November 2010.
- [I-D.ietf-pce-stateful-pce]  
Crabbe, E., Medved, J., Varga, R., and I. Minei, "PCEP  
Extensions for Stateful PCE",  
draft-ietf-pce-stateful-pce-01 (work in progress),  
July 2012.
- [RFC3137] Retana, A., Nguyen, L., White, R., Zinin, A., and D.  
McPherson, "OSPF Stub Router Advertisement", RFC 3137,  
June 2001.
- [RFC5250] Berger, L., Bryskin, I., Zinin, A., and R. Coltun, "The  
OSPF Opaque LSA Option", RFC 5250, July 2008.
- [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek,  
"Architecture for IP Flow Information Export", RFC 5470,  
March 2009.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J.,  
and D. McPherson, "Dissemination of Flow Specification  
Rules", RFC 5575, August 2009.



- [RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Authors' Addresses

Alia Atlas (editor)  
Juniper Networks  
10 Technology Park Drive  
Westford, MA 01886  
USA

Email: akatlas@juniper.net

Thomas Nadeau  
Juniper Networks  
1194 N. Mathilda Ave.  
Sunnyvale, CA 94089  
USA

Email: tnadeau@juniper.net

Dave Ward  
Cisco Systems  
Tasman Drive  
San Jose, CA 95134  
USA

Email: wardd@cisco.com

