

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 29, 2013

L. Huang
A. Clemm
Cisco Systems
A. Bierman
YumaWorks
February 25, 2013

YANG Data Model for Access Control List Configuration
draft-huang-netmod-acl-02.txt

Abstract

This document defines a YANG data model for the configuration of Access Control Lists (ACLs) on a device.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Copyright (c) 2013 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. Definitions and Acronyms	4
3. The Design of the ACL Data Model	5
3.1. Overall Model Structure	5
3.2. Data hierarchy	6
3.3. Other Considerations	9
3.3.1. Extensibility	9
3.3.2. ACL Chain Support	10
3.3.3. ACL Test Extensions	10
4. acl Module	11
4.1. Features	11
4.2. Types	11
4.3. Groupings	12
4.4. Containers	13
4.4.1. acls Container	13
4.4.2. port-groups Container	13
4.4.3. timerange-groups Container	14
4.4.4. ip-address-groups Container	15
5. acl-ip module	15
5.1. Groupings	15
5.1.1. IP-SOURCE-NETWORK grouping	16
5.1.2. IP-DESTINATION-NETWORK grouping	17
5.1.3. DSCP-OR-TOS Grouping	17
5.1.4. IP-ACE-FILTERS Grouping	18
5.2. augment	20
5.2.1. global-fragments leaf	20
6. acl-mac module	23
6.1. MAC-SOURCE-NETWORK grouping	23
6.2. MAC-DESTINATION-NETWORK grouping	24
6.3. augment	24
7. acl-arp module	24
7.1. augment	24
8. Data Model Structure	25
9. ACL Examples	33
9.1. Configuration Example	33
10. ACL YANG Module	35
11. ACL-IP YANG Module	48
12. ACL-MAC Configuration YANG Module	62
13. ACL-ARP Configuration YANG Module	68
14. COMMON-TYPES YANG Module	71
15. Security Considerations	79
16. Open items from the previous revision	79
17. Acknowledgements	80
18. Normative References	80

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of Access Control Lists (ACLs).

An ACL is an ordered set of rules that is used to filter traffic on a networking device, i.e. to define "firewall rules". Each rule is represented by an Access Control Entry (ACE). An ACE consists of two parts:

Filters with a set of matching criteria that a packet must satisfy for the rule to be applied.

Actions that specifies what to do with the packet when the matching criteria is met, for example, to drop the packet.

There are different types of ACL: MAC ACL, IP ACL, and ARP ACL.

MAC ACLs - MAC ACLs are used to filter traffic using the information in the Layer 2 header of each packet. MAC ACLs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC ACLs to all traffic.

IP ACLs: IP ACLs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets. The device applies IP ACLs only to IP traffic. IP ACL can be IPv4 or IPv6.

ARP ACLs - The device applies ARP ACLs to IP traffic.

Not every device implements every type of ACL. In addition, device implementations may vary greatly in terms of the filter constructs that they support. Therefore, acl YANG Module makes extensive use of the "feature" construct which allows implementations to support those ACL configuration features that lie within their capabilities.

How ACLs are applied in device configuration to interfaces and other components is outside the scope of this model.

2. Definitions and Acronyms

ACE: Access Control Entry

ACL: Access Control List

AFI: Address Field Identifier

ARP: Address Resolution Protocol

CoS: Class of Service

DSCP: Differentiated Services Code Point

ICMP: Internet Control Message Protocol

IGMP: Internet Group Management Protocol

IP: Internet Protocol

IPv4: Internet Protocol version 4

IPv6: Internet Protocol version 6

MAC: Media Access Control

QoS: Quality of Service

TCP: Transmission Control Protocol

ToS: Type of Service

TTL: Time To Live

UDP: User Datagram Protocol

VLAN: Virtual Local Area Network

VRF: Virtual Routing and Forwarding

3. The Design of the ACL Data Model

3.1. Overall Model Structure

The ACL data model consists of five YANG modules. The first module, "acl", defines generic ACL aspects which are common to all ACLs regardless of their type, as well as a set of auxiliary definitions. In effect, the module can be viewed as providing a generic ACL "superclass".

Three other modules, "acl-ip", "acl-mac", and "acl-arp", augment the "acl" module with definitions that are specific to different types of ACLs, specifically, ACLs for IP, MAC, and ARP, respectively. These specifics are for the largest part reflected in the Access Control Entries, that is, the rules which specify the filter criteria that a packet must meet for the rule to be applied, and the actions that are to be taken in case the filter matches. Keeping the modules separate provides for a more modular data model than would be the case if all

types were combined into a single monolithic module.

Finally, module "common-types" defines types that are used in the ACL data model but are not really specific to ACLs. These definitions could potentially be of interest to other models as well; keeping them in a separate module allows to import these definitions independent of the support for ACLs.

3.2. Data hierarchy

The data hierarchy that is defined by the `acl` module is depicted in the following Figure 1, where brackets enclose list keys, "rw" means configuration, "ro" means operational state data, and "?" means optional node. Parentheses enclose choice and case nodes. The structure is a collapsed structure and does not depict all definitions; it is intended to illustrate the overall structure. A fully expanded structure can be found in Data Model Structure Section (Section 8).

```

module: acl
  +--rw acls
    +--rw acl [name]
      +--rw name
      +--rw acl-type
      +--rw enable-capture-global?
      +--rw capture-session-id-global?
      +--rw (enable-match-counter-choices)?
      +--ro match?
    +--rw port-groups
      +--rw port-group [name]
        +--rw name
        +--rw port-group-entry
    +--rw timerange-groups
      +--rw timerange-group [name]
        +--rw name
        +--rw time-range
    +--rw ip-address-groups
      +--rw ip-address-group [name]
        +--rw name
        +--rw afi?
        +--rw ip-address
  
```

Figure 1

Data nodes in the `acl` module are contained under a single container node, "acls". This node contains a list, "acl". Each ACL is

represented by an element in that list and identified by a name that serves as key to the list. Interfaces (which are not part of the model) to which an ACL is applied can then refer to the ACL using that name. Each acl list element has furthermore a type, as indicated through "acl-type". The acl-type determines which types of ACEs can be contained in an ACL. The ACE definitions themselves are provided by the acl-ip, acl-mac, and acl-arp modules, which augment the acl definition in the acl module accordingly. The subsequent data nodes in the acl list allow to configure whether packets that match an ACL should be captured for further analysis. Finally, the list contains an object that maintains a counter of the number of ACL matches.

Auxiliary objects "port-groups", "ip-address-groups", "timerange-groups" are used to define groupings of ports and of IP-addresses as well as schedule information, respectively. They are in effect convenience objects which allow ACEs to refer to groupings and schedules by name, rather than needing to re-specify them in each ACE where they apply.

The following figure depicts how different types of ACEs are inserted into that structure. As indicated earlier, the corresponding definitions are provided in separate modules that augment the acl module. In the data structure, the augmenting module is indicated by the prefix of the corresponding data nodes: "acl-ip", "acl-mac", and "acl-arp", respectively. ACEs for IPv4 and for IPv6 are both defined in the same module, acl-ip. While it would have been possible to define each in its own separate module, it was a design decision to combine them, as they share enough commonality that a separation would have resulted in a considerable amount of definition redundancy.

The figure does not depict objects not pertinent to that structure, such as objects intended to make the definition of port groups ("port-groups"), timeranges ("time-range-groups"), and IP address groups ("ip-address-groups") reusable, as well as objects that are contained in acl list elements, such as "name" and "enable-capture-global".

```

module: acl
  +--rw acls
    +--rw acl [name]
      |   +--rw acl-ip:afi
      |   +--rw acl-ip:ipv6-aces
      |   |   +--rw acl-ip:ipv6-ace [name]
      |   |   +--rw acl-ip:name
      |   |   +--rw (remark-or-ipv6-case)?
      |   |       +--:(remark)

```

```

| | | | +--rw acl-ip:remark
| | | | +---:(ipv6-ace)
| | | | +--rw acl-ip:filters
| | | | |   +- filter parameters
| | | | +--rw acl-ip:actions
| | | | |   +- action parameters
| | | | +-- ro acl-ip:match
|
module: acl
+--rw acls
| +--rw acl [name]
| | +--rw acl-ip:afi
| | +--rw acl-ip:ipv4-aces
| | | +--rw acl-ip:ipv4-ace [name]
| | | +--rw acl-ip:name
| | | +--rw (remark-or-ipv4-ace)?
| | | | +---:(remark)
| | | | | +--rw acl-ip:remark
| | | | +---:(ipv4-ace)
| | | | +--rw acl-ip:filters
| | | | |   +- filter parameters
| | | | +--rw acl-ip:actions
| | | | |   +- action parameters
| | | | +-- ro acl-ip:match
|
module: acl
+--rw acls
| +--rw acl [name]
| | +--rw acl-mac:mac-aces
| | | +--rw acl-mac:mac-ace [name]
| | | +--rw acl-mac:name
| | | +--rw (remark-or-mac-ace)?
| | | | +---:(remark)
| | | | | +--rw acl-mac:remark
| | | | +---:(mac-ace)
| | | | +--rw acl-mac:filters
| | | | |   +- filter parameters
| | | | +--rw acl-mac:actions
| | | | |   +- action parameters
| | | | +-- ro acl-mac:match
|
module: acl
+--rw acls
| +--rw acl [name]
| | +--rw acl-arp:arp-aces
| | | +--rw acl-arp:arp-ace [name]
```

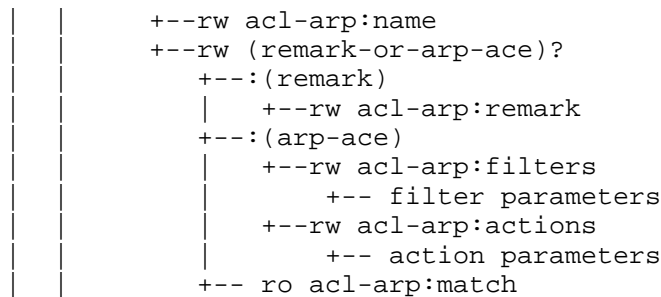



Figure 2

As is evident from Figure 2, the same generic design pattern is reflected in every ACL type. Each ACL contains a list of ACEs, identified by a name by which ACEs in the list are ordered. Each ACE consists either of a remark or of an actual access control rule. Remarks are in effect comment lines inside an ACL that are intended for human or administrator consumption. They are included in the YANG module to maintain consistency with CLI. Access control rules, on the other hand, consist of a left hand side ("filters") that specifies a set of matching criteria and a right hand side ("actions") that specifies the action to take when matching criteria are met. An overview of the full list of filter and parameters is given in Section 8.

Since the design pattern for each ACL type is the same, an alternative design to the YANG modules would have been to extend the "acl" module to include the data nodes up to the level depicted in Figure 2, as the real distinction occurs in the filter and action parameters that occur below it. In that case, however, the corresponding data nodes would have had to contend with more complex conditions. The modules defined here aim at keeping complexity of definitions within the modules as low as possible, at the price of repeating a few data nodes that provide the overall top level structure.

3.3. Other Considerations

3.3.1. Extensibility

If needed, the model can be extended for other types of ACLs in straightforward manner. New types of ACLs can be defined in additional YANG modules that apply the same design patterns much in the same way as in the case of IP, MAC, and ARP ACLs.

3.3.2. ACL Chain Support

ACL chains are used in some application domains. ACL chains are not included in the data model, but could be accommodated in the model through extensions in a straightforward way.

ACL chains work roughly as follows. In an ACL chain, as an alternative to an action, an ACE can point to another ACL. If a packet matches the filter condition, it is subjected to the other ACL. If the other ACL contains an ACE that matches, that action is executed. If there is no match, processing is returned to the first ACL and processing continues with the subsequent ACEs until a match is found. This way, chained ACLs can be considered as a special form of "ACL subroutine".

An example of an ACL chain might be a rule that contains a filter for a specific destination port number in an IP packet, then invokes another ACL that contains a specific set of firewall rules for traffic directed at that particular port. Even though the data model for ACL presented in this document uses a flat list of ACE in each ACL, the actions in the model can be augmented to support ACL chains.

The model can be extended with ACL chains roughly as follows: A new acl-chaining action is introduced, represented as a leaf whose value contains a reference to an ACL as a parameter. For ACLs that are expected to not terminate when no ACE matches, but return processing to the invoking ACL, an optional ACL parameter can be introduced that indicates for chained ACLs which chaining behavior should apply. Below is an example of how the acl-ip model could be extended to support ACL chains for ip-v4:

```
augment "/acl:acls/acl:acl/acl-ip:ipv4-aces" +
  "/acl-ip:ipv4-ace/acl-ip:actions" {

  leaf chain {
    type acl-ref ;
    description "Reference to another ACL name to chain the ACEs";
  }
}
```

3.3.3. ACL Test Extensions

Given the complexity of ACLs in many deployments, debugging ACLs and assessing whether an ACL has the actual desired effect can be a challenge. In order to facilitate those tasks and allow to check whether an ACL has indeed the intended effect, an additional administrative function that allows applications and users to test a

packet against the ACL can be introduced. The function can take the form of an RPC which takes as input parameter a leaf with the reference to the ACL that is to be tested, and a leaf with a packet. The output parameter includes a leaf indicating the action that is taken as a result, as well as a leaf with the reference to the matching ACE.

4. acl Module

Module "acl" is a top container module for all ACLs. It contains a container "acls" with a list "acl" of named ACLs. Modules "acl-ip", "acl-mac", and "acl-arp" augment this list with the objects that are specific to each respective type of ACL. In addition, module "acl" also defines a set of features, reusable types, and reusable groupings.

4.1. Features

When it comes to ACL implementations, a wide range of different capabilities exists across devices. For example, not every device implements every type of ACL. Some devices may support time-based ACLs that are only in effect during specified times, others may not.

In order to accommodate this wide range of capabilities, this data model makes extensive use of the "feature" construct. The defined features allow implementations to declare which capabilities they support, and only support the corresponding portions of the data model.

4.2. Types

The definition of ACLs requires a number of new data types introduced in this data model. Table 1 depicts data types that are unique to ACLs. Table 2 depicts data types that are required by ACLs, but not specific to them, and that may hence be reused by other models. Those data types are defined in module "common-types". For details of each type, please refer to the corresponding typedef descriptions and references in the model.

YANG type	base type
acl-comparator	enumeration
acl-action	enumeration
acl-remark	string
acl-type-ref	identityref
acl-ref	leafref
port-group-ref	leafref
ip-address-group-ref	leafref
time-range-Ref	leafref
weekdays	bits
acl-name-string	string

Table 1

YANG type	base type
cos	uint8
tos	uint8
precedence	uint8
tcp-flag-type	enumeration
ether-type	string
ip-protocol	uint8
igmp-code	uint8
icmp-type	uint32
icmp-code	uint32
vlan-identifier	uint16
time-to-live	uint32

Table 2

4.3. Groupings

The data model defines two groupings, ACE-COMMON and FILTER-COMMON.

- o ACE-COMMON is a collection of nodes that should be added to every ACE list entry. ACE-COMMON contains the actions container and a read-only match leaf. The actions container contains two leaves.
 - * An "action" leaf that specifies what to do with the packet when the matching criteria is met, for example, to drop the packet.
 - * A "log" leaf that indicates whether to create a log entry when an ace filter matches. (Some devices may not support a log

capability. Hence support of this leaf is conditional on declaration of a corresponding feature, as indicated by use of the "if-feature" construct.)

- o FILTER-COMMON is a collection of nodes that should be added to every 'filters' container within each ACE list entry.

4.4. Containers

4.4.1. acls Container

Container "acls" contains a list "acl" of named ACLs. Each list element "acl" contains the following global leaves. The list elements are augmented with additional data nodes defined in modules "acl-arp", "acl-mac", and "acl-ip".

- o name
- o acl-type
- o enable-capture-global
- o capture-session-id-global
- o enable-match-counter-choices: The difference of these two choices is that "enable-match-counter" indicates to collect total match statistics for all aces, whereas "enable-per-entry-match-counter" indicates to collect match statistics for each ACE.
- o match

4.4.2. port-groups Container

Container "port-groups" allows to classifying protocol port into groups. It contains a sequence of "port-group" data nodes. Each "port-group" defines a range of ports and can be referred to by name. Multiple ACEs can refer to the same port group. The following is a Netconf XML example of port-groups and how it is referred to from an ACE.

```
<src-port-group-name>
<port-group-name>port-tunnell</port-group>
</src-port-group-name>

<port-groups>
  <port-group>
    <name>port-tunnell</name>
    <port-group-entry>
      <name>http-proxy</name>
      <port-lower>21</port-lower>
      <port-upper> 22</port-upper>
    </port-group-entry>
  </port-group>
</port-groups>
```

4.4.3. timerange-groups Container

Container "timerange-groups" container contains a list, "timerange-group". Each of its elements defines a sequence of time ranges, "time-range". Each time-range object consists of either a remark (comments for the time range), or of an absolute time for start or end (or both) of the time range, or a periodic time for start or end or both. Object "remark" contains administrator-provided comments for the time-range that will be kept in the device. Like with port groups, the same time-range can be reused by different ACEs. The following is a Netconf XML example of a timerange group that contains a remark and a single time range.

```
<timerange-groups>
  <timerange-group>
    <name>weekday</name>
    <time-range>
      <name>10</name>
      <remark> email server maintenance</remark>
    </time-range>
    <time-range>
      <name>20</name>
      <periodic>
        <weekday>
          Monday Tuesday Wednesday Thursday Friday
        </weekday>
        <start> 21:00:00</start>
        <end> 24:00:00</end>
      </periodic>
    </time-range>
  </timerange-group>
</timerange-groups>
```

4.4.4. ip-address-groups Container

Container "ip-address-groups" contains is list "ip-address-group" of named IP address groups. Each IP address group is a sequence of pairs "ip-address" and "mask", or a pair of "host" and "host-address". Each IP address group can be referred from an ACE by name. The following is a Netconf XML example of an IP address group and how it is referred to from an ACE.

```
<ip-address-groups>

  <ip-address-group>
    <name>Email-Server-IPV4</name>
    <ip-addresses>
      <ip-address>
        <name>10</name>
        <ip-address>128.107.0,0</ip-address>
        <ip-mask>255.255.0.0</ip-mask>
      </ip-address>
      <ip-address>
        <name>20</name>
        <ip-address>139.207.0.0</ip-address>
        <ip-mask>255.255.0.0</ip-mask>
      </ip-address>
    </ip-addresses>
  </ip-address-group>
</ip-address-groups>

<ip-ace>
  <name>100</name>
  <afi>ipv4</afi>
  <actions>permit</actions>
  <filters>
    <ip-source-group>Email-Server-IPV4</ip-source-group>
    <ip-dest-any/>
  </filters>
</ip-ace>
```

5. acl-ip module

acl-ip is the module that defines IP-ACL. It augments acl list in acl module.

5.1. Groupings

5.1.1.1. IP-SOURCE-NETWORK grouping

```

IP-SOURCE-NETWORK
  +--rw (source-address-host-group)?
    +--:(source-ip)
      |   +--rw ip-source-address      inet:ip-address
      |   +--rw ip-source-mask         inet:ip-address
    +--:(ip-source-any)
      |   +--rw ip-source-any          empty
    +--:(source-host)
      |   +--:(ip-src-host-address-or-name)
      |     +--:(ip-source-host-address)
      |       +--rw ip-source-host-address      inet:ip-address
      |     +--:(ip-source-host-name)
      |       +--rw ip-source-host-name         inet:domain-name
    +--:(source-group)
      +--rw ip-source-group?              ip-address-group-ref

```

IP-SOURCE-NETWORK is a reusable grouping. It allows five ways to specify a network: ip with mask, any network, host-name or host address, reference to a predefined ip address group. Here are valid example instances:

- o ip with mask:

```

<ip-source-address>192.168.1.0</ip-source-address>
<ip-source-mask>255.255.255.0</ip-source-mask>

```

- o any network:

```

<ip-source-any/>

```

- o host-name:

```

<ip-source-host-name>switch1</ip-source-host-name>

```

- o host-address:

```

<ip-source-host-address>192.168.1.2</ip-source-host-address>

```

- o reference to a predefined ip address group (Email-Server-IPV4 is defined in Section 4.4.4):


```
<ip-source-group>Email-Server-IPV4</ip-source-group>
```

5.1.2. IP-DESTINATION-NETWORK grouping

```
IP-DESTINATION-NETWORK
  +--rw (dest-address-host-group)?
    +--:(dest-ip)
      |   +--rw ip-dest-address      inet:ip-address
      |   +--rw ip-dest-mask?        inet:ip-address
      +--:(ip-dest-any)
        |   +--rw ip-dest-any      empty
      +--:(dest-host)
        |   +--:(ip-dest-host-address-or-name)
        |     |   +--:(ip-dest-host-address)
        |     |     +--rw ip-dest-host-address      inet:ip-address
        |     +--:(ip-dest-host-name)
        |       +--rw ip-dest-host-name      inet:domain-name
      +--:(group)
        +--rw ip-dest-group?      ip-address-group-ref
```

IP-DESTINATION-ADDRESS is a reusable grouping. Its structure is similar to IP-SOURCE-NETWORK. The reason to have both IP-SOURCE-NETWORK and IP-DESTINATION-NETWORK groupings is to allow "ip-source-address" and "ip-destination-address" leaves to appear in the same container. For example:

```
<filters>
  <ip-source-address>192.168.1.0</ip-source-address>
  <ip-source-mask>255.255.255.0</ip-source-mask>
  <ip-dest-address>any</ip-dest-address>
</filters>
```

5.1.3. DSCP-OR-TOS Grouping

DSCP-OR-TOS grouping defines a choice, "dscp-or-tos". It allows two ways to filter for a QoS packet:

- o dscp: Match packet on DSCP value.
- o tos: Match packet on TOS and precedence value.

The typedef for "tos" and "precedence" is defined in module "common-types", which could be deprecated should IETF define a separate set of definitions.

5.1.4. IP-ACE-FILTERS Grouping

```

IP-ACE-FILTERS
  +--rw protocol?                               c-types:ip-protocol
  +--acl:FILTER-COMMON
  +--rw fragments?                             empty
  +--rw time-range?                            acl:Time-Range-Ref
  +-- (src-ports)?
    | +--rw (port-number-or-range)?
    | | +--:(port-number-range)
    | | | +--rw src-port-lower?             inet:port-number
    | | | +--rw src-port-upper?            inet:port-number
    | | +--:(port-number)
    | | | +--rw src-comparator               comparator
    | | | +--rw src-port?                    inet:port-number
    | +--:(port-group-ref)
    | | +--src-port-group-name
  +-- (des-ports)?
    | +--rw (port-number-or-range)?
    | | +--:(port-number-range)
    | | | +--rw des-port-lower?             inet:port-number
    | | | +--rw des-port-upper?            inet:port-number
    | | +--:(port-number)
    | | | +--rw des-comparator               comparator
    | | | +--rw des-port?                    inet:port-number
    | +--:(by-name)
    | | +-- des-port-group-name
  +--rw icmp-type?                             c-types:icmp-type
  +--rw icmp-code?                             c-types:icmp-type
  +--rw (packet-length-or-range)?
    | +--:(length)
    | | +--rw packet-length-comparator       acl:Comparator
    | | +--rw packet-length                  uint32
    | +--:(range)
    | | +--rw packet-length-upper            uint32
    | | +--rw packet-length-lower            uint32
  +--rw tcp-flag-value?                        c-types:tcp-flag-type
  +--rw tcp-flag-mask?                        c-types:tcp-flag-type
  +--rw tcp-flag-operation?                    enumeration
  +--rw (ttl-value-or-range)?
    | +--:(value)
    | | +--rw ttl-comparator?                acl:acl-comparator
    | | +--rw ttl-value?                      c-types:Time-to-Live
    | +--:(range)
    | | +--rw ttl-value-lower?                c-types:Time-to-Live
    | | +--rw :ttl-value--upper?              c-types:Time-to-Live

```

IP-ACE-FILTERS defines the following leaves that are used by both by IPv4 and IPv6 ACEs:

- o protocol
- o acl:FILTER-COMMON: see Section 4.3
- o fragments: When present, it matches the non-initial fragment.
- o time-range: Enable packet capture on this filter for a timerange-group by name. time-range is Time-Range-Ref type which is a leafref.
- o src-ports choice: Allows the following three ways to define a group of ports.
 - * port-number-range: Use "src-port-lower" and "src-port-upper" leaves to specify a port range. The value of "src-port-lower" has to be less than or equal the value of "src-port-upper".
 - * port-number: Use "comparator" and "src-port" leaves to specify a port range. See Comparator typedef in the model for the possible values the "comparator" leaf.
 - * port range ref: Refer to a named port group that is defined using port-groups. For example:

 <port-group-name>port-tunnell</port-group-name>
- o dest-ports choice: Analogous to "src-ports".
- o packet-length-or-range: Allows two ways to specify packet length range.
 - * case length: Use comparator and a single packet-length to specify the range.
 - * case range: Use packet-length-lower and packet-length-upper to specify a range. The value of packet-length-lower must be lower than or equal to the value of packet-length-upper.
- o icmp-type
- o icmp-code
- o packet-length-or-range choice

- o tcp-flag-value: tcp-flag-value, tcp-flag-mask and tcp-flag-operation allow to match any combination of packet tcp flag values.

The following example is to match the packet tcp flag ack=1, syn=1, and fin=0;

```
<tcp-flag-value> ack syn <tcp-flag-value>
<tcp-flag-mask>ack syn fin</tcp-flag-mask>
<tcp-flag-operation>match-all</tcp-flag-operation>
```

- o tcp-flag-mask
- o tcp-flag-operation
- o ttl-value-or-range

5.2. augment

The module "acl-ip" augments the definition of data node "/acl:acls/acl:acl" with additional leaves and subcomponents.

- o afi
- o ipv6-aces: It contains a list of ipv6-ace. Each ipv6-ace is either a remark or a real access control filters. The case ipv6-ace defines the filters and actions for ipv6-ace. The ace uses filters defined in grouping IP-SOURCE-NETWORK, IP-DESTINATION-NETWORK, IP-ACE-FILTERS, DSCP-OR-TOS. In addition, it also allows filter on igmp-type and flow-label,
- o ipv4-aces: ipv4-ace has similar structure to ipv6-aces.
- o global-fragments

5.2.1. global-fragments leaf

global-fragments is an optional leaf. It has an enumeration value of not-set, permit-all, deny-all. not-set is the default value. When the global-fragments is permit-all or deny-all, it is to permit or deny the implicit ace fragment filter. Here is an example of implicit ace and how the implicit ace is affected when global-fragments is set.

Example 1: The acl configuration from the management interface with global-fragments is absent.

YANG instance of this cli configuration:

```
<acls>
  <acl>
    <name>fragment_test1</name>
    <afi>ipv4</afi>
    <acl-type>ip-acl</acl-type>
    <ip-aces>
      <name>10</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>192.168.5.0</ip-source-address>
        <ip-source-mask>255.255.255.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
      </filters>
    </ip-aces>
    <ip-aces>
      <name>20</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>189.168.0.0</ip-source-address>
        <ip-source-mask>255.255.0.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
        <fragments/>
      </filters>
    </ip-aces>
  </acl>
</acls>
```

By taking all the tags out, the above yang can be express in a summary of cli format like the following:

```
fragment_test1 ip-acl ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
```

The acl configuration together with implicit ace in the device will be:

```
fragment_test1 ip-acl ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
11 permit ip 192.168.5.0 255.255.255.0 any fragment
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
100 deny any any
110 deny any any fragment
```

Notice three lines of configuration. 11, 100 and 110, are implicit.

Example 2: The acl configuration from the management interface with global-fragments

```
<acls>
  <acl>
    <name>fragment_test2</name>
    <acl-type>ip-acl</acl-type>
    <global-fragments>deny-all</global-fragments>
    <afi>ipv4</afi>
    <ip-aces>
      <name>10</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>192.168.5.0</ip-source-address>
        <ip-source-mask>255.255.255.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
      </filters>
    </ip-aces>
    <ip-aces>
      <name>20</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>189.168.0.0</ip-source-address>
        <ip-source-mask>255.255.0.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
        <fragments/>
      </filters>
    </ip-aces>
  </acl>
</acls>
```

The acl configuration in the device with implicit aces. The deny-all void "11 permit ip 1.1.1.1/16 any fragment" ace in previous example.

By taking all the tags out, the above yang can be express in a summary of cli format like the following:

```
fragment_test2 ip-acl ipv4 deny-all
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
```

The acl configuration together with implicit ace in the device will be:

```
fragment_test2 ip-acl ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
100 deny any any
110 deny any any fragment
```

6. acl-mac module

6.1. MAC-SOURCE-NETWORK grouping

```
MAC-SOURCE-NETWORK
+--rw (source-network)?
+--:(source-mac)
|   +--rw source-address          yang:mac-address
|   +--rw source-address-mask     yang:mac-address
+--:(source-any)
|   +--rw source-any              empty
+--:(source-host)
|   +--rw acl-mac:source-host-name inet:host
```

MAC-SOURCE-ADDRESS is a reusable grouping. It allows to express the three kinds network.

any network: use source-any to express any network.

```
<mac-source-kind>any</mac-source-kind>
```

single host network.

```
<source-host-name>my-host</source-host-name>
```

host address with a mask.

```
<source-address>0180.c200.000</source-address>
<source-address-mask>0000.0000.0000</source-address-mask>
```

6.2. MAC-DESTINATION-NETWORK grouping

```

MAC-DESTINATION-NETWORK
  +--rw (dest-network)?
  +--:(address)
  |   +--rw dest-address          yang:mac-address
  |   +--rw dest-address-mask     yang:mac-address
  +--:(dest-any)
  |   +--rw dest-any              empty
  +--:(host)
  |   +--rw acl-mac:dest-host-name      inet:host

```

MAC-DESTINATION-ADDRESS is a reusable grouping similar to MAC-SOURCE-ADDRESS. The reason to have both MAC-SOURCE-ADDRESS and MAC-DESTINATION-ADDRESS grouping is to allow source-address and destination-address leaves appear in the same container. For example:

```

<filters>
  <source-address>0180.c200.000</source-address>
  <source-address-mask>0000.0000.0000</source-address-mask>
  <dest-any/>
</filters>

```

6.3. augment

The module "acl-mac" augments the definition of data node "/acl:acls/acl:acl" with additional leaves and subcomponents. acl-mac has similar structure as acl-ipv4 and acl-ipv6 except the filters are different. mac-ace has filters defined in grouping MAC-SOURCE-NETWORK, MAC-DESTINATION-NETWORK, acl:FILTER-COMMON, ethertype-mask, cos, time-range, and vlan.

7. acl-arp module

7.1. augment

The module "acl-arp" augments the definition of data node "/acl:acls/acl:acl" with additional leaves and subcomponents.


```

augment "/acl:acls/acl:acl"
  +--rw acl-arp:arp-aces
    +--rw acl-arp:arp-ace [name]
      +--rw acl-arp:name          acl:acl-name-string
      +--rw (remark-or-arp-ace)?
        +--:(remark)
        | +--rw acl-arp:remark?    acl:acl-remark
        +--:(arp-ace)
          +--rw filters
            +--rw direction?          enumeration
            +--acl-ip:IP-SOURCE-NETWORK
            +--acl-ip:IP-DESTINATION-NETWORK
            +--acl-mac:MAC-SOURCE-NETWORK
            +--acl-mac:MAC-DESTINATION-NETWORK
            +--acl:FILTER-COMMON
          +acl:ACE-COMMON

```

8. Data Model Structure

The combined data model for ACL configuration is structured as follows. "acl" defines the generic components of an acl system. "acl-ip", "acl-mac", "acl-arp" augment the "acl" module with additional data nodes that are needed for ip, mac, and arp acl respectively.

```

module: acl
  +--rw acls
    +--rw acl [name]
      +--rw name
      +--rw acl-type
      +--rw enable-capture-global?
      +--rw capture-session-id-global?
      +--rw (enable-match-counter-choices)?
        +--:(match)
        | +--rw enable-match-counter?
        +--:(per-entry-match)
        | +--rw enable-per-entry-match-counter?
      +--ro match?
      +--rw acl-ip:afi?
      +--rw acl-ip:ipv6-aces
        +--rw acl-ip:ipv6-ace [name]
          +--rw acl-ip:name          acl:acl-name-string
          +--rw (remark-or-ipv6-case)?
            +--:(remark)
            | +--rw acl-ip:remark?    acl:acl-remark
            +--:(ipv6-ace)
              +--rw acl-ip:filters

```

```

+--rw (source-address-host-group)
|   +--:(source-ip)
|   |   +--rw acl-ip:ip-source-address
|   |   +--rw acl-ip:ip-source-mask
|   +--:(ip-source-any)
|   |   +--rw acl-ip:ip-source-any?
|   +--:(source-host)
|   |   +--rw (ip-src-address-or-name)
|   |   |   +--:(ip-source-host-address)
|   |   |   |   +--rw acl-ip:ip-source-host-address?
|   |   |   +--:(ip-source-host-name)
|   |   |   |   +--rw acl-ip:ip-source-host-name?
|   +--:(source-group)
|   |   +--rw acl-ip:ip-source-group?
+--rw (dest-address-host-group)
|   +--:(dest-ip)
|   |   +--rw acl-ip:ip-dest-address
|   |   +--rw acl-ip:ip-dest-mask
|   +--:(ip-dest-any)
|   |   +--rw acl-ip:ip-dest-any?
|   +--:(dest-host)
|   |   +--rw (ip-dest-address-or-name)
|   |   |   +--:(ip-dest-host-address)
|   |   |   |   +--rw acl-ip:ip-dest-host-address?
|   |   |   +--:(ip-dest-host-name)
|   |   |   |   +--rw acl-ip:ip-dest-host-name?
|   +--:(dest-group)
|   |   +--rw acl-ip:ip-dest-group?
+--rw acl-ip:protocol?
+--rw acl-ip:enable-capture?
+--rw acl-ip:capture-session-id?
+--rw acl-ip:fragments?
+--rw acl-ip:time-range?
+--rw (src-ports)?
|   +--:(port-number-range)
|   |   +--rw acl-ip:src-port-lower
|   |   +--rw acl-ip:src-port-upper
|   +--:(port-number)
|   |   +--rw acl-ip:src-comparator
|   |   +--rw acl-ip:src-port
|   +--:(port-group-ref)
|   |   +--rw acl-ip:src-port-group-name
+--rw (dest-ports)?
|   +--:(port-number-range)
|   |   +--rw acl-ip:des-port-lower
|   |   +--rw acl-ip:des-port-upper
|   +--:(port-number)
|   |   +--rw acl-ip:des-comparator

```

```

| | | | +--rw acl-ip:des-port
| | | | +---:(port-group-ref)
| | | | +--rw acl-ip:des-port-group-name
+--rw acl-ip:icmp-type?
+--rw acl-ip:icmp-code?
+--rw (packet-length-or-range)?
| | | | +---:(length)
| | | | | +--rw acl-ip:packet-length-comparator
| | | | | +--rw acl-ip:packet-length
+---:(range)
| | | | +--rw acl-ip:packet-length-upper
| | | | +--rw acl-ip:packet-length-lower
+--rw acl-ip:tcp-flag-value?
+--rw acl-ip:tcp-flag-mask?
+--rw acl-ip:tcp-flag-operation?
+--rw (ttl-value-or-range)?
| | | | +---:(value)
| | | | | +--rw acl-ip:ttl-comparator?
| | | | | +--rw acl-ip:ttl-value?
+---:(range)
| | | | +--rw acl-ip:ttl-value-lower?
| | | | +--rw acl-ip:ttl-value-upper?
+--rw (dscp-or-tos)?
| | | | +---:(dscp)
| | | | | +--rw acl-ip:dscp?
+---:(tos)
| | | | +--rw acl-ip:tos?
| | | | +--rw acl-ip:precedence?
+--rw acl-ip:igmp-type?
+--rw acl-ip:flow-label?
+--rw acl-ip:actions
| | | | +--rw acl-ip:action
| | | | +--rw acl-ip:log?
+--ro acl-ip:match?
+--rw acl-ip:ipv4-aces
| | | | +--rw acl-ip:ipv4-ace [name]
| | | | | +--rw acl-ip:name          acl:acl-name-string
| | | | | +--rw (remark-or-ipv4-ace)?
| | | | | +---:(remark)
| | | | | | +--rw acl-ip:remark?      acl:acl-remark
+---:(ipv4-ace)
| | | | +--rw acl-ip:filters
| | | | | +--rw (source-address-host-group)
| | | | | | +---:(source-ip)
| | | | | | | +--rw acl-ip:ip-source-address
| | | | | | | +--rw acl-ip:ip-source-mask
| | | | | | +---:(ip-source-any)
| | | | | | +--rw acl-ip:ip-source-any?

```

```

+---:(source-host)
|   +---rw (ip-src-address-or-name)
|       +---:(ip-source-host-address)
|           | +---rw acl-ip:ip-source-host-address?
|           +---:(ip-source-host-name)
|               +---rw acl-ip:ip-source-host-name?
+---:(source-group)
|   +---rw acl-ip:ip-source-group?
+---rw (dest-address-host-group)
|   +---:(dest-ip)
|       +---rw acl-ip:ip-dest-address
|       +---rw acl-ip:ip-dest-mask
+---:(ip-dest-any)
|   +---rw acl-ip:ip-dest-any?
+---:(dest-host)
|   +---rw (ip-dest-address-or-name)
|       +---:(ip-dest-host-address)
|           | +---rw acl-ip:ip-dest-host-address?
|           +---:(ip-dest-host-name)
|               +---rw acl-ip:ip-dest-host-name?
+---:(dest-group)
|   +---rw acl-ip:ip-dest-group?
+---rw acl-ip:protocol?
+---rw acl-ip:enable-capture?
+---rw acl-ip:capture-session-id?
+---rw acl-ip:fragments?
+---rw acl-ip:time-range?
+---rw (src-ports)?
|   +---:(port-number-range)
|       +---rw acl-ip:src-port-lower
|       +---rw acl-ip:src-port-upper
+---:(port-number)
|   +---rw acl-ip:src-comparator
|   +---rw acl-ip:src-port
+---:(port-group-ref)
|   +---rw acl-ip:src-port-group-name
+---rw (dest-ports)?
|   +---:(port-number-range)
|       +---rw acl-ip:des-port-lower
|       +---rw acl-ip:des-port-upper
+---:(port-number)
|   +---rw acl-ip:des-comparator
|   +---rw acl-ip:des-port
+---:(port-group-ref)
|   +---rw acl-ip:des-port-group-name
+---rw acl-ip:icmp-type?
+---rw acl-ip:icmp-code?
+---rw (packet-length-or-range)?

```

```

| | | | | +--:(length)
| | | | | | +--rw acl-ip:packet-length-comparator
| | | | | | +--rw acl-ip:packet-length
| | | | | +--:(range)
| | | | | | +--rw acl-ip:packet-length-upper
| | | | | | +--rw acl-ip:packet-length-lower
| | | | +--rw acl-ip:tcp-flag-value?
| | | | +--rw acl-ip:tcp-flag-mask?
| | | | +--rw acl-ip:tcp-flag-operation?
| | | | +--rw (ttl-value-or-range)?
| | | | | +--:(value)
| | | | | | +--rw acl-ip:ttl-comparator?
| | | | | | +--rw acl-ip:ttl-value?
| | | | | +--:(range)
| | | | | | +--rw acl-ip:ttl-value-lower?
| | | | | | +--rw acl-ip:ttl-value-upper?
| | | | +--rw (dscp-or-tos)?
| | | | | +--:(dscp)
| | | | | | +--rw acl-ip:dscp?
| | | | | +--:(tos)
| | | | | | +--rw acl-ip:tos?
| | | | | | +--rw acl-ip:precedence?
| | | | +--rw acl-ip:actions
| | | | | +--rw acl-ip:action      acl:acl-action
| | | | | +--rw acl-ip:log?      empty
| | | | +--ro acl-ip:match?      yang:counter64
+--rw acl-ip:global-fragments?   enumeration
+--rw acl-mac:mac-aces
| +--rw acl-mac:mac-ace [name]
| | +--rw acl-mac:name          acl:acl-name-string
| | +--rw (remark-or-mac-ace)?
| | | +--:(remark)
| | | | +--rw acl-mac:remark?    acl:acl-remark
| | | +--:(mac-ace)
| | +--rw acl-mac:filters
| | | +--rw (source-network)
| | | | +--:(source-mac)
| | | | | +--rw acl-mac:source-address
| | | | | +--rw acl-mac:source-address-mask
| | | | +--:(source-any)
| | | | | +--rw acl-mac:source-any?
| | | | +--:(source-host)
| | | | | +--rw (src-address-or-name)
| | | | | | +--:(source-host-address)
| | | | | | | +--rw acl-mac:source-host-address?
| | | | | | +--:(source-host-name)
| | | | | | | +--rw acl-mac:source-host-name?
| | | +--rw (dest-network)

```

```

| | | | | +---:(dest-mac)
| | | | | | +---rw acl-mac:dest-address
| | | | | | +---rw acl-mac:dest-address-mask
| | | | | +---:(dest-any)
| | | | | | +---rw acl-mac:dest-any?
| | | | | +---:(dest-host)
| | | | | | +---rw (dest-address-or-name)
| | | | | | | +---:(dest-host-address)
| | | | | | | | +---rw acl-mac:dest-host-address?
| | | | | | | +---:(dest-host-name)
| | | | | | | | +---rw acl-mac:dest-host-name?
| | | | | +---rw acl-mac:ethertype?
| | | | | +---rw acl-mac:ethertype-mask?
| | | | | +---rw acl-mac:cos?
| | | | | +---rw acl-mac:time-range?
| | | | | +---rw acl-mac:vlan?
| | | | | +---rw acl-mac:enable-capture?
| | | | | +---rw acl-mac:capture-session-id?
| | | | +---rw acl-mac:actions
| | | | | +---rw acl-mac:action
| | | | | +---rw acl-mac:log?
| | | | +---ro acl-mac:match?
+---rw acl-arp:arp-aces
| +---rw acl-arp:arp-ace [name]
| | +---rw acl-arp:name
| | +---rw (remark-or-arp-ace)?
| | | +---:(remark)
| | | | +---rw acl-arp:remark?
| | | +---:(arp-ace)
| | +---rw acl-arp:filters
| | | +---rw acl-arp:direction?
| | | +---rw (source-address-host-group)
| | | | +---:(source-ip)
| | | | | +---rw acl-arp:ip-source-address
| | | | | +---rw acl-arp:ip-source-mask
| | | | +---:(ip-source-any)
| | | | | +---rw acl-arp:ip-source-any?
| | | | +---:(source-host)
| | | | | +---rw (ip-src-address-or-name)
| | | | | | +---:(ip-source-host-address)
| | | | | | | +---rw acl-arp:ip-source-host-address?
| | | | | | +---:(ip-source-host-name)
| | | | | | | +---rw acl-arp:ip-source-host-name?
| | | | +---:(source-group)
| | | | | +---rw acl-arp:ip-source-group?
+---rw (dest-address-host-group)
| | +---:(dest-ip)
| | | +---rw acl-arp:ip-dest-address

```

```

| | | | | +--rw acl-arp:ip-dest-mask
| | | | | +---:(ip-dest-any)
| | | | | | +--rw acl-arp:ip-dest-any?
| | | | | +---:(dest-host)
| | | | | | +--rw (ip-dest-address-or-name)
| | | | | | | +---:(ip-dest-host-address)
| | | | | | | | +--rw acl-arp:ip-dest-host-address?
| | | | | | | | +---:(ip-dest-host-name)
| | | | | | | | +--rw acl-arp:ip-dest-host-name?
| | | | | +---:(dest-group)
| | | | | | +--rw acl-arp:ip-dest-group?
+--rw (source-network)
| | | | | +---:(source-mac)
| | | | | | +--rw acl-arp:source-address
| | | | | | +--rw acl-arp:source-address-mask
+---:(source-any)
| | | | | | +--rw acl-arp:source-any?
+---:(source-host)
| | | | | | +--rw (src-address-or-name)
| | | | | | | +---:(source-host-address)
| | | | | | | | +--rw acl-arp:source-host-address?
| | | | | | | | +---:(source-host-name)
| | | | | | | | +--rw acl-arp:source-host-name?
+--rw (dest-network)
| | | | | +---:(dest-mac)
| | | | | | +--rw acl-arp:dest-address
| | | | | | +--rw acl-arp:dest-address-mask
+---:(dest-any)
| | | | | | +--rw acl-arp:dest-any?
+---:(dest-host)
| | | | | | +--rw (dest-address-or-name)
| | | | | | | +---:(dest-host-address)
| | | | | | | | +--rw acl-arp:dest-host-address?
| | | | | | | | +---:(dest-host-name)
| | | | | | | | +--rw acl-arp:dest-host-name?
+--rw acl-arp:enable-capture?
+--rw acl-arp:capture-session-id?
+--rw acl-arp:actions
| | | | | +--rw acl-arp:action
| | | | | +--rw acl-arp:log?
+---ro acl-arp:match?
+--rw port-groups
| | | | | +--rw port-group [name]
| | | | | | +--rw name
| | | | | | +--rw port-group-entry [name]
| | | | | | | +--rw name
| | | | | | | +--rw (port-number-or-range)?
| | | | | | | | +---:(port-number-range)

```

```

|         |   +--rw port-lower
|         |   +--rw port-upper
|         +---:(port-number)
|         |   +--rw comparator
|         |   +--rw port
+--rw timerange-groups
|   +--rw timerange-group [name]
|   |   +--rw name
|   |   +--rw time-range [name]
|   |   |   +--rw name
|   |   |   +--rw remark?
|   |   |   +--rw (range-type)?
|   |   |   |   +---:(absolute)
|   |   |   |   |   +--rw absolute
|   |   |   |   |   |   +--rw start?
|   |   |   |   |   |   +--rw end?
|   |   |   +---:(periodic)
|   |   |   |   +--rw periodic
|   |   |   |   |   +--rw weekdays?
|   |   |   |   |   +--rw start?
|   |   |   |   |   +--rw end?
+--rw ip-address-groups
|   +--rw ip-address-group [name]
|   |   +--rw name
|   |   +--rw afi?
|   |   +--rw ip-address [name]
|   |   |   +--rw name
|   |   |   +--rw (ip-network-kind)
|   |   |   |   +---:(ip)
|   |   |   |   |   +--rw ip-address?
|   |   |   |   |   +--rw ip-mask
|   |   |   +---:(ip-any)
|   |   |   |   +--rw ip-any?
|   |   |   +---:(host)
|   |   |   |   +--rw (address-or-name)
|   |   |   |   |   +---:(ip-host-address)
|   |   |   |   |   |   +--rw ip-host-address?
|   |   |   |   |   +---:(ip-host-name)
|   |   |   |   |   |   +--rw ip-host-name?
module: acl-ip
module: acl-mac
module: acl-arp

```

Figure 3

9. ACL Examples

9.1. Configuration Example

Requirement: Denies TELNET traffic from 14.3.6.234 bound for host 6.5.4.1 from leaving. Denies all TFTP traffic bound for TFTP servers. Permits all other IP traffic.

In order to achieve the requirement, an name access control list is needed. In the acl, we need three aces. The acl and aces can be described in CLI: as the following:

```
access-list ip iacl
```

```
deny tcp 14.3.6.234 0.0.0.0 host 6.5.4.1 eq 23
deny udp any any eq tftp
permit ip any any
```

Here is the example acl configuration xml:

```
<rpc message-id="101"
  xmlns:nc="urn:cisco:params:xml:ns:yang:acl:1.0"
  xmlns:acl-ip="urn:cisco:params:xml:ns:yang:acl-ip"
  // replace with IANA namespace when assigned
<edit-config>
  <target>
    <running/>
  </target>
  <config>
    <top xmlns="http://example.com/schema/1.2/config">

      <acls>
        <acl >
          <name>sample-ip-acl</name>
          <acl-type>ip-acl</acl-type>
          <enable-match-counter>false</enable-match-counter>
          <acl-ip:afi>ipv4</acl-ip:afi>
          <acl-ip:ipv4-aces>

            <acl-ip:ipv4-ace>
              <acl-ip:name>acl10</acl-ip:name>
              <acl-ip:filters>
                <acl-ip:protocol>6</acl-ip:protocol>
                <acl-ip:ip-source-address>
                  14.3.6.234
                </acl-ip:ip-source-address>
                <acl-ip:ip-source-mask>0.0.0.0</acl-ip:ip-source-mask>
                <acl-ip:ip-dest-host-address>
```

```
        6.5.4.1
        </acl-ip:ip-dest-host-address>
        <acl-ip:des-comparator>eq</acl-ip:des-comparator>
        <acl-ip:des-port>23</acl-ip:des-port>
    </acl-ip:filters>
    <acl-ip:actions>
        <acl-ip:action>deny</acl-ip:action>
    </acl-ip:actions>
</acl-ip:ipv4-ace>

<acl-ip:ipv4-ace>
    <acl-ip:name>ace20</acl-ip:name>
    <acl-ip:filters>
        <acl-ip:protocol>17</acl-ip:protocol>
        <acl-ip:ip-source-any/>
        <acl-ip:ip-dest-any/>
        <acl-ip:des-comparator>eq</acl-ip:des-comparator>
        <acl-ip:des-port>69</acl-ip:des-port>
    </acl-ip:filters>
    <acl-ip:actions>
        <acl-ip:action>deny</acl-ip:action>
    </acl-ip:actions>
</acl-ip:ipv4-ace>

<acl-ip:ipv4-ace>
    <acl-ip:name>ace30</acl-ip:name>
    <acl-ip:filters>
        <acl-ip:ip-source-any/>
        <acl-ip:ip-dest-any/>
    </acl-ip:filters>
    <acl-ip:actions>
        <acl-ip:action>permit</acl-ip:action>
    </acl-ip:actions>
</acl-ip:ipv4-ace>
</acl-ip:ipv4-aces>

</acl>
</acls>

</top>
</config>
</edit-config>
</rpc>
```

10. ACL YANG Module

This module imports type definitions from [RFC6021].

```
<CODE BEGINS> file "acl@2012-10-12.yang"
module acl {
  namespace "urn:cisco:params:xml:ns:yang:acl";
  // replace with IANA namespace when assigned
  prefix acl;

  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: netmod@ietf.org

    WG Chair: David Kessens
    david.kessens@nsn.com

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

    Editor: Lisa Huang
    yihuan@cisco.com

    Editor: Alexander Clemm
    alex@cisco.com

    Editor: Andy Bierman
    andy@yumaworks.com";

  description
    "This YANG module defines a component that describing the
    configuration of Access Control Lists (ACLs).

    An ACL is an ordered set of rules and actions used to filter
    traffic. Each set of rules and actions is represented
    as an Access Control Entries (ACE). Each ACE is evaluated
    sequentially. When the rule matches then action for that
```

rule is applied to the packet.

There are three types of ACL.

IP ACLs - IP ACLs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets.

The device applies IP ACLs only to IP traffic. IP ACL can be IPv4 or IPv6.

MAC ACLs - MAC ACLs are used to filter traffic using the information in the Layer 2 header of each packet.

MAC ACLs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC ACLs to all traffic.

ARP ACLs - The device applies ARP ACLs to IP traffic.

This module should be used with `acl-ip`, `acl-arp`, or `acl-mac` depends on what feature the device supports.

This YANG module also includes auxiliary definitions that are needed in conjunction with configuration of ACLs, such as reusable containers and references for ports and IP.

Terms and Acronyms

ACE (`ace`): Access Control Entry

ACL (`acl`): Access Control List

AFI (`afi`): Authority and Format Identifier (Address Field Identifier)

ARP (`arp`): Address Resolution Protocol

IP (`ip`): Internet Protocol

IPv4 (`ipv4`): Internet Protocol Version 4

IPv6 (`ipv6`): Internet Protocol Version 6

MAC: Media Access Control

TCP (`tcp`): Transmission Control Protocol

TTL (`ttl`): Time to Live

VLAN (`vlan`): Virtual Local Area Network

";

```
reference
  "Access List Commands on Cisco IOS XR Software,
  Cisco Nexus 7000 Series NX-OS Security Configuration Guide,
  Catalyst 6500 Release 12.2SX Software Configuration Guide,
  ACL TCP Flags Filtering";

revision 2012-10-12 {
  description "Initial revision. ";
}

/* Features */

feature capture-session-id {
  if-feature packet-capture;
  description
    "The ability to configure ACL capture in order to
    selectively monitor traffic on an interface or VLAN.
    When the capture option for an ACL rule
    is enabled, packets that match this rule are
    either forwarded or dropped based on the specified permit
    or deny action and may also be copied to an alternate
    destination port for further analysis.
    An ACL rule with the capture option can be applied
    as follows:
      On a VLAN
      In the ingress direction on all interfaces
      In the egress direction on all Layer 3 interfaces
    The statistics data for the capture-session are capture
    in the device where the ACL rule applied to.";
}

feature host-by-name {
  description
    "The capability to reference a host by DNS name.";
}

feature ip-address-groups {
  description
    "The ability to define named groups for lists of
    ip addresses. ";
}

feature logging {
  description
    "The ability to log messages upon the matching of ACLs.";
}

feature match-counter {
```

```
        description
            "The ability to maintain global or local match statistics
            for each ACL rules.";
    }

    feature packet-capture {
        description "The ability to capture packets that
            match the filter.";
    }

    feature packet-length {
        description "The ability to filter packets by packet length";
    }

    feature port-groups {
        description
            "The ability to define named groups for lists of ports. ";
    }

    /* Identities */

    identity acl-type {
        description "Base acl type for all ACL type identifiers.";
    }

    /* Types */

    typedef acl-comparator {
        description "A data type used to express comparator string";
        type enumeration {
            enum "eq" {
                value 0;
                description "match only equal to any giving number.";
            }
            enum "gt" {
                value 1;
                description
                    "match only greater than any giving number.";
            }
            enum "lt" {
                value 2;
                description
                    "match only lower than any giving number.";
            }
            enum "neq" {
                value 3;
            }
        }
    }
```

```
        description
            "match only not equal to any giving number";
    }
}

typedef acl-action {
    description "An enumeration data type to express acl
        action when match.";
    type enumeration {
        enum deny {
            description "Apply deny action to the traffic";
        }
        enum permit {
            description "Apply permit action to the traffic";
        }
    }
}

typedef acl-remark {
    type string {
        length "0..100";
    }
    description
        "A remark is a comment that can be
        associated with an ACE in order to make
        the access list easier for the network
        administrator to understand.
        It is retained to facilitate
        co-existence with CLI.";
}

typedef acl-type-ref {
    description
        "This type is used to refer to an Access Control List
        (ACL) type";
    type identityref {
        base "acl-type";
    }
}

typedef acl-ref {
    description "This type refers to an ACL.";
    type leafref {
        path "/acl:acls/acl:acl/acl:name";
    }
}
```

```
}

typedef port-group-ref {
  description
    "This type is used to refer to a Portgroup object.";
  type leafref {
    path "/acls/port-groups/port-group/name";
  }
}

typedef ip-address-group-ref {
  description
    "This type is used to refer to a time range object.";
  type leafref {
    path "/acls/ip-address-groups/ip-address-group/name";
  }
}

typedef time-range-ref {
  description
    "This type is used to refer to a time range object.";
  type leafref {
    path "/acls/timerange-groups/timerange-group/name";
  }
}

typedef weekdays {
  type bits {
    bit Sunday {
      position 0;
    }
    bit Monday {
      position 1;
    }
    bit Tuesday {
      position 2;
    }
    bit Wednesday {
      position 3;
    }
    bit Thursday {
      position 4;
    }
    bit Friday {
      position 5;
    }
  }
}
```



```
        bit Saturday {
            position 6;
        }
    }
}

typedef acl-name-string {
    type string {
        length "1 .. 64";
    }
}

/* Groupings */

grouping ACE-COMMON {
    description
        "A collection of nodes that should be added to
        every ACE list entry";

    container actions {
        leaf action {
            type acl:acl-action;
            mandatory true;
            description "Permit/deny action.";
        }

        leaf log {
            if-feature acl:logging;
            type empty;
            description
                "Causes an informational logging message about the
                packet that matches the entry to be sent to the
                console.";
        }
    }

    leaf match {
        if-feature acl:match-counter;
        config false;
        type yang:counter64;
        description
            "The total packet that have matched for the
            particular ACE";
    }
}

grouping FILTER-COMMON {
    description
```

```
"A collection of nodes that should be added to
every 'filters' container within each
ACE list entry";

leaf enable-capture {
  if-feature acl:packet-capture;
  type boolean;
  description
    "Enable packet capture on this filter
    for this session.";
}

leaf capture-session-id {
  if-feature acl:capture-session-id;
  when "../enable-capture = 'true'";
  type uint32 {
    range "1..48";
  }
  description
    "Enable packet capture on this filter
    for this session id.";
}
}

/* Data Nodes */

container acls {
  description
    "This is the top container that contains a list of
    named ACL and reusable acl object groups.";
  list acl {
    key name;
    leaf name {
      description "ACL/access group name.";
      type acl-name-string;
    }

    leaf acl-type {
      type acl-type-ref;
      description "Type of ACL";
      mandatory true;
    }

    leaf enable-capture-global {
      if-feature packet-capture;
      type boolean;
      description "Enable packet capture on this filter
        for this session. Session ID range is 1 to 48";
      default "false";
    }
  }
}
```

```
    }
    leaf capture-session-id-global {
      if-feature capture-session-id;
      when "../enable-capture-global = 'true'";
      type uint32 {
        range "1..48";
      }
      description "Enable packet capture on this filter
        for this session. Session ID range is 1 to 48";
    }
    choice enable-match-counter-choices {
      if-feature match-counter;
      case match {
        leaf enable-match-counter {
          type boolean;
          description
            "Enable to collect statistics for the ACL";
          default false;
        }
      }
      case per-entry-match {
        leaf enable-per-entry-match-counter {
          type boolean;
          description "Enable to collect match
            statistics for each ACL entry(ACE).";
          default false;
        }
      }
    }
  }

  leaf match {
    if-feature match-counter;
    config false;
    type yang:counter64;
    description
      "The total packet that have matched for the
        particular access list";
  }
}

container port-groups {
  if-feature port-groups;
  list port-group {
    key "name";
    leaf name {
      type acl-name-string;
    }
  }
}
```

```
list port-group-entry {
  key "name";
  ordered-by user;
  leaf name {
    type acl-name-string;
  }
  //unique "comparator port-number
  //port-lower port-upper";

  choice port-number-or-range {
    case port-number-range {
      description
        "Port group includes all ports between
        port-lowerand port-upper (including those)";
      leaf port-lower {
        type inet:port-number;
        description "Lower Port number.";
        mandatory true;
      }
      leaf port-upper {
        type inet:port-number;
        description "Upper Port number.";
        mandatory true;
        must "../port-lower <= ../port-upper";
      }
    }
    case port-number {
      description
        "Port group includes all ports that are greater
        than, greater or equal, less than, less or
        equal, or not equal the port, per the
        indicated comparator.
        It is possible for the port group to be empty
        (for example, in case a port group that
        is less than the minimum port number is
        specified).";
      leaf comparator {
        type acl-comparator;
        mandatory true;
      }
      leaf port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
      }
    }
  } // choice port-number-or-range
} // list port-group-entry
```

```
    } // list port-group
  } // container port-groups

  container timerange-groups {
    description "Define time range entries to restrict
      the access. The time range is identified by a name
      and then referenced by a function, so that those
      time restrictions are imposed on the function itself.";
    list timerange-group {
      key "name";
      leaf name {
        type acl-name-string;
      }
      list time-range {
        key "name";
        ordered-by user;
        leaf name {
          type acl-name-string;
        }

        leaf remark {
          type acl-remark;
        }
      }

      choice range-type {
        // absolute or periodic time range
        container absolute {
          description
            "Absolute time and date that
              the associated function starts
              going into effect.";
          leaf start {
            type yang:date-and-time;
            description
              "Absolute start time and date";
          }
          leaf end {
            type yang:date-and-time;
            description "Absolute end time and date";
          }
        }
        container periodic {
          description
            "To specify a periodic time and date.";
          leaf weekdays {
            type weekdays;
          }
          leaf start {
```

```
        type yang:timestamp;
        description "Start time";
    }
    leaf end {
        type yang:timestamp;
        description "End time";
    }
} // choice range-type
} // list time-range
} // list timerange-group
} // container timerange-groups

container ip-address-groups {
    if-feature ip-address-groups;
    description
        "This contains a list of named ip address group. Each
        group defines a range of address and mask pair.";
    list ip-address-group {
        key "name";
        leaf name {
            type acl-name-string;
        }
        leaf afi {
            default "ipv4";
            type inet:ip-version;
            description "Address Field Identifier (AFI).";
        }
        list ip-address {
            key "name";
            ordered-by user;
            leaf name {
                type acl-name-string;
            }
            //unique "ip-address ip-mask";
            //unique "ip-host-address";

            grouping IP-HOST {
                description
                    "Choice within a case not allowed so need
                    this grouping.";
                choice address-or-name {
                    mandatory true;
                    leaf ip-host-address {
                        type inet:ip-address;
                    }
                    leaf ip-host-name {
                        if-feature acl:host-by-name;
                    }
                }
            }
        }
    }
}
```

```

        type inet:domain-name;
    }
}

choice ip-network-kind {
    mandatory true;

    case ip {
        leaf ip-address {
            type inet:ip-address;
        }
        leaf ip-mask {
            type inet:ip-prefix;
            mandatory true;
        }
    }
    leaf ip-any {
        type empty;
        description "To express Any network or address.
            Use the any keyword as an abbreviation
            for an address and a mask of 0.0.0.0
            255.255.255.255. For example:
            0.0.0.0/255.255.255.255 means 'any'";
    }
    case host {
        description
            "Use the host address combination as an
            abbreviation for an address and wildcard
            of address 0.0.0.0";

        uses IP-HOST;
    }
    // case group not allowed here!
}

    } // list ip-address
    } // list ip-address-group
    } // container ip-address-groups
} // container acls

}
<CODE ENDS>

```

11. ACL-IP YANG Module

This module imports type definitions from [RFC6021] and common-types yang defined with acl model.

```
<CODE BEGINS> file "acl-ip@2012-10-12.yang"
module acl-ip {
  namespace "urn:cisco:params:xml:ns:yang:acl-ip";
  // replace with IANA namespace when assigned
  prefix acl-ip;

  import acl {
    prefix acl;
  }
  import ietf-inet-types {
    prefix "inet";
  }
  import common-types {
    prefix "c-types";
  }

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/netmod/
    WG List: netmod@ietf.org

    WG Chair: David Kessens
    david.kessens@nsn.com

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

    Editor: Lisa Huang
    yihuan@cisco.com

    Editor: Alexander Clemm
    alex@cisco.com

    Editor: Andy Bierman
    andy@yumaworks.com";

  description
    "This YANG module augments the 'acl' module with configuration
    and operational data for IPv4 and IPv6 access control list.

    An ACL is an ordered set of rules and actions used to filter
```


traffic.

Each set of rules and actions is represented as an Access Control Entries (ACE). Each ACE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

IP ACLs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets.

The device applies IP ACLs only to IP traffic. IP ACL can be IPv4 or IPv6.

Terms and Acronyms

ACE (ace): Access Control Entry

ACL (acl): Access Control List

AFI (afi): Authority and Format Identifier (Address Field Identifier)

DSCP (dscp): Differentiated Services Code Point

ICMP (icmp): Internet Control Message Protocol

IGMP (igmp): Internet Group Management Protocol

IP (ip): Internet Protocol

IPv4 (ipv4): Internet Protocol Version 4

IPv6 (ipv6): Internet Protocol Version 6

QoS: Quality of Service

TCP (tcp): Transmission Control Protocol

ToS (tos): Type of Service

TTL (ttl): Time to Live

UDP (udp): User Datagram Protocol

VLAN (vlan): Virtual Local Area Network

VRF(vrf) : Virtual Routing and Forwarding

";

reference

"Access List Commands on Cisco IOS XR Software,

Cisco Nexus 7000 Series NX-OS Security Configuration Guide,
Catalyst 6500 Release 12.2SX Software Configuration Guide,
ACL TCP Flags Filtering";

```
revision 2012-10-12 {
  description "Initial revision. ";
}

/* Features */

feature time-to-live {
  description "The ability to filter packets based on their
    time-to-live (TTL) value (0 to 255)";
  reference "ACL Support for Filtering on TTL Value";
}

feature flow-label {
  description
    "The ability to filter packets based on flow lable.
    The 20-bit Flow Label field in the IPv6 header
    is used by a source to label packets
    of a flow. This is an IPv6 ACEs option.";
  reference "RFC 3697 IPv6 Flow Label Specification";
}

/* Identities */

identity ip-acl {
  base "acl:acl-type";
  description "layer 3 ACL type";
}

/* Groupings */

grouping IP-SOURCE-NETWORK {
  description "Reusable IP address and mask pair.";

  grouping IP-SOURCE-HOST {
    description
      "Choice within a case not allowed so need
      this grouping.";
    choice ip-src-address-or-name {
      mandatory true;
      leaf ip-source-host-address {
        type inet:ip-address;
      }
      leaf ip-source-host-name {
```

```

        if-feature acl:host-by-name;
        type inet:domain-name;
    }
}

choice source-address-host-group {
    mandatory true;
    case source-ip {
        description "Used with address and mask couple
            to express network.";

        leaf ip-source-address {
            type inet:ip-address;
            mandatory true;
        }
        leaf ip-source-mask {
            type inet:ip-address;
            mandatory true;
        }
    }
    leaf ip-source-any {
        type empty;
        description "To express Any network or address.
            Use the any keyword as an abbreviation
            for an address and a mask of 0.0.0.0
            255.255.255.255. For example:
            0.0.0.0/255.255.255.255 means 'any'";
    }
    case source-host {
        description "Used with host address to express a
            single host
            Use the host address(or name)
            combination is the same as an address
            and mask of address 0.0.0.0.
            For example: '10.1.1.2/0.0.0.0' is the same
            as 'host 10.1.1.2'";
        uses IP-SOURCE-HOST;
    }
    case source-group {
        if-feature acl:ip-address-groups;
        leaf ip-source-group {
            type acl:ip-address-group-ref;
        }
    }
}

```

```
grouping IP-DESTINATION-NETWORK {
  description
    "Reusable IP address and mask pair for destination.";

  grouping IP-DESTINATION-HOST {
    description
      "Choice within a case not allowed so need
      this grouping.";
    choice ip-dest-address-or-name {
      mandatory true;
      leaf ip-dest-host-address {
        type inet:ip-address;
      }
      leaf ip-dest-host-name {
        if-feature acl:host-by-name;
        type inet:domain-name;
      }
    }
  }

  choice dest-address-host-group {
    mandatory true;
    case dest-ip {
      description "Used with address and mask couple
        to express network.";
      leaf ip-dest-address {
        type inet:ip-address;
        mandatory true;
      }
      leaf ip-dest-mask {
        type inet:ip-address;
        mandatory true;
      }
    }
    leaf ip-dest-any {
      type empty;
      description "To express Any network or address.
        Use the any keyword as an abbreviation
        for an address and a mask of 0.0.0.0
        255.255.255.255. For example:
        0.0.0.0/255.255.255.255 means 'any'";
    }
    case dest-host {
      description "Used with host address to express a
        single host
        Use the host address(or name)
        combination is the same as an address
        and mask of address 0.0.0.0."
    }
  }
}
```

For example: '10.1.1.2/0.0.0.0' is the same
as 'host 10.1.1.2'";

```

        uses IP-DESTINATION-HOST;
    }
    case dest-group {
        if-feature acl:ip-address-groups;
        description "Use the group keyword and group name
            to refer to a pre-defined address object group
            which is a list of address and mask.";

        leaf ip-dest-group {
            type acl:ip-address-group-ref;
        }
    }
}

grouping DSCP-OR-TOS {
    choice dscp-or-tos {
        leaf dscp {
            type inet:dscp;
            description
                "Match packets with given dscp value";
        }

        case tos {
            leaf tos {
                type c-types:tos;
                description
                    "Match packets with given TOS value";
            }
            leaf precedence {
                when "boolean(..tos)" ;
                type c-types:precedence;
                description
                    "Match packets with given precedence value";
            }
        }
    }
}

grouping IP-ACE-FILTERS {
    leaf protocol {
        type c-types:ip-protocol;
        description "IP protocol number.";
    }
}

```

```
uses acl:FILTER-COMMON;

leaf fragments {
  type empty;
  description "Check non-initial fragments";
}

leaf time-range {
  type acl:time-range-ref;
  description
    "Refer a time range object by
     name (Max Size 64).";
}

choice src-ports {
  when "protocol = '6' or protocol = '17' or " +
        "protocol = '132'";

  description
    "Apply only when the protocol is TCP,
     UDP or SCTP.";

  case port-number-range {
    description
      "Port group includes all ports between port-lower
       and port-upper (including those)";
    leaf src-port-lower {
      type inet:port-number;
      description "Lower Port number.";
      mandatory true;
    }
    leaf src-port-upper {
      type inet:port-number;
      description "Upper Port number.";
      mandatory true;
      must "../src-port-lower <= ../src-port-upper";
    }
  }
  case port-number {
    description
      "Port group includes all ports that are greater
       than, greater or equal, less than, less or equal,
       or not equal the port, per the indicated
       comparator. It is possible for the port group
       to be empty (for example, in case a port group
       that is less than the minimum port number is
       specified).";
    leaf src-comparator {
```

```
        type acl:acl-comparator;
        mandatory true;
    }
    leaf src-port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
    }
}
case port-group-ref {
    if-feature acl:port-groups;
    leaf src-port-group-name {
        type acl:port-group-ref;
        mandatory true;
        description
            "Reference a port group by the Port
            Group name.";
    }
}
} // choice src-ports

choice dest-ports {
    when "protocol = '6' or protocol = '17' or " +
        "protocol = '132'";
    description
        "Apply only when the protocol is TCP,
        UDP or SCTP.";

    case port-number-range {
        description "Port group includes all ports between
            port-lower and port-upper (including those)";
        leaf des-port-lower {
            type inet:port-number;
            description "Lower Port number.";
            mandatory true;
        }
        leaf des-port-upper {
            type inet:port-number;
            description "Upper Port number.";
            mandatory true;
            must "../des-port-lower <= ../des-port-upper";
        }
    }
}
case port-number {
    description "Port group includes all ports that
        are greater than, greater or equal, less than,
        less or equal, or not equal the port, per the
        indicated comparator. It is possible for the
```

```
        port group to be empty (for example, in case a
        port group that is less than the minimum port
        number is specified).";
    leaf des-comparator {
        type acl:acl-comparator;
        mandatory true;
    }
    leaf des-port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
    }
}
case port-group-ref {
    if-feature acl:port-groups;
    leaf des-port-group-name {
        type acl:port-group-ref;
        mandatory true;
        description
            "Reference a port group by the Port Group name.";
    }
}
} // choice dest-ports

leaf icmp-type {
    when "../protocol = '1'";
    type c-types:icmp-type;
    description
        "ICMP message type number.
        Apply only when the protocol is icmp";
}

leaf icmp-code {
    when "boolean(..icmp-type) ";
    type c-types:icmp-code;
    description
        "ICMP subtype for a given icmp type.";
}

choice packet-length-or-range {
    if-feature acl:packet-length;
    case length {
        leaf packet-length-comparator {
            type acl:acl-comparator;
            description
                "Operant that compare the packet
                length. Operands are lt (less than),
                gt (greater than), eq (equal), and neq
```



```
        (not equal).";
        mandatory true;
    }
    leaf packet-length {
        type uint32 {
            range "20..9210";
        }
        description
            "Packet length value for
            operation gt, eq, etc, other
            than range";
        //TODO need to find out why package is
        // less than 9210
        mandatory true;
    }
}
case range {
    description
        "Packet operator 'range' takes
        both lower and upper value.";

    leaf packet-length-upper {
        type uint32 {
            range "20..9210";
        }
        mandatory true;
        description "Upper Packet length";
    }

    leaf packet-length-lower {
        type uint32 {
            range "20..9210";
        }
        must "number(..../packet-length-lower) <= " +
            "number(..../packet-length-upper)";
        mandatory true;
        description "Lower packet length";
    }
}
}

leaf tcp-flag-value {
    type c-types:tcp-flag-type ;
    description "TCP flag bits that needs to be checked";
}

leaf tcp-flag-mask {
    when "boolean(..../tcp-flag-value)" ;
}
```

```
    type c-types:tcp-flag-type ;
    description "TCP flag bit that needs to be checked";
}

leaf tcp-flag-operation {
    when "boolean(..tcp-flag-value)" ;
    description
        "TCP flag Match option.
        A match occurs if the TCP
        datagram has certain TCP flags
        set or not set. You use the
        match-any keyword to allow a match
        to occur if any of the specified
        TCP flags are present, or you can
        use the match-all keyword to allow
        a match to occur only if all of
        the specified TCP flags are
        present. You must follow the
        match-any and match-all keywords
        with the + or - keyword and the
        flag-name argument to match on
        one or more TCP flags. ";
    default match-any;
    type enumeration {
        enum match-any {
            description "match any";
        }
        enum match-all {
            description "match all";
        }
    }
}

choice ttl-value-or-range {
    if-feature time-to-live;
    case value {
        leaf ttl-comparator {
            type acl:acl-comparator;

            description
                "Compares the TTL value in the packet
                to the TTL value specified in this
                ACE statement. Operands are lt (less
                than), gt (greater than), and eq
                (equal), neq (not equal).";
        }
        leaf ttl-value {
            type c-types:time-to-live;
        }
    }
}
```

```

    }
  }
  case range {
    leaf ttl-value-lower {
      type c-types:time-to-live;
      description "Lower ttl number.";
    }
    leaf ttl-value--upper {
      type c-types:time-to-live;
      description "Upper ttl number.";
    }
  }
}

/* Data Nodes */

augment "/acl:acls/acl:acl" {
  when "acl:acl-type = 'ip-acl'";

  leaf afi {
    type inet:ip-version ;
    default "ipv4";
  }

  container ipv6-aces {
    when "../afi = 'ipv6' " ;

    description
      " The ip-aces container contains a list of ip-ace.
      Each ip-ace is made of a unique ID, an optional
      remark (comment), and a filter. The filter
      requires a mandatory action (permit/deny) and one or
      more options such as source-address with mask,ttl etc";

    list ipv6-ace {
      key "name";
      ordered-by user;
      description "Layer 3 Access Control Element (ACE)";

      leaf name {
        type acl:acl-name-string;
        description "Unique ACE identifier.";
      }

      choice remark-or-ipv6-case {
        leaf remark {

```

```

    type acl:acl-remark;
    // mandatory true;
  }
  case ipv6-ace {
    container filters {

      uses IP-SOURCE-NETWORK;
      uses IP-DESTINATION-NETWORK;
      uses IP-ACE-FILTERS;
      uses DSCP-OR-TOS;

      leaf igmp-type {
        when "../protocol = '2' ";
        type c-types:igmp-code;
        description
          "IGMP message type (0 to 15) for
           filtering IGMP packets. Apply only
           when the protocol is igmp in ipv4";
      }

      leaf flow-label {
        if-feature flow-label;
        when "../protocol = '17'";
        type uint64 {
          range "0..1048575";
        }
        description
          "Flow label value. Apply only when
           the protocol is UDP in ipv6.";
        reference
          "RFC3697 IPv6 Flow Label Specification";
      }
    } // container filters

    uses acl:ACE-COMMON;
  } // case ipv6-ace
} // choice remark-or-ipv6-ace
} // list ipv6-ace
} // container ipv6-aces

container ipv4-aces {
  when "../afi = 'ipv4' " ;

  description
    "The ip-aces container contains a list of ip-ace.
     Each ip-ace is made of a unique ID, an optional
     remark (comment), and a filter. The filter requires a
     mandatory action (permit/deny) and one or more options

```

```
such as source-address with mask,ttl etc";

list ipv4-ace {
  key "name";
  ordered-by user;
  description "Layer 3 Access Control Element (ACE)";

  leaf name {
    type acl:acl-name-string;
    description "Unique ACE identifier";
  }

  choice remark-or-ipv4-ace {
    leaf remark {
      type acl:acl-remark;
      // mandatory true;
    }
    case ipv4-ace {
      container filters {
        uses IP-SOURCE-NETWORK;
        uses IP-DESTINATION-NETWORK;
        uses IP-ACE-FILTERS;
        uses DSCP-OR-TOS;
      }
      uses acl:ACE-COMMON;
    } // case ipv4-ace
  } // choice remark-or-ipv4-ace
} // list ipv4-ace
} // container ipv4-aces

leaf global-fragments {
  default "not-set";
  type enumeration {
    enum not-set;
    enum permit-all {
      description "Allow all fragments";
    }
    enum deny-all {
      description "Drop all fragments";
    }
  }
}
description
  "Optimizes fragment handling for noninitial fragments.
  When this leaf is set to 'permit-all', noninitial
  fragments will be permitted unless explicitly denied.
  When this leaf is set to 'deny-all', noninitial
  fragments will be denied unless explicitly
  permitted. ";
```

```
    }  
  }  
  
}  
  
</CODE ENDS>
```

12. ACL-MAC Configuration YANG Module

This module imports type definitions from common-types YANG defined in this model.

```
<CODE BEGINS> file "acl-mac@2012-10-12.yang"  
  
module acl-mac {  
  namespace "urn:cisco:params:xml:ns:yang:acl-mac";  
  // replace with IANA namespace when assigned  
  prefix acl-mac;  
  
  import acl { prefix acl; }  
  
  import common-types {  
    prefix "c-types";  
  }  
  
  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: netmod@ietf.org  
  
    WG Chair: David Kessens  
    david.kessens@nsn.com  
  
    WG Chair: Juergen Schoenwaelder  
    j.schoenwaelder@jacobs-university.de  
  
    Editor: Lisa Huang  
    yihuan@cisco.com
```

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module augments the 'acl' module with configuration and operational data for MAC access control list

An ACL is an ordered set of rules and actions used to filter traffic.

Each set of rules and actions is represented as an Access Control Entries (ACE). Each ACE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

MAC ACLs - MAC ACLs are used to filter traffic using the information in the Layer 2 header of each packet. MAC ACLs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC ACLs to all traffic.

Terms and Acronyms

ACE (ace): Access Control Entry

ACL (acl): Access Control List

AFI (afi): Authority and Format Identifier (Address Field Identifier)

CoS (cos): Class of Service

MAC: Media Access Control

TTL (ttl): Time to Live

VLAN (vlan): Virtual Local Area Network

VRF(vrf) : Virtual Routing and Forwarding

";

reference

"Access List Commands on Cisco IOS XR Software, Cisco Nexus 7000 Series NX-OS Security Configuration Guide, Catalyst 6500 Release 12.2SX Software Configuration Guide";

revision 2012-10-12 {
description "Initial revision. ";

```
}

/* Features */

feature ethertype-mask {
  description
    "The ability to filter packets based on ether-type mask
    in hex 0x0-0xFFFF.";
}

/* Identities */

identity mac-acl {
  base acl:acl-type;
  description "layer 2 ACL type";
}

/* Groupings */

grouping MAC-SOURCE-NETWORK {
  description "MAC address and mask pair for source.";

  grouping MAC-SOURCE-HOST {
    description
      "Choice within a case not allowed so need
      this grouping.";
    choice src-address-or-name {
      mandatory true;
      leaf source-host-address {
        type inet:ip-address;
        description
          "Use the host address combination as an
          abbreviation for an address and wildcard
          of address 0.0.0.0";
      }
      leaf source-host-name {
        if-feature acl:host-by-name;
        type inet:domain-name;
      }
    }
  }

  choice source-network {
    mandatory true;
    case source-mac {
      description
        "Used with address and mask couple to
        express network.";
    }
  }
}
```



```
        leaf source-address {
            type yang:mac-address;
            mandatory true;
            description "A source MAC address.";
        }
        leaf source-address-mask {
            type yang:mac-address;
            mandatory true;
            description "A source MAC address mask.";
        }
    }
    leaf source-any {
        type empty;
        description "To express Any network or address";
    }
    case source-host {
        description
            "Use the host address combination as an
            abbreviation for an address and wildcard
            of address 0.0.0.0";
        uses MAC-SOURCE-HOST;
    }
}

grouping MAC-DESTINATION-NETWORK {
    description
        "MAC address and mask pair for destination.";

    grouping MAC-DESTINATION-HOST {
        description
            "Choice within a case not allowed so need
            this grouping.";
        choice dest-address-or-name {
            mandatory true;
            leaf dest-host-address {
                type inet:ip-address;
                description
                    "Use the host address combination as an
                    abbreviation for an address and wildcard
                    of address 0.0.0.0";
            }
            leaf dest-host-name {
                if-feature acl:host-by-name;
                type inet:domain-name;
            }
        }
    }
}
```

```
choice dest-network {
  mandatory true;
  case dest-mac {
    description
      "Used with address and mask couple to
      express network.";
    leaf dest-address {
      type yang:mac-address;
      mandatory true;
      description "A source MAC address.";
    }
    leaf dest-address-mask {
      type yang:mac-address;
      mandatory true;
      description "A source MAC address mask.";
    }
  }
  leaf dest-any {
    type empty;
    description "To express Any network or address";
  }
  case dest-host {
    description
      "Use the host address combination as an
      abbreviation for an address and wildcard
      of address 0.0.0.0";
    uses MAC-DESTINATION-HOST;
  }
}

/* Layer 2 ACL */

augment "/acl:acls/acl:acl" {
  when "acl:acl-type = 'mac-acl'";
  description
    "Layer 2 Access Control Entry (ACE). The mac-aces
    container contains a list of mac-ace. Each mac-ace is
    comprised of a name, an optional remark
    and a rule.
    A rule is referred to as 'packet-filter', although it
    contains both a filter and an action.
    The packet-filter requires a mandatory action (permit/deny)
    and one or more options such as source-address with mask,
    ethertype, vlan etc.";
  container mac-aces {
    list mac-ace {
      key name;
    }
  }
}
```

```
ordered-by user;

leaf name {
  type acl:acl-name-string;
  description "Unique ACE identifier";
}

choice remark-or-mac-ace {
  leaf remark {
    type acl:acl-remark;
    // mandatory true;
  }
  case mac-ace {
    container filters {
      uses MAC-SOURCE-NETWORK;
      uses MAC-DESTINATION-NETWORK;

      leaf ethertype {
        type c-types:ether-type;
        description "ether-type (also known as
          protocol) in hex 0x0-0xffff";
      }

      leaf ethertype-mask {
        if-feature ethertype-mask;
        when "boolean(..ethertype)";
        type c-types:ether-type;
        default "0x0000";
        description
          "Ether-type mask in hex 0x0-0xFFFF.
          0x0 is exactly match of the Ethertype..";
      }
    }

    leaf cos {
      type c-types:cos;
      description "CoS value <0-7>";
    }

    leaf time-range {
      type acl:time-range-ref;
      description
        "Enable packet capture on this
        filter for a specify time range
        by name.";
    }

    leaf vlan {
      type c-types:vlan-identifier;
    }
  }
}
```

```
        description "VLAN number";
    }

    uses acl:FILTER-COMMON;

    } // container filters

    uses acl:ACE-COMMON;

    } // case mac-ace
    } // choice remark-or-ace
    } // list mac-ace
    } // container mac-aces
} // augment

}
```

</CODE ENDS>

13. ACL-ARP Configuration YANG Module

```
<CODE BEGINS> file "acl-arp@2012-10-12.yang"

module acl-arp {
    namespace "urn:cisco:params:xml:ns:yang:acl-arp";
    // replace with IANA namespace when assigned
    prefix acl-arp;

    import acl { prefix acl; }
    import acl-ip { prefix acl-ip; }
    import acl-mac { prefix acl-mac; }

    organization
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

    contact
        "WG Web: http://tools.ietf.org/wg/netmod/
        WG List: netmod@ietf.org

        WG Chair: David Kessens
        david.kessens@nsn.com

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

        Editor: Lisa Huang
        yihuan@cisco.com
```

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module augments the 'acl' module with configuration and operational data for ARP access control list

An ACL is an ordered set of rules and actions used to filter traffic.

Each set of rules and actions is represented as an Access Control Entries (ACE). Each ACE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

ARP ACLs - The device applies ARP ACLs to IP traffic.

Terms and Acronyms

ACE (ace): Access Control Entry

ACL (acl): Access Control List

ARP (arp): Address Resolution Protocol

IP (ip): Internet Protocol

MAC: Media Access Control

VLAN (vlan): Virtual Local Area Network

";

reference

"Access List Commands on Cisco IOS XR Software, Cisco Nexus 7000 Series NX-OS Security Configuration Guide, Catalyst 6500 Release 12.2SX Software Configuration Guide, ACL TCP Flags Filtering";

revision 2012-10-12 {
 description "Initial revision. ";
}

/* Identities */

identity arp-acl {
 base "acl:acl-type";
 description "ARP ACL type";
}

```
/* Data Nodes */

augment "/acl:acls/acl:acl" {
    when "acl:acl-type = 'arp-acl'";

    description "ARP Access Control Entry (ACE).";
    container arp-aces {
        list arp-ace {
            key "name";
            ordered-by user;

            leaf name {
                type acl:acl-name-string;
            }

            choice remark-or-arp-ace {
                leaf remark {
                    type acl:acl-remark;
                    // mandatory true;
                }
                case arp-ace {
                    container filters {
                        leaf direction {
                            default "bi-direction";
                            type enumeration {
                                enum bi-direction;
                                enum request;
                                enum response;
                            }
                        }
                        description "ARP request/response.";
                    }
                }
            }

            uses acl-ip:IP-SOURCE-NETWORK;
            uses acl-ip:IP-DESTINATION-NETWORK {
                when "../direction = 'response'";
            }

            uses acl-mac:MAC-SOURCE-NETWORK;
            uses acl-mac:MAC-DESTINATION-NETWORK {
                when "../direction = 'response'";
            }

            uses acl:FILTER-COMMON;
        } // container filters
    }

    uses acl:ACE-COMMON;
}
```

```
        } // case arp-ace
      } // choice remark-or-arp-ace
    } // list arp-ace
  } // container arp-aces
} // augment

}

</CODE ENDS>
```

14. COMMON-TYPES YANG Module

```
<CODE BEGINS> file "common-types@2012-10-12.yang"

module common-types {
  namespace "urn:cisco:params:xml:ns:yang:common-types";
  // replace with IANA namespace when assigned
  prefix c-types;

  organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  contact
    "WG Web: http://tools.ietf.org/wg/netmod/
    WG List: netmod@ietf.org

    WG Chair: David Kessens
    david.kessens@nsn.com

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

    Editor: Lisa Huang
    yihuan@cisco.com

    Editor: Alexander Clemm
    alex@cisco.com

    Editor: Andy Bierman
    andy@yumaworks.com";

  description
    "This module contains a collection of generally useful
    YANG types could be referred from multiple speciality
    components.

    Terms and Acronyms
```

```
    CoS (cos): Class of Service

    ICMP (icmp): Internet Control Message Protocol

    IGMP (igmp): Internet Group Management Protocol

    IP (ip): Internet Protocol

    IPv4 (ipv4): Internet Protocol Version 4

    IPv6 (ipv6): Internet Protocol Version 6

    TCP (tcp): Transmission Control Protocol

    ToS (tos): Type of Service

    TTL (ttl): Time to Live

    UDP (udp): User Datagram Protocol

    VLAN (vlan): Virtual Local Area Network
";
revision 2012-10-12 {
    description "Initial revision. ";
}

/* Typedefs */

typedef cos {
    type uint8 {
        range "0..7";
    }
    description
        "Class of Service.
        An integer that is in the range of the layer 2 CoS values.
        This corresponds to the 802.1p and ISL CoS values.";
    reference "IEEE 802.1p";
}

typedef tos {
    type uint8 {
        range "0..15";
    }
    description
        "tos stands for Type of service .
        The tos field are five bits in the IPv4 header.
        It could specify a datagrams priority and
        request a route for low-delay, high-throughput,
```


or highly-reliable service.

Based on these TOS values, a packet would be placed in an prioritized outgoing queue, or take a route with appropriate latency, throughput, or reliability. The following are TOS field values (expressed as binary numbers):

1000	--	minimize delay
0100	--	maximize throughput
0010	--	maximize reliability
0001	--	minimize monetary cost
0000	--	normal service

.";

reference

"RFC 791 Internet Protocol
Protocol Specification
RFC 1122 Requirements for Internet Hosts --
Communication Layers
RFC 1349 Type of Service in the Internet Protocol
Suite
RFC 2474 Definition of the Differentiated Services
Field (DS Field)
in the IPv4 and IPv6 Headers
RFC 3168 The Addition of Explicit Congestion
Notification (ECN) to IP
";

}

```
typedef precedence {
  type uint8 {
    range "0..7";
  }
}
```

description

"Indicates the IP precedence.
Precedence is three bits in IP header.

Value	Description
000 (0)	Routine or Best Effort
001 (1)	Priority
010 (2)	Immediate
011 (3)	Flash - mainly used for Voice Signaling or for Video.
100 (4)	Flash Override
101 (5)	Critical -mainly used for Voice RTP.

```
        110 (6)      Internet
        111 (7)      Network";

    reference
        "RFC 791 Internet Protocol Chapter 3.1
        Protocol Specification";
}

typedef tcp-flag-type {
    type bits {
        bit fin {
            position 0;
            description "No more data from sender";
        }
        bit syn {
            position 1;
            description "Synchronize sequence numbers";
        }
        bit rst {
            position 2;
            description "Reset the connection";
        }
        bit psh {
            position 3;
            description "Push Function";
        }
        bit ack {
            position 4;
            description "Acknowledgment field significant";
        }
        bit urg {
            position 5;
            description "Urgent Pointer field significant";
        }
    }
    description "TCP flag type";
    reference "RFC 793 TRANSMISSION CONTROL PROTOCOL";
}

typedef ether-type {
    type string {
        pattern '0x[0-9a-fA-F]{4}';
    }
    description
        "ether-type is 0x0-0xffff. The protocol number
        is a four-byte hexadecimal number prefixed with 0x.
        Valid protocol numbers are from 0x0 to 0xffff.
```

This list shows the EtherType values and their corresponding protocol keywords:

0x0600 xns-idp Xerox XNS IDP

0x0BAD vines-ip Banyan VINES IP

0x0baf vines-echo Banyan VINES Echo

0x6000 etype-6000 DEC unassigned, experimental

0x6001 mop-dump DEC Maintenance Operation Protocol
(MOP) Dump/Load Assistance

0x6002 mop-console DEC MOP Remote Console

0x6003 decnet-iv DEC DECnet Phase IV Route

0x6004 lat DEC Local Area Transport (LAT)

0x6005 diagnostic DEC DECnet Diagnostics

0x6007 lavc-sca DEC Local-Area VAX Cluster (LAVC), SCA

0x6008 amber DEC AMBER

0x6009 mumps DEC MUMPS

0x0800 ip Malformed, invalid, or deliberately corrupt
IP frames

0x8038 dec-spanning DEC LANBridge Management

0x8039 dsm DEC DSM/DDP

0x8040 netbios DEC PATHWORKS DECnet NETBIOS Emulation

0x8041 msdos DEC Local Area System Transport

0x8042 etype-8042 DEC unassigned

0x809B appletalk Kinetics EtherTalk (AppleTalk over
Ethernet)

0x80F3 aarp Kinetics AppleTalk Address Resolution
Protocol (AARP)

bpdu-sap BPDUs SAP encapsulated packets

```

        bpdusnap      BPDUs encapsulated packets
        ipx-arp        IPX Advanced Research Projects Agency
                        (ARPA)
        ipx-non-arp     IPX non arp
        lacp            Link Aggregation Control Protocol(LACP)
                        encapsulated packets
        pagp            Port Aggregation Protocol(PAGP)
                        encapsulated packets
        vtp             VTP packets
        ";
    }

typedef ip-protocol {
    type uint8{
        range "0..255";
    }
    description
        "The Internet Protocol (IP) is the principal communications
        protocol used for relaying datagrams (also known as network
        packets) across an internetwork using the Internet Protocol
        Suite.

        IP protocol number value is 0 to 255. It is an 8 bit field
        in the packet header";
    reference
        "IANA Protocol Numbers
        RFC5237 IANA Allocation Guidelines for the Protocol Field";
}

typedef igmp-code {
    //TODO: need more work. In NxOs, range is 0..15.
    // Could not match the IGMP with 0..15
    type uint8 /* {
        range "0..15";
    } */
    //IGMP v1 4 bits 0-15
    //IGMP v2 8bits. 0-
    //NXOS only support v1, but XR support v2.
    //

    description
        "Many of these IGMP types have a 'code' field. Here is
        the list of the types again with their assigned
        code fields."

        Type      Name      Reference
        -----
        0x11      IGMP Membership Query      [RFC1112]

```

```

0x12      IGMPv1 Membership Report      [RFC1112]
0x13      DVMRP                        [RFCDVMRP]
0x14      PIM version 1                 [PIMv1]
0x15      Cisco Trace Messages
0x16      IGMPv2 Membership Report      [RFC2236]
0x17      IGMPv2 Leave Group           [RFC2236]
0x1e      Multicast Traceroute Response [Fenner]
0x1f      Multicast Traceroute         [Fenner]
0x22      IGMPv3 Membership Report      [RFC3376]
";
reference
  "IANA Internet Group Management Protocol (IGMP) Type
  Numbers";
}

```

```

typedef icmp-type {
  type uint32 {
    range "0..255";
  }
  description
    "icmp-type is the Internet Control Message Protocol (ICMP)
    'type' field.
    The ICMP header starts after the IPv4 header. All ICMP
    packets will have an 8-byte header and variable-sized
    data section.
    The first 4 bytes of the header will be consistent.
    The first byte is for the ICMP type. The second byte is
    for the ICMP code.
    ICMP type is specified below

```

Type	Name	Reference
0	Echo Reply	[RFC792]
1	Unassigned	[JBP]
2	Unassigned	[JBP]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
7	Unassigned	[JBP]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Selection	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request	[RFC792]

```

    16      Information Reply                                [RFC792]
    17      Address Mask Request                            [RFC950]
    18      Address Mask Reply                              [RFC950]
    19      Reserved (for Security)                        [Solo]
    20-29   Reserved (for Robustness Experiment)           [ZSu]
    30      Traceroute                                     [RFC1393]
    31      Datagram Conversion Error                      [RFC1475]
    32      Mobile Host Redirect                           [David Johnson]
    33      IPv6 Where-Are-You                             [Bill Simpson]
    34      IPv6 I-Am-Here                                 [Bill Simpson]
    35      Mobile Registration Request                    [Bill Simpson]
    36      Mobile Registration Reply                      [Bill Simpson]
    37-255   Reserved                                     [JBP]";
reference
  "RFC1700 ASSIGNED NUMBERS
  RFC792 Internet Control Message Protocol
  RFC4443 Internet Control Message Protocol (ICMPv6)
    for the Internet Protocol Version 6 (IPv6)
    Specification
  RFC2780 IANA Allocation Guidelines For Values In
    the Internet Protocol and Related Headers";
}

typedef icmp-code {
  type uint32 {
    range "0..255";
  }
  description
    "ICMP subtype to the given type.
    The ICMP header starts after the IPv4 header. All ICMP
    packets will have an 8-byte header and variable-sized
    data section.
    The first 4 bytes of the header will be consistent.
    The first byte is for the ICMP type. The second byte
    is for the ICMP code. ";
  reference "RFC2 INTERNET CONTROL MESSAGE PROTOCOL";
}

typedef vlan-identifier {
  type uint16 {
    range "1 .. 4095";
  }
  description
    "This type denotes a VLAN tag. ";
  reference
    "RFC3069 VLAN Aggregation for Efficient IP Address
    Allocation
    IEEE 802.1Q";
}

```

```
    }  
  
    typedef time-to-live {  
        type uint8 {  
            range "0..255";  
        }  
        description "The TTL is an 8-bit field in IP header.  
            The maximum TTL value is 255.";  
    }  
}
```

</CODE ENDS>

15. Security Considerations

.

16. Open items from the previous revision

1. Are there any compatibility issues related to ACE ordering because a YANG user-order list is used instead of sequence IDs? This item is closely related to bullet item 3, see below.

2. Is an administrative function to test a packet against a specified ACL needed? The server would return an indication of permit or deny, and a leaf-list of the ACE entries that were evaluated. We believe that this addition would be valuable and have incorporated this suggestion into the "Additional Considerations" section. We expect to move it into the data model in the next revision.

3. Is the model applicable to multiple implementations - can other ACL models be accommodated? We have followed up with Juniper Yang experts, Kent Watsen and Phil Shafer, to review and check for applicability to Junos implementation. The initial feedback from Phil indicates that there do not seem to be any showstoppers and that the model does seem to be applicable. However, he suggested further scrutiny should occur. Kent identified additional Juniper experts to scrutinize the model more closely; so far no further comments have been received. We also followed up regarding whether there are other standardized models of ACLs, for example in conjunction with the Desktop Management Task Force's (DMTF) CIM (Common Information Model). ACL is not covered by the standardized portion of CIM, but there are vendor-specific extensions by vendors. We inspected one such vendor specific model and found that in essence the same design patterns were used as in the model specified in this Internet Draft, with an ACL corresponding to an ordered list of rules with filters or matching

criteria, and actions to be taken in response. It appears that mappings between the models can be accommodated in a straightforward manner.

17. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Louis Fourie, Dana Blair, Tula Kraiser, Patrick Gili, George Serpa, Martin Bjorklund, Kent Watsen, and Phil Shafer.

18. Normative References

- [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.

Authors' Addresses

Lisa Huang
Cisco Systems
EMail: yihuan@cisco.com

Alexander Clemm
Cisco Systems
EMail: alex@cisco.com

Andy Bierman
YumaWorks
EMail: andy@yumaworks.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
        "HDLc (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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2013

J. Lange
GE Digital Energy
July 09, 2012

IANA Timezone Database YANG Module
draft-ietf-netmod-iana-timezones-00

Abstract

This document defines the initial version of the iana-timezones YANG module.

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. IANA Maintained Timezones YANG Module	3
3. IANA Considerations	38
4. Security Considerations	39
5. Normative References	39
Author's Address	40

1. Introduction

This document defines the initial version of the iana-timezones YANG module for timezone configuration.

The iana-timezones module reflects IANA's existing "timezone database". The latest revision of the module can be obtained from the IANA web site.

Whenever a new timezone name is added to the IANA "timezone database", the iana-timezones module is updated by IANA.

2. IANA Maintained Timezones YANG Module

```
<CODE BEGINS> file "iana-timezones@2012-07-09.yang"
module iana-timezones {
  namespace "urn:ietf:params:xml:ns:yang:iana-timezones";
  prefix ianatz;

  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-timezone typedef, which
    contains YANG definitions for IANA-registered timezones.

    This YANG module is maintained by IANA, and reflects the
    IANA Time Zone Database.
    (http://www.iana.org/time-zones)

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

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions

Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).

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

```
revision 2012-07-09 {
  description
    "Initial revision. Using IANA Time Zone Data v. 2012c
    (Released 2012-03-27)";
  reference "RFC XXXX: TITLE";
}
typedef iana-timezone {
  description
    "A timezone location as defined by the IANA timezone
    database (http://www.iana.org/time-zones)";
  type enumeration {
    enum "Europe/Andorra" {
      value 0;
    }
    enum "Asia/Dubai" {
      value 1;
    }
    enum "Asia/Kabul" {
      value 2;
    }
    enum "America/Antigua" {
      value 3;
    }
    enum "America/Anguilla" {
      value 4;
    }
    enum "Europe/Tirane" {
      value 5;
    }
    enum "Asia/Yerevan" {
      value 6;
    }
    enum "Africa/Luanda" {
      value 7;
    }
    enum "Antarctica/McMurdo" {
      value 8;
      description
        "McMurdo Station, Ross Island";
    }
    enum "Antarctica/South_Pole" {
      value 9;
    }
  }
}
```

```
        description
            "Amundsen-Scott Station, South Pole";
    }
    enum "Antarctica/Rothera" {
        value 10;
        description
            "Rothera Station, Adelaide Island";
    }
    enum "Antarctica/Palmer" {
        value 11;
        description
            "Palmer Station, Anvers Island";
    }
    enum "Antarctica/Mawson" {
        value 12;
        description
            "Mawson Station, Holme Bay";
    }
    enum "Antarctica/Davis" {
        value 13;
        description
            "Davis Station, Vestfold Hills";
    }
    enum "Antarctica/Casey" {
        value 14;
        description
            "Casey Station, Bailey Peninsula";
    }
    enum "Antarctica/Vostok" {
        value 15;
        description
            "Vostok Station, Lake Vostok";
    }
    enum "Antarctica/DumontDUrville" {
        value 16;
        description
            "Dumont-d'Urville Station, Terre Adelie";
    }
    enum "Antarctica/Syowa" {
        value 17;
        description
            "Syowa Station, E Ongul I";
    }
    enum "Antarctica/Macquarie" {
        value 18;
        description
            "Macquarie Island Station, Macquarie Island";
    }
}
```



```
enum "America/Argentina/Buenos_Aires" {
    value 19;
    description
        "Buenos Aires (BA, CF)";
}
enum "America/Argentina/Cordoba" {
    value 20;
    description
        "most locations (CB, CC, CN, ER, FM, MN, SE, SF)";
}
enum "America/Argentina/Salta" {
    value 21;
    description
        "(SA, LP, NQ, RN)";
}
enum "America/Argentina/Jujuy" {
    value 22;
    description
        "Jujuy (JY)";
}
enum "America/Argentina/Tucuman" {
    value 23;
    description
        "Tucuman (TM)";
}
enum "America/Argentina/Catamarca" {
    value 24;
    description
        "Catamarca (CT), Chubut (CH)";
}
enum "America/Argentina/La_Rioja" {
    value 25;
    description
        "La Rioja (LR)";
}
enum "America/Argentina/San_Juan" {
    value 26;
    description
        "San Juan (SJ)";
}
enum "America/Argentina/Mendoza" {
    value 27;
    description
        "Mendoza (MZ)";
}
enum "America/Argentina/San_Luis" {
    value 28;
    description
```

```
        "San Luis (SL)";
    }
    enum "America/Argentina/Rio_Gallegos" {
        value 29;
        description
            "Santa Cruz (SC)";
    }
    enum "America/Argentina/Ushuaia" {
        value 30;
        description
            "Tierra del Fuego (TF)";
    }
    enum "Pacific/Pago_Pago" {
        value 31;
    }
    enum "Europe/Vienna" {
        value 32;
    }
    enum "Australia/Lord_Howe" {
        value 33;
        description
            "Lord Howe Island";
    }
    enum "Australia/Hobart" {
        value 34;
        description
            "Tasmania - most locations";
    }
    enum "Australia/Currie" {
        value 35;
        description
            "Tasmania - King Island";
    }
    enum "Australia/Melbourne" {
        value 36;
        description
            "Victoria";
    }
    enum "Australia/Sydney" {
        value 37;
        description
            "New South Wales - most locations";
    }
    enum "Australia/Broken_Hill" {
        value 38;
        description
            "New South Wales - Yancowinna";
    }
}
```

```
enum "Australia/Brisbane" {
    value 39;
    description
        "Queensland - most locations";
}
enum "Australia/Lindeman" {
    value 40;
    description
        "Queensland - Holiday Islands";
}
enum "Australia/Adelaide" {
    value 41;
    description
        "South Australia";
}
enum "Australia/Darwin" {
    value 42;
    description
        "Northern Territory";
}
enum "Australia/Perth" {
    value 43;
    description
        "Western Australia - most locations";
}
enum "Australia/Eucla" {
    value 44;
    description
        "Western Australia - Eucla area";
}
enum "America/Aruba" {
    value 45;
}
enum "Europe/Mariehamn" {
    value 46;
}
enum "Asia/Baku" {
    value 47;
}
enum "Europe/Sarajevo" {
    value 48;
}
enum "America/Barbados" {
    value 49;
}
enum "Asia/Dhaka" {
    value 50;
}
```

```
enum "Europe/Brussels" {  
    value 51;  
}  
enum "Africa/Ouagadougou" {  
    value 52;  
}  
enum "Europe/Sofia" {  
    value 53;  
}  
enum "Asia/Bahrain" {  
    value 54;  
}  
enum "Africa/Bujumbura" {  
    value 55;  
}  
enum "Africa/Porto-Novo" {  
    value 56;  
}  
enum "America/St_Barthelemy" {  
    value 57;  
}  
enum "Atlantic/Bermuda" {  
    value 58;  
}  
enum "Asia/Brunei" {  
    value 59;  
}  
enum "America/La_Paz" {  
    value 60;  
}  
enum "America/Kralendijk" {  
    value 61;  
}  
enum "America/Noronha" {  
    value 62;  
    description  
        "Atlantic islands";  
}  
enum "America/Belem" {  
    value 63;  
    description  
        "Amapa, E Para";  
}  
enum "America/Fortaleza" {  
    value 64;  
    description  
        "NE Brazil (MA, PI, CE, RN, PB)";  
}
```

```
enum "America/Recife" {
  value 65;
  description
    "Pernambuco";
}
enum "America/Araguaina" {
  value 66;
  description
    "Tocantins";
}
enum "America/Maceio" {
  value 67;
  description
    "Alagoas, Sergipe";
}
enum "America/Bahia" {
  value 68;
  description
    "Bahia";
}
enum "America/Sao_Paulo" {
  value 69;
  description
    "S & SE Brazil (GO, DF, MG, ES, RJ, SP, PR, SC, RS)";
}
enum "America/Campo_Grande" {
  value 70;
  description
    "Mato Grosso do Sul";
}
enum "America/Cuiaba" {
  value 71;
  description
    "Mato Grosso";
}
enum "America/Santarem" {
  value 72;
  description
    "W Para";
}
enum "America/Porto_Velho" {
  value 73;
  description
    "Rondonia";
}
enum "America/Boa_Vista" {
  value 74;
  description
```

```
        "Roraima";
    }
    enum "America/Manaus" {
        value 75;
        description
            "E Amazonas";
    }
    enum "America/Eirunepe" {
        value 76;
        description
            "W Amazonas";
    }
    enum "America/Rio_Branco" {
        value 77;
        description
            "Acre";
    }
    enum "America/Nassau" {
        value 78;
    }
    enum "Asia/Thimphu" {
        value 79;
    }
    enum "Africa/Gaborone" {
        value 80;
    }
    enum "Europe/Minsk" {
        value 81;
    }
    enum "America/Belize" {
        value 82;
    }
    enum "America/St_Johns" {
        value 83;
        description
            "Newfoundland Time, including SE Labrador";
    }
    enum "America/Halifax" {
        value 84;
        description
            "Atlantic Time - Nova Scotia (most places), PEI";
    }
    enum "America/Glace_Bay" {
        value 85;
        description
            "Atlantic Time - Nova Scotia - places that did not observe
            DST 1966-1971";
    }
}
```

```
enum "America/Moncton" {
  value 86;
  description
    "Atlantic Time - New Brunswick";
}
enum "America/Goose_Bay" {
  value 87;
  description
    "Atlantic Time - Labrador - most locations";
}
enum "America/Blanc-Sablon" {
  value 88;
  description
    "Atlantic Standard Time - Quebec - Lower North Shore";
}
enum "America/Montreal" {
  value 89;
  description
    "Eastern Time - Quebec - most locations";
}
enum "America/Toronto" {
  value 90;
  description
    "Eastern Time - Ontario - most locations";
}
enum "America/Nipigon" {
  value 91;
  description
    "Eastern Time - Ontario & Quebec - places that did not
    observe DST 1967-1973";
}
enum "America/Thunder_Bay" {
  value 92;
  description
    "Eastern Time - Thunder Bay, Ontario";
}
enum "America/Iqaluit" {
  value 93;
  description
    "Eastern Time - east Nunavut - most locations";
}
enum "America/Pangnirtung" {
  value 94;
  description
    "Eastern Time - Pangnirtung, Nunavut";
}
enum "America/Resolute" {
  value 95;
```

```
    description
      "Central Standard Time - Resolute, Nunavut";
  }
  enum "America/Atikokan" {
    value 96;
    description
      "Eastern Standard Time - Atikokan, Ontario and Southampton I,
      Nunavut";
  }
  enum "America/Rankin_Inlet" {
    value 97;
    description
      "Central Time - central Nunavut";
  }
  enum "America/Winnipeg" {
    value 98;
    description
      "Central Time - Manitoba & west Ontario";
  }
  enum "America/Rainy_River" {
    value 99;
    description
      "Central Time - Rainy River & Fort Frances, Ontario";
  }
  enum "America/Regina" {
    value 100;
    description
      "Central Standard Time - Saskatchewan - most locations";
  }
  enum "America/Swift_Current" {
    value 101;
    description
      "Central Standard Time - Saskatchewan - midwest";
  }
  enum "America/Edmonton" {
    value 102;
    description
      "Mountain Time - Alberta, east British Columbia & west
      Saskatchewan";
  }
  enum "America/Cambridge_Bay" {
    value 103;
    description
      "Mountain Time - west Nunavut";
  }
  enum "America/Yellowknife" {
    value 104;
    description
```



```
        "Mountain Time - central Northwest Territories";
    }
    enum "America/Inuvik" {
        value 105;
        description
            "Mountain Time - west Northwest Territories";
    }
    enum "America/Creston" {
        value 106;
        description
            "Mountain Standard Time - Creston, British Columbia";
    }
    enum "America/Dawson_Creek" {
        value 107;
        description
            "Mountain Standard Time - Dawson Creek & Fort Saint John,
            British Columbia";
    }
    enum "America/Vancouver" {
        value 108;
        description
            "Pacific Time - west British Columbia";
    }
    enum "America/Whitehorse" {
        value 109;
        description
            "Pacific Time - south Yukon";
    }
    enum "America/Dawson" {
        value 110;
        description
            "Pacific Time - north Yukon";
    }
    enum "Indian/Cocos" {
        value 111;
    }
    enum "Africa/Kinshasa" {
        value 112;
        description
            "west Dem. Rep. of Congo";
    }
    enum "Africa/Lubumbashi" {
        value 113;
        description
            "east Dem. Rep. of Congo";
    }
    enum "Africa/Bangui" {
        value 114;
```

```
}
enum "Africa/Brazzaville" {
    value 115;
}
enum "Europe/Zurich" {
    value 116;
}
enum "Africa/Abidjan" {
    value 117;
}
enum "Pacific/Rarotonga" {
    value 118;
}
enum "America/Santiago" {
    value 119;
    description
        "most locations";
}
enum "Pacific/Easter" {
    value 120;
    description
        "Easter Island & Sala y Gomez";
}
enum "Africa/Douala" {
    value 121;
}
enum "Asia/Shanghai" {
    value 122;
    description
        "east China - Beijing, Guangdong, Shanghai, etc.";
}
enum "Asia/Harbin" {
    value 123;
    description
        "Heilongjiang (except Mohe), Jilin";
}
enum "Asia/Chongqing" {
    value 124;
    description
        "central China - Sichuan, Yunnan, Guangxi, Shaanxi, Guizhou,
        etc.";
}
enum "Asia/Urumqi" {
    value 125;
    description
        "most of Tibet & Xinjiang";
}
enum "Asia/Kashgar" {
```

```
        value 126;
        description
            "west Tibet & Xinjiang";
    }
    enum "America/Bogota" {
        value 127;
    }
    enum "America/Costa_Rica" {
        value 128;
    }
    enum "America/Havana" {
        value 129;
    }
    enum "Atlantic/Cape_Verde" {
        value 130;
    }
    enum "America/Curacao" {
        value 131;
    }
    enum "Indian/Christmas" {
        value 132;
    }
    enum "Asia/Nicosia" {
        value 133;
    }
    enum "Europe/Prague" {
        value 134;
    }
    enum "Europe/Berlin" {
        value 135;
    }
    enum "Africa/Djibouti" {
        value 136;
    }
    enum "Europe/Copenhagen" {
        value 137;
    }
    enum "America/Dominica" {
        value 138;
    }
    enum "America/Santo_Domingo" {
        value 139;
    }
    enum "Africa/Algiers" {
        value 140;
    }
    enum "America/Guayaquil" {
        value 141;
```

```
        description
            "mainland";
    }
    enum "Pacific/Galapagos" {
        value 142;
        description
            "Galapagos Islands";
    }
    enum "Europe/Tallinn" {
        value 143;
    }
    enum "Africa/Cairo" {
        value 144;
    }
    enum "Africa/El_Aaiun" {
        value 145;
    }
    enum "Africa/Asmara" {
        value 146;
    }
    enum "Europe/Madrid" {
        value 147;
        description
            "mainland";
    }
    enum "Africa/Ceuta" {
        value 148;
        description
            "Ceuta & Melilla";
    }
    enum "Atlantic/Canary" {
        value 149;
        description
            "Canary Islands";
    }
    enum "Africa/Addis_Ababa" {
        value 150;
    }
    enum "Europe/Helsinki" {
        value 151;
    }
    enum "Pacific/Fiji" {
        value 152;
    }
    enum "Atlantic/Stanley" {
        value 153;
    }
    enum "Pacific/Chuuk" {
```

```
        value 154;
        description
            "Chuuk (Truk) and Yap";
    }
    enum "Pacific/Pohnpei" {
        value 155;
        description
            "Pohnpei (Ponape)";
    }
    enum "Pacific/Kosrae" {
        value 156;
        description
            "Kosrae";
    }
    enum "Atlantic/Faroe" {
        value 157;
    }
    enum "Europe/Paris" {
        value 158;
    }
    enum "Africa/Libreville" {
        value 159;
    }
    enum "Europe/London" {
        value 160;
    }
    enum "America/Grenada" {
        value 161;
    }
    enum "Asia/Tbilisi" {
        value 162;
    }
    enum "America/Cayenne" {
        value 163;
    }
    enum "Europe/Guernsey" {
        value 164;
    }
    enum "Africa/Accra" {
        value 165;
    }
    enum "Europe/Gibraltar" {
        value 166;
    }
    enum "America/Godthab" {
        value 167;
        description
            "most locations";
    }
```

```
}
enum "America/Danmarkshavn" {
  value 168;
  description
    "east coast, north of Scoresbysund";
}
enum "America/Scoresbysund" {
  value 169;
  description
    "Scoresbysund / Ittoqqortoormiit";
}
enum "America/Thule" {
  value 170;
  description
    "Thule / Pituffik";
}
enum "Africa/Banjul" {
  value 171;
}
enum "Africa/Conakry" {
  value 172;
}
enum "America/Guadeloupe" {
  value 173;
}
enum "Africa/Malabo" {
  value 174;
}
enum "Europe/Athens" {
  value 175;
}
enum "Atlantic/South_Georgia" {
  value 176;
}
enum "America/Guatemala" {
  value 177;
}
enum "Pacific/Guam" {
  value 178;
}
enum "Africa/Bissau" {
  value 179;
}
enum "America/Guyana" {
  value 180;
}
enum "Asia/Hong_Kong" {
  value 181;
```

```
}
enum "America/Tegucigalpa" {
    value 182;
}
enum "Europe/Zagreb" {
    value 183;
}
enum "America/Port-au-Prince" {
    value 184;
}
enum "Europe/Budapest" {
    value 185;
}
enum "Asia/Jakarta" {
    value 186;
    description
        "Java & Sumatra";
}
enum "Asia/Pontianak" {
    value 187;
    description
        "west & central Borneo";
}
enum "Asia/Makassar" {
    value 188;
    description
        "east & south Borneo, Sulawesi (Celebes), Bali, Nusa
        Tenggara, west Timor";
}
enum "Asia/Jayapura" {
    value 189;
    description
        "west New Guinea (Irian Jaya) & Maluku (Moluccas)";
}
enum "Europe/Dublin" {
    value 190;
}
enum "Asia/Jerusalem" {
    value 191;
}
enum "Europe/Isle_of_Man" {
    value 192;
}
enum "Asia/Kolkata" {
    value 193;
}
enum "Indian/Chagos" {
    value 194;
```

```
}
enum "Asia/Baghdad" {
    value 195;
}
enum "Asia/Tehran" {
    value 196;
}
enum "Atlantic/Reykjavik" {
    value 197;
}
enum "Europe/Rome" {
    value 198;
}
enum "Europe/Jersey" {
    value 199;
}
enum "America/Jamaica" {
    value 200;
}
enum "Asia/Amman" {
    value 201;
}
enum "Asia/Tokyo" {
    value 202;
}
enum "Africa/Nairobi" {
    value 203;
}
enum "Asia/Bishkek" {
    value 204;
}
enum "Asia/Phnom_Penh" {
    value 205;
}
enum "Pacific/Tarawa" {
    value 206;
    description
        "Gilbert Islands";
}
enum "Pacific/Enderbury" {
    value 207;
    description
        "Phoenix Islands";
}
enum "Pacific/Kiritimati" {
    value 208;
    description
        "Line Islands";
}
```



```
}
enum "Indian/Comoro" {
    value 209;
}
enum "America/St_Kitts" {
    value 210;
}
enum "Asia/Pyongyang" {
    value 211;
}
enum "Asia/Seoul" {
    value 212;
}
enum "Asia/Kuwait" {
    value 213;
}
enum "America/Cayman" {
    value 214;
}
enum "Asia/Almaty" {
    value 215;
    description
        "most locations";
}
enum "Asia/Qyzylorda" {
    value 216;
    description
        "Qyzylorda (Kyzylorda, Kzyl-Orda)";
}
enum "Asia/Aqtobe" {
    value 217;
    description
        "Aqtobe (Aktobe)";
}
enum "Asia/Aqtau" {
    value 218;
    description
        "Atyrau (Atirau, Gur'yev), Mangghystau (Mankistau)";
}
enum "Asia/Oral" {
    value 219;
    description
        "West Kazakhstan";
}
enum "Asia/Vientiane" {
    value 220;
}
enum "Asia/Beirut" {
```

```
    value 221;
  }
  enum "America/St_Lucia" {
    value 222;
  }
  enum "Europe/Vaduz" {
    value 223;
  }
  enum "Asia/Colombo" {
    value 224;
  }
  enum "Africa/Monrovia" {
    value 225;
  }
  enum "Africa/Maseru" {
    value 226;
  }
  enum "Europe/Vilnius" {
    value 227;
  }
  enum "Europe/Luxembourg" {
    value 228;
  }
  enum "Europe/Riga" {
    value 229;
  }
  enum "Africa/Tripoli" {
    value 230;
  }
  enum "Africa/Casablanca" {
    value 231;
  }
  enum "Europe/Monaco" {
    value 232;
  }
  enum "Europe/Chisinau" {
    value 233;
  }
  enum "Europe/Podgorica" {
    value 234;
  }
  enum "America/Marigot" {
    value 235;
  }
  enum "Indian/Antananarivo" {
    value 236;
  }
  enum "Pacific/Majuro" {
```

```
        value 237;
        description
            "most locations";
    }
    enum "Pacific/Kwajalein" {
        value 238;
        description
            "Kwajalein";
    }
    enum "Europe/Skopje" {
        value 239;
    }
    enum "Africa/Bamako" {
        value 240;
    }
    enum "Asia/Rangoon" {
        value 241;
    }
    enum "Asia/Ulaanbaatar" {
        value 242;
        description
            "most locations";
    }
    enum "Asia/Hovd" {
        value 243;
        description
            "Bayan-Olgii, Govi-Altai, Hovd, Uvs, Zavkhan";
    }
    enum "Asia/Choibalsan" {
        value 244;
        description
            "Dornod, Sukhbaatar";
    }
    enum "Asia/Macau" {
        value 245;
    }
    enum "Pacific/Saipan" {
        value 246;
    }
    enum "America/Martinique" {
        value 247;
    }
    enum "Africa/Nouakchott" {
        value 248;
    }
    enum "America/Montserrat" {
        value 249;
    }
}
```

```
enum "Europe/Malta" {
    value 250;
}
enum "Indian/Mauritius" {
    value 251;
}
enum "Indian/Maldives" {
    value 252;
}
enum "Africa/Blantyre" {
    value 253;
}
enum "America/Mexico_City" {
    value 254;
    description
        "Central Time - most locations";
}
enum "America/Cancun" {
    value 255;
    description
        "Central Time - Quintana Roo";
}
enum "America/Merida" {
    value 256;
    description
        "Central Time - Campeche, Yucatan";
}
enum "America/Monterrey" {
    value 257;
    description
        "Mexican Central Time - Coahuila, Durango, Nuevo Leon,
        Tamaulipas away from US border";
}
enum "America/Matamoros" {
    value 258;
    description
        "US Central Time - Coahuila, Durango, Nuevo Leon, Tamaulipas
        near US border";
}
enum "America/Mazatlan" {
    value 259;
    description
        "Mountain Time - S Baja, Nayarit, Sinaloa";
}
enum "America/Chihuahua" {
    value 260;
    description
        "Mexican Mountain Time - Chihuahua away from US border";
}
```

```
}
enum "America/Ojinaga" {
  value 261;
  description
    "US Mountain Time - Chihuahua near US border";
}
enum "America/Hermosillo" {
  value 262;
  description
    "Mountain Standard Time - Sonora";
}
enum "America/Tijuana" {
  value 263;
  description
    "US Pacific Time - Baja California near US border";
}
enum "America/Santa_Isabel" {
  value 264;
  description
    "Mexican Pacific Time - Baja California away from US border";
}
enum "America/Bahia_Banderas" {
  value 265;
  description
    "Mexican Central Time - Bahia de Banderas";
}
enum "Asia/Kuala_Lumpur" {
  value 266;
  description
    "peninsular Malaysia";
}
enum "Asia/Kuching" {
  value 267;
  description
    "Sabah & Sarawak";
}
enum "Africa/Maputo" {
  value 268;
}
enum "Africa/Windhoek" {
  value 269;
}
enum "Pacific/Noumea" {
  value 270;
}
enum "Africa/Niamey" {
  value 271;
}
```

```
enum "Pacific/Norfolk" {
    value 272;
}
enum "Africa/Lagos" {
    value 273;
}
enum "America/Managua" {
    value 274;
}
enum "Europe/Amsterdam" {
    value 275;
}
enum "Europe/Oslo" {
    value 276;
}
enum "Asia/Kathmandu" {
    value 277;
}
enum "Pacific/Nauru" {
    value 278;
}
enum "Pacific/Niue" {
    value 279;
}
enum "Pacific/Auckland" {
    value 280;
    description
        "most locations";
}
enum "Pacific/Chatham" {
    value 281;
    description
        "Chatham Islands";
}
enum "Asia/Muscat" {
    value 282;
}
enum "America/Panama" {
    value 283;
}
enum "America/Lima" {
    value 284;
}
enum "Pacific/Tahiti" {
    value 285;
    description
        "Society Islands";
}
```

```
enum "Pacific/Marquesas" {  
    value 286;  
    description  
        "Marquesas Islands";  
}  
enum "Pacific/Gambier" {  
    value 287;  
    description  
        "Gambier Islands";  
}  
enum "Pacific/Port_Moresby" {  
    value 288;  
}  
enum "Asia/Manila" {  
    value 289;  
}  
enum "Asia/Karachi" {  
    value 290;  
}  
enum "Europe/Warsaw" {  
    value 291;  
}  
enum "America/Miquelon" {  
    value 292;  
}  
enum "Pacific/Pitcairn" {  
    value 293;  
}  
enum "America/Puerto_Rico" {  
    value 294;  
}  
enum "Asia/Gaza" {  
    value 295;  
    description  
        "Gaza Strip";  
}  
enum "Asia/Hebron" {  
    value 296;  
    description  
        "West Bank";  
}  
enum "Europe/Lisbon" {  
    value 297;  
    description  
        "mainland";  
}  
enum "Atlantic/Madeira" {  
    value 298;
```

```
        description
            "Madeira Islands";
    }
    enum "Atlantic/Azores" {
        value 299;
        description
            "Azores";
    }
    enum "Pacific/Palau" {
        value 300;
    }
    enum "America/Asuncion" {
        value 301;
    }
    enum "Asia/Qatar" {
        value 302;
    }
    enum "Indian/Reunion" {
        value 303;
    }
    enum "Europe/Bucharest" {
        value 304;
    }
    enum "Europe/Belgrade" {
        value 305;
    }
    enum "Europe/Kaliningrad" {
        value 306;
        description
            "Moscow-01 - Kaliningrad";
    }
    enum "Europe/Moscow" {
        value 307;
        description
            "Moscow+00 - west Russia";
    }
    enum "Europe/Volgograd" {
        value 308;
        description
            "Moscow+00 - Caspian Sea";
    }
    enum "Europe/Samara" {
        value 309;
        description
            "Moscow+00 - Samara, Udmurtia";
    }
    enum "Asia/Yekaterinburg" {
        value 310;
```



```
        description
            "Moscow+02 - Urals";
    }
    enum "Asia/Omsk" {
        value 311;
        description
            "Moscow+03 - west Siberia";
    }
    enum "Asia/Novosibirsk" {
        value 312;
        description
            "Moscow+03 - Novosibirsk";
    }
    enum "Asia/Novokuznetsk" {
        value 313;
        description
            "Moscow+03 - Novokuznetsk";
    }
    enum "Asia/Krasnoyarsk" {
        value 314;
        description
            "Moscow+04 - Yenisei River";
    }
    enum "Asia/Irkutsk" {
        value 315;
        description
            "Moscow+05 - Lake Baikal";
    }
    enum "Asia/Yakutsk" {
        value 316;
        description
            "Moscow+06 - Lena River";
    }
    enum "Asia/Vladivostok" {
        value 317;
        description
            "Moscow+07 - Amur River";
    }
    enum "Asia/Sakhalin" {
        value 318;
        description
            "Moscow+07 - Sakhalin Island";
    }
    enum "Asia/Magadan" {
        value 319;
        description
            "Moscow+08 - Magadan";
    }
}
```

```
enum "Asia/Kamchatka" {  
    value 320;  
    description  
        "Moscow+08 - Kamchatka";  
}  
enum "Asia/Anadyr" {  
    value 321;  
    description  
        "Moscow+08 - Bering Sea";  
}  
enum "Africa/Kigali" {  
    value 322;  
}  
enum "Asia/Riyadh" {  
    value 323;  
}  
enum "Pacific/Guadalcanal" {  
    value 324;  
}  
enum "Indian/Mahe" {  
    value 325;  
}  
enum "Africa/Khartoum" {  
    value 326;  
}  
enum "Europe/Stockholm" {  
    value 327;  
}  
enum "Asia/Singapore" {  
    value 328;  
}  
enum "Atlantic/St_Helena" {  
    value 329;  
}  
enum "Europe/Ljubljana" {  
    value 330;  
}  
enum "Arctic/Longyearbyen" {  
    value 331;  
}  
enum "Europe/Bratislava" {  
    value 332;  
}  
enum "Africa/Freetown" {  
    value 333;  
}  
enum "Europe/San_Marino" {  
    value 334;  
}
```

```
}
enum "Africa/Dakar" {
  value 335;
}
enum "Africa/Mogadishu" {
  value 336;
}
enum "America/Paramaribo" {
  value 337;
}
enum "Africa/Juba" {
  value 338;
}
enum "Africa/Sao_Tome" {
  value 339;
}
enum "America/El_Salvador" {
  value 340;
}
enum "America/Lower_Princes" {
  value 341;
}
enum "Asia/Damascus" {
  value 342;
}
enum "Africa/Mbabane" {
  value 343;
}
enum "America/Grand_Turk" {
  value 344;
}
enum "Africa/Ndjamena" {
  value 345;
}
enum "Indian/Kerguelen" {
  value 346;
}
enum "Africa/Lome" {
  value 347;
}
enum "Asia/Bangkok" {
  value 348;
}
enum "Asia/Dushanbe" {
  value 349;
}
enum "Pacific/Fakaofo" {
  value 350;
}
```

```
}
enum "Asia/Dili" {
  value 351;
}
enum "Asia/Ashgabat" {
  value 352;
}
enum "Africa/Tunis" {
  value 353;
}
enum "Pacific/Tongatapu" {
  value 354;
}
enum "Europe/Istanbul" {
  value 355;
}
enum "America/Port_of_Spain" {
  value 356;
}
enum "Pacific/Funafuti" {
  value 357;
}
enum "Asia/Taipei" {
  value 358;
}
enum "Africa/Dar_es_Salaam" {
  value 359;
}
enum "Europe/Kiev" {
  value 360;
  description
    "most locations";
}
enum "Europe/Uzhgorod" {
  value 361;
  description
    "Ruthenia";
}
enum "Europe/Zaporozhye" {
  value 362;
  description
    "Zaporozh'ye, E Lugansk / Zaporizhia, E Luhansk";
}
enum "Europe/Simferopol" {
  value 363;
  description
    "central Crimea";
}
```

```
enum "Africa/Kampala" {
    value 364;
}
enum "Pacific/Johnston" {
    value 365;
    description
        "Johnston Atoll";
}
enum "Pacific/Midway" {
    value 366;
    description
        "Midway Islands";
}
enum "Pacific/Wake" {
    value 367;
    description
        "Wake Island";
}
enum "America/New_York" {
    value 368;
    description
        "Eastern Time";
}
enum "America/Detroit" {
    value 369;
    description
        "Eastern Time - Michigan - most locations";
}
enum "America/Kentucky/Louisville" {
    value 370;
    description
        "Eastern Time - Kentucky - Louisville area";
}
enum "America/Kentucky/Monticello" {
    value 371;
    description
        "Eastern Time - Kentucky - Wayne County";
}
enum "America/Indiana/Indianapolis" {
    value 372;
    description
        "Eastern Time - Indiana - most locations";
}
enum "America/Indiana/Vincennes" {
    value 373;
    description
        "Eastern Time - Indiana - Daviess, Dubois, Knox & Martin
        Counties";
}
```

```
}
enum "America/Indiana/Winamac" {
  value 374;
  description
    "Eastern Time - Indiana - Pulaski County";
}
enum "America/Indiana/Marengo" {
  value 375;
  description
    "Eastern Time - Indiana - Crawford County";
}
enum "America/Indiana/Petersburg" {
  value 376;
  description
    "Eastern Time - Indiana - Pike County";
}
enum "America/Indiana/Vevay" {
  value 377;
  description
    "Eastern Time - Indiana - Switzerland County";
}
enum "America/Chicago" {
  value 378;
  description
    "Central Time";
}
enum "America/Indiana/Tell_City" {
  value 379;
  description
    "Central Time - Indiana - Perry County";
}
enum "America/Indiana/Knox" {
  value 380;
  description
    "Central Time - Indiana - Starke County";
}
enum "America/Menominee" {
  value 381;
  description
    "Central Time - Michigan - Dickinson, Gogebic, Iron &
    Menominee Counties";
}
enum "America/North_Dakota/Center" {
  value 382;
  description
    "Central Time - North Dakota - Oliver County";
}
enum "America/North_Dakota/New_Salem" {
```

```
    value 383;
    description
      "Central Time - North Dakota - Morton County (except Mandan
        area)";
  }
  enum "America/North_Dakota/Beulah" {
    value 384;
    description
      "Central Time - North Dakota - Mercer County";
  }
  enum "America/Denver" {
    value 385;
    description
      "Mountain Time";
  }
  enum "America/Boise" {
    value 386;
    description
      "Mountain Time - south Idaho & east Oregon";
  }
  enum "America/Shiprock" {
    value 387;
    description
      "Mountain Time - Navajo";
  }
  enum "America/Phoenix" {
    value 388;
    description
      "Mountain Standard Time - Arizona";
  }
  enum "America/Los_Angeles" {
    value 389;
    description
      "Pacific Time";
  }
  enum "America/Anchorage" {
    value 390;
    description
      "Alaska Time";
  }
  enum "America/Juneau" {
    value 391;
    description
      "Alaska Time - Alaska panhandle";
  }
  enum "America/Sitka" {
    value 392;
    description
```

```
        "Alaska Time - southeast Alaska panhandle";
    }
    enum "America/Yakutat" {
        value 393;
        description
            "Alaska Time - Alaska panhandle neck";
    }
    enum "America/Nome" {
        value 394;
        description
            "Alaska Time - west Alaska";
    }
    enum "America/Adak" {
        value 395;
        description
            "Aleutian Islands";
    }
    enum "America/Metlakatla" {
        value 396;
        description
            "Metlakatla Time - Annette Island";
    }
    enum "Pacific/Honolulu" {
        value 397;
        description
            "Hawaii";
    }
    enum "America/Montevideo" {
        value 398;
    }
    enum "Asia/Samarkand" {
        value 399;
        description
            "west Uzbekistan";
    }
    enum "Asia/Tashkent" {
        value 400;
        description
            "east Uzbekistan";
    }
    enum "Europe/Vatican" {
        value 401;
    }
    enum "America/St_Vincent" {
        value 402;
    }
    enum "America/Caracas" {
        value 403;
```



```
    }
    enum "America/Tortola" {
        value 404;
    }
    enum "America/St_Thomas" {
        value 405;
    }
    enum "Asia/Ho_Chi_Minh" {
        value 406;
    }
    enum "Pacific/Efate" {
        value 407;
    }
    enum "Pacific/Wallis" {
        value 408;
    }
    enum "Pacific/Apia" {
        value 409;
    }
    enum "Asia/Aden" {
        value 410;
    }
    enum "Indian/Mayotte" {
        value 411;
    }
    enum "Africa/Johannesburg" {
        value 412;
    }
    enum "Africa/Lusaka" {
        value 413;
    }
    enum "Africa/Harare" {
        value 414;
    }
  }
}
```

<CODE ENDS>

3. IANA Considerations

This document defines the initial version of the IANA-maintained iana-timezones YANG module.

The iana-timezones module is intended to reflect the IANA "timezone database". When a timezone location is added to the database, the

"iana-timezone" enumeration MUST be updated as defined in RFC 6020 Section 10 to add the newly created timezone location to the enumeration. The new "enum" statement MUST be added to the "iana-timezone" typedef with the same name as the newly added timezone location. A new enum value MUST be allocated by IANA and applied to the newly created enum entry. New entries MAY be placed in any order in the enumeration as long as the previously assigned enumeration values are not changed.

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

This document registers one 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:iana-timezones

Registrant Contact: IANA.

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

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

name: iana-timezones

namespace: urn:ietf:params:xml:ns:yang:iana-timezones

prefix: ianatz

reference: RFC XXXX

4. Security Considerations

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

5. 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

Jeffrey Lange
GE Digital Energy

Email: jeffrey.k.lange@ge.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 10, 2013

M. Bjorklund
Tail-f Systems
February 6, 2013

A YANG Data Model for Interface Management
draft-ietf-netmod-interfaces-cfg-09

Abstract

This document defines a YANG data model for the management of network interfaces. It is expected that interface type specific 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 August 10, 2013.

Copyright Notice

Copyright (c) 2013 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	10
6. IANA Considerations	23
7. Security Considerations	24
8. Acknowledgments	25
9. References	26
9.1. Normative References	26
9.2. Informative References	26
Appendix A. Example: Ethernet Interface Module	27
Appendix B. Example: Ethernet Bonding Interface Module	29
Appendix C. Example: VLAN Interface Module	30
Appendix D. Example: NETCONF <get> reply	31
Appendix E. Examples: Interface Naming Schemes	32
E.1. Router with Restricted Interface Names	32
E.2. Router with Arbitrary Interface Names	33
E.3. Ethernet Switch with Restricted Interface Names	33
E.4. Generic Host with Restricted Interface Names	34
E.5. Generic Host with Arbitrary Interface Names	35
Appendix F. ChangeLog	36
F.1. Version -08	36
F.2. Version -07	36
F.3. Version -06	36
F.4. Version -05	36
F.5. Version -04	36
F.6. Version -03	36
F.7. Version -02	37
F.8. Version -01	37
Author's Address	38

1. Introduction

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

Network interfaces are central to the management 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, "?" means that the leaf is optional, and "*" denotes a leaf-list:

```
+--rw interfaces
  +--rw interface [name]
    +--rw name string
    +--rw description? string
    +--rw type ianaift:iana-if-type
    +--rw location? string
    +--rw enabled? boolean
    +--ro oper-status? enumeration
    +--ro last-change? yang:date-and-time
    +--ro if-index? int32
    +--rw link-up-down-trap-enable? enumeration
    +--ro phys-address? yang:phys-address
    +--ro higher-layer-if* interface-ref
    +--ro lower-layer-if* interface-ref
    +--ro speed? yang:gauge64
    +--ro statistics
      +--ro discontinuity-time? yang:date-and-time
      +--ro in-octets? yang:counter64
      +--ro in-unicast-pkts? yang:counter64
      +--ro in-broadcast-pkts? yang:counter64
      +--ro in-multicast-pkts? yang:counter64
      +--ro in-discards? yang:counter32
      +--ro in-errors? yang:counter32
      +--ro in-unknown-protos? yang:counter32
      +--ro out-octets? yang:counter64
      +--ro out-unicast-pkts? yang:counter64
      +--ro out-broadcast-pkts? yang:counter64
      +--ro out-multicast-pkts? yang:counter64
      +--ro out-discards? yang:counter32
      +--ro out-errors? yang:counter32
```

3.1. The interface List

The data model for interfaces 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.

```
import 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";
    }
  }
  // other bonding config params, failover times etc.
}
```

Two state data leaf-lists, "higher-layer-if" and "lower-layer-if", represent a read-only view of the interface layering hierarchy.

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
oper-status	ifOperStatus
last-change	ifLastChange
if-index	ifIndex
link-up-down-trap-enable	ifLinkUpDownTrapEnable
phys-address	ifPhysAddress
higher-layer-if / lower-layer-if	ifStackTable
speed	ifSpeed
in-octets	ifHCInOctets
in-unicast-pkts	ifHCInUcastPkts
in-broadcast-pkts	ifHCInBroadcastPkts
in-multicast-pkts	ifHCInMulticastPkts
in-discards	ifInDiscards
in-errors	ifInErrors
in-unknown-protos	ifInUnknownProtos
out-octets	ifHCOctets
out-unicast-pkts	ifHCOctetsUcastPkts
out-broadcast-pkts	ifHCOctetsBroadcastPkts
out-multicast-pkts	ifHCOctetsMulticastPkts
out-discards	ifOutDiscards
out-errors	ifOutErrors

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@2013-02-06.yang"

```
module ietf-interfaces {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";  
    prefix if;  
  
    import ietf-yang-types {  
        prefix yang;  
    }  
    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  
        managing 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 2013-02-06 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Interface Management";
}

/* 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 arbitrary-names {
  description
    "This feature indicates that the server allows interfaces to
    be named arbitrarily.";
}

feature 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 interfaces on the device.";

  leaf name {
    type string;
    description
      "The name of the interface.

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

      If the device allows arbitrarily named interfaces, the
      feature 'arbitrary-names' is advertised.

      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.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifType";
}

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 a
    NETCONF server is sent a value that doesn't match this
    format, it MUST reply with an rpc-error with the error-tag
    'invalid-value'.

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

    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 oper-status {
  type enumeration {
    enum up {
      value 1;
      description
        "Ready to pass packets.";
    }
    enum down {
      value 2;
      description
        "The interface does not pass any packets.";
    }
    enum testing {
      value 3;
      description
        "In some test mode.  No operational packets can
        be passed.";
    }
    enum unknown {
      value 4;
      description
        "Status cannot be determined for some reason.";
    }
    enum dormant {
      value 5;
      description
        "Waiting for some external event.";
    }
    enum not-present {
      value 6;
      description
        "Some component (typically hardware) is missing.";
    }
    enum lower-layer-down {
      value 7;
      description
        "Down due to state of lower-layer interface(s).";
    }
  }
  config false;
  description
    "The current operational state of the interface.

    If 'enabled' is 'false' then 'oper-status'
    should be 'down'.  If 'enabled' is changed to 'true'
    then 'oper-status' should change to 'up' if the interface
    is ready to transmit and receive network traffic; it
    should change to 'dormant' if the interface is waiting for
```

```
        external actions (such as a serial line waiting for an
        incoming connection); it should remain in the 'down' state
        if and only if there is a fault that prevents it from
        going to the 'up' state; it should remain in the
        'not-present' state if the interface has missing
        (typically, hardware) components.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifOperStatus";
}

leaf last-change {
    type yang:date-and-time;
    config false;
    description
        "The time the interface entered its current operational
        state.  If the current state was entered prior to the
        last re-initialization of the local network management
        subsystem, then this node is not present.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifLastChange";
}

leaf if-index {
    if-feature 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 link-up-down-trap-enable {
    if-feature 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 (i.e., there are
   no 'lower-layer-if' entries), and 'disabled' otherwise.";
reference
  "RFC 2863: The Interfaces Group MIB -
   ifLinkUpDownTrapEnable";
}

leaf phys-address {
  type yang:phys-address;
  config false;
  description
    "The interface's address at its protocol sub-layer. For
    example, for an 802.x interface, this object normally
    contains a MAC address. The interface's media-specific
    modules must define the bit and byte ordering and the
    format of the value of this object. For interfaces that do
    not have such an address (e.g., a serial line), this node
    is not present.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifPhysAddress";
}

leaf-list higher-layer-if {
  type interface-ref;
  config false;
  description
    "A list of references to interfaces layered on top of this
    interface.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifStackTable";
}

leaf-list lower-layer-if {
  type interface-ref;
  config false;
  description
    "A list of references to interfaces layered underneath this
    interface.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifStackTable";
}
```

```
leaf speed {
  type yang:gauge64;
  units "bits / second";
  config false;
  description
    "An estimate of the interface's current bandwidth in bits
    per second.  For interfaces which do not vary in
    bandwidth or for those where no accurate estimation can
    be made, this node should contain the nominal bandwidth.
    For interfaces that has no concept of bandwidth, this
    node is not present.";
  reference
    "RFC 2863: The Interfaces Group MIB -
    ifSpeed, ifHighSpeed";
}

container statistics {
  config false;
  description
    "A collection of interface-related statistics objects.";

  leaf discontinuity-time {
    type yang:date-and-time;
    description
      "The time on the most recent occasion at which any one or
      more of this interface's counters suffered a
      discontinuity.  If no such discontinuities have occurred
      since the last re-initialization of the local management
      subsystem, then this node contains the time the local
      management subsystem re-initialized itself.";
  }

  leaf in-octets {
    type yang:counter64;
    description
      "The total number of octets received on the interface,
      including framing characters.

      Discontinuities in the value of this counter can occur
      at re-initialization of the management system, and at
      other times as indicated by the value of
      'discontinuity-time'.";
    reference
      "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
  }

  leaf in-unicast-pkts {
    type yang:counter64;
    description
```

```
"The number of packets, delivered by this sub-layer to a
higher (sub-)layer, which were not addressed to a
multicast or broadcast address at this sub-layer.

Discontinuities in the value of this counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
'discontinuity-time'.";
reference
  "RFC 2863: The Interfaces Group MIB - ifHCInUcastPkts";
}
leaf in-broadcast-pkts {
  type yang:counter64;
  description
    "The number of packets, delivered by this sub-layer to a
    higher (sub-)layer, which were addressed to a broadcast
    address at this sub-layer.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
  reference
    "RFC 2863: The Interfaces Group MIB -
    ifHCInBroadcastPkts";
}
leaf in-multicast-pkts {
  type yang:counter64;
  description
    "The number of packets, delivered by this sub-layer to a
    higher (sub-)layer, which were addressed to a multicast
    address at this sub-layer. For a MAC layer protocol,
    this includes both Group and Functional addresses.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
  reference
    "RFC 2863: The Interfaces Group MIB -
    ifHCInMulticastPkts";
}
leaf in-discards {
  type yang:counter32;
  description
    "The number of inbound packets which were chosen to be
    discarded even though no errors had been detected to
    prevent their being deliverable to a higher-layer
```

protocol. One possible reason for discarding such a packet could be to free up buffer space.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";

reference

"RFC 2863: The Interfaces Group MIB - ifInDiscards";

}

leaf in-errors {

type yang:counter32;

description

"For packet-oriented interfaces, the number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol. For character-oriented or fixed-length interfaces, the number of inbound transmission units that contained errors preventing them from being deliverable to a higher-layer protocol.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";

reference

"RFC 2863: The Interfaces Group MIB - ifInErrors";

}

leaf in-unknown-protos {

type yang:counter32;

description

"For packet-oriented interfaces, the number of packets received via the interface which were discarded because of an unknown or unsupported protocol. For character-oriented or fixed-length interfaces that support protocol multiplexing the number of transmission units received via the interface which were discarded because of an unknown or unsupported protocol. For any interface that does not support protocol multiplexing, this counter is not present.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'.";

reference

"RFC 2863: The Interfaces Group MIB - ifInUnknownProtos";

}

```
leaf out-octets {
  type yang:counter64;
  description
    "The total number of octets transmitted out of the
    interface, including framing characters.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifHCOutOctets";
}
leaf out-unicast-pkts {
  type yang:counter64;
  description
    "The total number of packets that higher-level protocols
    requested be transmitted, and which were not addressed
    to a multicast or broadcast address at this sub-layer,
    including those that were discarded or not sent.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifHCOutUcastPkts";
}
leaf out-broadcast-pkts {
  type yang:counter64;
  description
    "The total number of packets that higher-level protocols
    requested be transmitted, and which were addressed to a
    broadcast address at this sub-layer, including those
    that were discarded or not sent.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
  reference
    "RFC 2863: The Interfaces Group MIB -
    ifHCOutBroadcastPkts";
}
leaf out-multicast-pkts {
  type yang:counter64;
  description
    "The total number of packets that higher-level protocols
```


requested be transmitted, and which were addressed to a multicast address at this sub-layer, including those that were discarded or not sent. For a MAC layer protocol, this includes both Group and Functional addresses.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."

reference

"RFC 2863: The Interfaces Group MIB - ifHCOutMulticastPkts";

}

leaf out-discards {

type yang:counter32;

description

"The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."

reference

"RFC 2863: The Interfaces Group MIB - ifOutDiscards";

}

leaf out-errors {

type yang:counter32;

description

"For packet-oriented interfaces, the number of outbound packets that could not be transmitted because of errors. For character-oriented or fixed-length interfaces, the number of outbound transmission units that could not be transmitted because of errors.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of 'discontinuity-time'."

reference

"RFC 2863: The Interfaces Group MIB - ifOutErrors";

}

}

}

```
}  
}
```

```
<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.

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" {
        description
          "If a vlan-id is defined, a base-interface must
           be specified.";
      }
    }
  }
}
```

Appendix D. Example: NETCONF <get> 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. Examples: Interface Naming Schemes

This section gives examples of some implementation strategies.

E.1. Router with Restricted Interface Names

In this example, a router has support for 4 line cards, each with 8 ports. The slots for the cards are physically numbered from 0 to 3, and the ports on each card from 0 to 7. Each card has fast- or gigabit-ethernet ports.

The implementation restricts the names of the interfaces to one of "fastethernet-N/M" or "gigabitethernet-N/M". The "location" of an interface is a string on the form "N/M". The implementation auto-initializes the values for "type" and "location" based on the interface name.

An operator can configure a new interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>fastethernet-1/0</name>
</interface>
```

When the server processes this request, it will set the leaf "type" to "ethernetCsmacd" and "location" to "1/0". Thus, if the client performs a <get-config> right after the <edit-config> above, it will get:

```
<interface>
  <name>fastethernet-1/0</name>
  <type>ethernetCsmacd</type>
  <location>1/0</location>
</interface>
```

If the client tries to change the location of this interface with an <edit-config> containing:

```
<interface nc:operation="merge">
  <name>fastethernet-1/0</name>
  <location>1/1</location>
</interface>
```

then the server will reply with an "invalid-value" error, since the new location does not match the name.

E.2. Router with Arbitrary Interface Names

In this example, a router has support for 4 line cards, each with 8 ports. The slots for the cards are physically numbered from 0 to 3, and the ports on each card from 0 to 7. Each card has fast- or gigabit-ethernet ports.

The implementation does not restrict the interface names. This allows to more easily apply the interface configuration to a different physical interface. However, the additional level of indirection also makes it a bit more complex to map interface names found in other protocols to configuration entries. The "location" of an interface is a string on the form "N/M".

An operator can configure a new interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>acme-interface</name>
  <type>ethernetCsmacd</type>
  <location>1/0</location>
</interface>
```

If necessary, the operator can move the configuration named "acme-interface" over to a different physical interface with an <edit-config> containing:

```
<interface nc:operation="merge">
  <name>acme-interface</name>
  <location>2/4</location>
</interface>
```

E.3. Ethernet Switch with Restricted Interface Names

In this example, an ethernet switch has a number of ports, each port identified by a simple port number.

The implementation restricts the interface names to numbers that match the physical port number.

An operator can configure a new interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>6</name>
</interface>
```

When the server processes this request, it will set the leaf "type"

to "ethernetCsmacd" and "location" to "6". Thus, if the client performs a <get-config> right after the <edit-config> above, it will get:

```
<interface>
  <name>6</name>
  <type>ethernetCsmacd</type>
  <location>6</location>
</interface>
```

If the client tries to change the location of this interface with an <edit-config> containing:

```
<interface nc:operation="merge">
  <name>6</name>
  <location>5</location>
</interface>
```

then the server will reply with an "invalid-value" error, since the new location does not match the name.

E.4. Generic Host with Restricted Interface Names

In this example, a generic host has interfaces named by the kernel and without easily usable location information. The system identifies the physical interface by the name assigned by the operating system to the interface.

The implementation restricts the interface name to the operating system level name of the physical interface.

An operator can configure a new interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>eth8</name>
</interface>
```

When the server processes this request, it will set the leaf "type" to "ethernetCsmacd" and "location" to "eth8". Thus, if the client performs a <get-config> right after the <edit-config> above, it will get:

```
<interface>
  <name>eth8</name>
  <type>ethernetCsmacd</type>
  <location>eth8</location>
</interface>
```

If the client tries to change the location of this interface with an `<edit-config>` containing:

```
<interface nc:operation="merge">
  <name>eth8</name>
  <location>eth7</location>
</interface>
```

then the server will reply with an "invalid-value" error, since the new location does not match the name.

E.5. Generic Host with Arbitrary Interface Names

In this example, a generic host has interfaces named by the kernel and without easily usable location information. The system identifies the physical interface by the name assigned by the operating system to the interface.

The implementation does not restrict the interface name to the operating system level name of the physical interface. This allows to more easily apply the interface configuration to a different physical interface. However, the additional level of indirection also makes it a bit more complex to map interface names found in other protocols to configuration entries.

An operator can configure a new interface by sending an `<edit-config>` containing:

```
<interface nc:operation="create">
  <name>acme-interface</name>
  <type>ethernetCsmacd</type>
  <location>eth4</location>
</interface>
```

If necessary, the operator can move the configuration named "acme-interface" over to a different physical interface with an `<edit-config>` containing:

```
<interface nc:operation="merge">
  <name>acme-interface</name>
  <location>eth3</location>
</interface>
```

Appendix F. ChangeLog

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

F.1. Version -08

- o Removed the mtu leaf.
- o Added examples of different interface naming schemes.

F.2. Version -07

- o Made leaf speed config false.

F.3. Version -06

- o Added oper-status leaf.
- o Added leaf-lists higher-layer-if and lower-layer-if, that show the interface layering.
- o Added container statistics with counters.

F.4. 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.

F.5. 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.

F.6. 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.

F.7. Version -02

- o Editorial fixes

F.8. 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

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

M. Bjorklund
Tail-f Systems
February 11, 2013

A YANG Data Model for IP Management
draft-ietf-netmod-ip-cfg-09

Abstract

This document defines a YANG data model for management 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 August 15, 2013.

Copyright Notice

Copyright (c) 2013 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	15
6. Security Considerations	16
7. Acknowledgments	18
8. References	19
8.1. Normative References	19
8.2. Informative References	19
Appendix A. Example: NETCONF <get> reply	21
Author's Address	22

1. Introduction

This document defines a YANG [RFC6020] data model for management 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 parameters.

Parameters to manage 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 forwarding?      boolean
      +--rw mtu?              uint16
      +--rw address [ip]
        +--rw ip              inet:ipv4-address-no-zone
        +--rw (subnet)
          +--:(prefix-length)
            +--rw ip:prefix-length?  uint8
          +--:(netmask)
            +--rw ip:netmask?        yang:dotted-quad
      +--rw neighbor [ip]
        +--rw ip              inet:ipv4-address-no-zone
        +--rw phys-address?    yang:phys-address
    +--rw ipv6?
      +--rw enabled?          boolean
      +--rw forwarding?      boolean
      +--rw mtu?              uint32
      +--rw address [ip]
        +--rw ip              inet:ipv6-address-no-zone
        +--rw prefix-length    uint8
      +--rw neighbor [ip]
        +--rw ip              inet:ipv6-address-no-zone
        +--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 "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 addresses, and a list of 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 "address" entry's interface.

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/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 [I-D.ietf-netmod-rfc6021-bis] and [I-D.ietf-netmod-interfaces-cfg], and references [RFC0791], [RFC0826], [RFC2460], [RFC4861], [RFC4862], and [RFC4941].

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

<CODE BEGINS> file "ietf-ip@2013-02-11.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 2013-02-11 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for IP Management";
}

/* Features */

feature ipv4-non-contiguous-netmasks {
  description
    "Indicates support for configuring non-contiguous
    subnet masks.";
}

feature ipv6-privacy-autoconf {
  description
    "Indicates support for Privacy Extensions for Stateless Address
    Autoconfiguration in IPv6.";
  reference
    "RFC 4941: Privacy Extensions for Stateless Address
    Autoconfiguration in IPv6";
}

/* 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 {
  presence "Configure IPv4 on this interface.";
  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 forwarding {
    type boolean;
    default false;
    description
      "Controls if IPv4 packet forwarding is enabled or disabled
       on this interface.";
  }
  leaf mtu {
    type uint16 {
      range "68..max";
    }
    units octets;
    description
      "The size, in octets, of the largest IPv4 packet that the
       interface will send and receive.

       The server may restrict the allowed values for this leaf
       depending on the interface's type.

       If this leaf is not configured, the operationally used mtu
       depends on the interface's type.";
    reference
      "RFC 791: Internet Protocol";
  }
  list address {
    key "ip";
    description
      "The list of IPv4 addresses on the interface.";

    leaf ip {
      type inet:ipv4-address-no-zone;
      description
        "The IPv4 address on the interface.";
    }
    choice subnet {
      mandatory true;
    }
  }
}
```

```
    description
      "The subnet can be specified as a prefix-length, or,
       if the server supports non-contiguous netmasks, as
       a netmask.";
    leaf prefix-length {
      type uint8 {
        range "0..32";
      }
      description
        "The length of the subnet prefix.";
    }
    leaf netmask {
      if-feature ipv4-non-contiguous-netmasks;
      type yang:dotted-quad;
      description
        "The subnet specified as a netmask.";
    }
  }
}
list neighbor {
  key "ip";
  description
    "A list of 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-no-zone;
    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 {
  presence "Configure IPv6 on this interface.";
  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 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";
}
leaf mtu {
  type uint32 {
    range "1280..max";
  }
  units octets;
  description
    "The size, in octets, of the largest IPv6 packet that the
    interface will send and receive.

    The server may restrict the allowed values for this leaf
    depending on the interface's type.

    If this leaf is not configured, the operationally used mtu
    depends on the interface's type.";
  reference
    "RFC 2460: IPv6 Specification
    Section 5";
}
list address {
  key "ip";
  description
    "The list of IPv6 addresses on the interface.";

  leaf ip {
    type inet:ipv6-address-no-zone;
    description
      "The IPv6 address on the interface.";
  }
  leaf prefix-length {
    type uint8 {
      range "0..128";
    }
  }
}
```

```
    }
    mandatory true;
    description
        "The length of the subnet prefix.";
    }
}
list neighbor {
    key "ip";
    description
        "A list of 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-no-zone;
        description
            "The IPv6 address of a neighbor node.";
    }
    leaf phys-address {
        type yang:phys-address;
        description
            "The physical level address of the neighbor 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 {
  if-feature ipv6-privacy-autoconf;
  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 {
  if-feature ipv6-privacy-autoconf;
  type uint32;
  units "seconds";
  default 604800;
  description
    "The time period during which the temporary address
    is valid.";
  reference
    "RFC 4941: Privacy Extensions for Stateless Address
    Autoconfiguration in IPv6
    - TEMP_VALID_LIFETIME";
}
leaf temporary-preferred-lifetime {
  if-feature ipv6-privacy-autoconf;
  type uint32;
  units "seconds";
  default 86400;
  description
    "The time period during which 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/forwarding and ipv6/forwarding: These leafs allow a client to enable or disable the forwarding functions on the entity. By disabling the forwarding functions, an attacker would possibly be able to deny service to users. By enabling the forwarding 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.

ipv4/mtu and ipv6/mtu: Setting these leafs to very small values can be used to slow down interfaces.

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-09 (work in progress), July 2012.
- [I-D.ietf-netmod-rfc6021-bis]
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-rfc6021-bis-00 (work in progress), Feb 2013.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [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.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, 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 <get> 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">
          <mtu>1280</mtu>
          <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

Network Working Group
Internet-Draft
Obsoletes: 6021 (if approved)
Intended status: Standards Track
Expires: August 10, 2013

J. Schoenwaelder, Ed.
Jacobs University
February 6, 2013

Common YANG Data Types
draft-ietf-netmod-rfc6021-bis-00

Abstract

This document introduces a collection of common data types to be used with the YANG data modeling language. This document obsoletes RFC 6021.

Status of this Memo

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

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

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

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

Copyright Notice

Copyright (c) 2013 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Overview	4
3. Core YANG Derived Types	6
4. Internet-Specific Derived Types	17
5. IANA Considerations	27
6. Security Considerations	28
7. Contributors	29
8. Acknowledgments	30
9. References	31
9.1. Normative References	31
9.2. Informative References	31
Appendix A. Changes from RFC 6021	35
Author's Address	36

1. Introduction

YANG [RFC6020] is a data modeling language used to model configuration and state data manipulated by the Network Configuration Protocol (NETCONF) [RFC6241]. The YANG language supports a small set of built-in data types and provides mechanisms to derive other types from the built-in types.

This document introduces a collection of common data types derived from the built-in YANG data types. The derived types are designed to be applicable for modeling all areas of management information. The definitions are organized in several YANG modules. The "ietf-yang-types" module contains generally useful data types. The "ietf-inet-types" module contains definitions that are relevant for the Internet protocol suite.

This version of the document adds new type definitions to the YANG modules and obsoletes [RFC6021]. For the further details, see the revision statement of the YANG modules.

The key words "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].

2. Overview

This section provides a short overview of the types defined in subsequent sections and their equivalent Structure of Management Information Version 2 (SMIv2) [RFC2578][RFC2579] data types. A YANG data type is equivalent to an SMIv2 data type if the data types have the same set of values and the semantics of the values are equivalent.

Table 1 lists the types defined in the ietf-yang-types YANG module and the corresponding SMIv2 types (- indicates there is no corresponding SMIv2 type).

YANG type	Equivalent SMIv2 type (module)
counter32	Counter32 (SNMPv2-SMI)
zero-based-counter32	ZeroBasedCounter32 (RMON2-MIB)
counter64	Counter64 (SNMPv2-SMI)
zero-based-counter64	ZeroBasedCounter64 (HCNUM-TC)
gauge32	Gauge32 (SNMPv2-SMI)
gauge64	CounterBasedGauge64 (HCNUM-TC)
object-identifier	-
object-identifier-128	OBJECT IDENTIFIER
yang-identifier	-
date-and-time	-
timeticks	TimeTicks (SNMPv2-SMI)
timestamp	TimeStamp (SNMPv2-TC)
phys-address	PhysAddress (SNMPv2-TC)
mac-address	MacAddress (SNMPv2-TC)
xpath1.0	-
hex-string	-
uuid	-
dotted-quad	-

Table 1: ietf-yang-types

Table 2 lists the types defined in the ietf-inet-types YANG module and the corresponding SMIv2 types (if any).

YANG type	Equivalent SMIV2 type (module)
ip-version	InetVersion (INET-ADDRESS-MIB)
dscp	Dscp (DIFFSERV-DSCP-TC)
ipv6-flow-label	IPv6FlowLabel (IPV6-FLOW-LABEL-MIB)
port-number	InetPortNumber (INET-ADDRESS-MIB)
as-number	InetAutonomousSystemNumber (INET-ADDRESS-MIB)
ip-address	-
ipv4-address	-
ipv6-address	-
ip-address-no-zone	-
ipv4-address-no-zone	-
ipv6-address-no-zone	-
ip-prefix	-
ipv4-prefix	-
ipv6-prefix	-
domain-name	-
host	-
uri	Uri (URI-TC-MIB)

Table 2: ietf-inet-types

3. Core YANG Derived Types

The ietf-yang-types YANG module references [IEEE802], [ISO9834-1], [RFC2578], [RFC2579], [RFC2856], [RFC3339], [RFC4122], [RFC4502], [RFC6020], [XPath], and [XSD-TYPES].

```
<CODE BEGINS> file "ietf-yang-types@2013-01-20.yang"
```

```
module ietf-yang-types {

  namespace "urn:ietf:params:xml:ns:yang: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:    Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>";

  description
    "This module contains a collection of generally useful derived
    YANG data types.

    Copyright (c) 2013 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 2013-01-20 {
    description
```



```
"This revision adds the following new data types:
- yang-identifier
- hex-string
- uuid
- dotted-quad";
reference
  "RFC XXXX: Common YANG Data Types";
}

revision 2010-09-24 {
  description
    "Initial revision.";
  reference
    "RFC 6021: Common YANG Data Types";
}

/*** collection of counter and gauge types ***/

typedef counter32 {
  type uint32;
  description
    "The counter32 type represents a non-negative integer
    that monotonically increases until it reaches a
    maximum value of 2^32-1 (4294967295 decimal), when it
    wraps around and starts increasing again from zero.

    Counters have no defined 'initial' value, and thus, a
    single value of a counter has (in general) no information
    content. Discontinuities in the monotonically increasing
    value normally occur at re-initialization of the
    management system, and at other times as specified in the
    description of a schema node using this type. If such
    other times can occur, for example, the creation of
    a schema node of type counter32 at times other than
    re-initialization, then a corresponding schema node
    should be defined, with an appropriate type, to indicate
    the last discontinuity.

    The counter32 type should not be used for configuration
    schema nodes. A default statement SHOULD NOT be used in
    combination with the type counter32.

    In the value set and its semantics, this type is equivalent
    to the Counter32 type of the SMIV2.";
  reference
    "RFC 2578: Structure of Management Information Version 2
    (SMIV2)";
}
```

```
typedef zero-based-counter32 {  
  type yang:counter32;  
  default "0";  
  description  
    "The zero-based-counter32 type represents a counter32  
    that has the defined 'initial' value zero.  
  
    A schema node of this type will be set to zero (0) on creation  
    and will thereafter increase monotonically until it reaches  
    a maximum value of 2^32-1 (4294967295 decimal), when it  
    wraps around and starts increasing again from zero.  
  
    Provided that an application discovers a new schema node  
    of this type within the minimum time to wrap, it can use the  
    'initial' value as a delta. It is important for a management  
    station to be aware of this minimum time and the actual time  
    between polls, and to discard data if the actual time is too  
    long or there is no defined minimum time.  
  
    In the value set and its semantics, this type is equivalent  
    to the ZeroBasedCounter32 textual convention of the SMIV2."  
  reference  
    "RFC 4502: Remote Network Monitoring Management Information  
    Base Version 2";  
}
```

```
typedef counter64 {  
  type uint64;  
  description  
    "The counter64 type represents a non-negative integer  
    that monotonically increases until it reaches a  
    maximum value of 2^64-1 (18446744073709551615 decimal),  
    when it wraps around and starts increasing again from zero.  
  
    Counters have no defined 'initial' value, and thus, a  
    single value of a counter has (in general) no information  
    content. Discontinuities in the monotonically increasing  
    value normally occur at re-initialization of the  
    management system, and at other times as specified in the  
    description of a schema node using this type. If such  
    other times can occur, for example, the creation of  
    a schema node of type counter64 at times other than  
    re-initialization, then a corresponding schema node  
    should be defined, with an appropriate type, to indicate  
    the last discontinuity.  
  
    The counter64 type should not be used for configuration  
    schema nodes. A default statement SHOULD NOT be used in
```

combination with the type counter64.

In the value set and its semantics, this type is equivalent to the Counter64 type of the SMIV2.";

reference

"RFC 2578: Structure of Management Information Version 2 (SMIV2)";

}

typedef zero-based-counter64 {

type yang:counter64;

default "0";

description

"The zero-based-counter64 type represents a counter64 that has the defined 'initial' value zero.

A schema node of this type will be set to zero (0) on creation and will thereafter increase monotonically until it reaches a maximum value of $2^{64}-1$ (18446744073709551615 decimal), when it wraps around and starts increasing again from zero.

Provided that an application discovers a new schema node of this type within the minimum time to wrap, it can use the 'initial' value as a delta. It is important for a management station to be aware of this minimum time and the actual time between polls, and to discard data if the actual time is too long or there is no defined minimum time.

In the value set and its semantics, this type is equivalent to the ZeroBasedCounter64 textual convention of the SMIV2.";

reference

"RFC 2856: Textual Conventions for Additional High Capacity Data Types";

}

typedef gauge32 {

type uint32;

description

"The gauge32 type represents a non-negative integer, which may increase or decrease, but shall never exceed a maximum value, nor fall below a minimum value. The maximum value cannot be greater than $2^{32}-1$ (4294967295 decimal), and the minimum value cannot be smaller than 0. The value of a gauge32 has its maximum value whenever the information being modeled is greater than or equal to its maximum value, and has its minimum value whenever the information being modeled is smaller than or equal to its minimum value. If the information being modeled subsequently decreases

below (increases above) the maximum (minimum) value, the gauge32 also decreases (increases).

In the value set and its semantics, this type is equivalent to the Gauge32 type of the SMIV2.";

reference

"RFC 2578: Structure of Management Information Version 2 (SMIV2)";

}

typedef gauge64 {

type uint64;

description

"The gauge64 type represents a non-negative integer, which may increase or decrease, but shall never exceed a maximum value, nor fall below a minimum value. The maximum value cannot be greater than $2^{64}-1$ (18446744073709551615), and the minimum value cannot be smaller than 0. The value of a gauge64 has its maximum value whenever the information being modeled is greater than or equal to its maximum value, and has its minimum value whenever the information being modeled is smaller than or equal to its minimum value. If the information being modeled subsequently decreases below (increases above) the maximum (minimum) value, the gauge64 also decreases (increases).

In the value set and its semantics, this type is equivalent to the CounterBasedGauge64 SMIV2 textual convention defined in RFC 2856";

reference

"RFC 2856: Textual Conventions for Additional High Capacity Data Types";

}

/** collection of identifier related types */

typedef object-identifier {

type string {

pattern '([0-1](\.[1-3]?[0-9]))|(2\.(0|([1-9]\d*)))' + '(\.(0|([1-9]\d*)))';

}

description

"The object-identifier type represents administratively assigned names in a registration-hierarchical-name tree.

Values of this type are denoted as a sequence of numerical non-negative sub-identifier values. Each sub-identifier value MUST NOT exceed $2^{32}-1$ (4294967295). Sub-identifiers

are separated by single dots and without any intermediate whitespace.

The ASN.1 standard restricts the value space of the first sub-identifier to 0, 1, or 2. Furthermore, the value space of the second sub-identifier is restricted to the range 0 to 39 if the first sub-identifier is 0 or 1. Finally, the ASN.1 standard requires that an object identifier has always at least two sub-identifier. The pattern captures these restrictions.

Although the number of sub-identifiers is not limited, module designers should realize that there may be implementations that stick with the SMIV2 limit of 128 sub-identifiers.

This type is a superset of the SMIV2 OBJECT IDENTIFIER type since it is not restricted to 128 sub-identifiers. Hence, this type SHOULD NOT be used to represent the SMIV2 OBJECT IDENTIFIER type, the object-identifier-128 type SHOULD be used instead.";

reference

"ISO9834-1: Information technology -- Open Systems Interconnection -- Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the ASN.1 Object Identifier tree";

}

```
typedef object-identifier-128 {
  type object-identifier {
    pattern '\d*(\.\d*){1,127}';
  }
}
```

description

"This type represents object-identifiers restricted to 128 sub-identifiers.

In the value set and its semantics, this type is equivalent to the OBJECT IDENTIFIER type of the SMIV2.";

reference

"RFC 2578: Structure of Management Information Version 2 (SMIV2)";

}

```
typedef yang-identifier {
  type string {
    length "1..max";
    pattern '[a-zA-Z_][a-zA-Z0-9\-\_]*';
    pattern '\.|..|^[xX].*|^[mM].*|^[lL].*';
  }
}
```

```
}
description
  "A YANG identifier string as defined in RFC 6020, page 163.
  An identifier must start with an alphabetic character or
  an underscore followed by an arbitrary sequence of
  alphabetic or numeric characters, underscores, hyphens
  or dots.

  A YANG identifier MUST NOT start with any possible
  combination of the lower-case or upper-case character
  sequence 'xml'.";
reference
  "RFC 6020: YANG - A Data Modeling Language for the Network
  Configuration Protocol (NETCONF)";
}

/*** collection of date and time related types ***/

typedef date-and-time {
  type string {
    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d+)?'
      + '(Z|[\+\-]\d{2}:\d{2})';
  }
  description
    "The date-and-time type is a profile of the ISO 8601
    standard for representation of dates and times using the
    Gregorian calendar. The profile is defined by the
    date-time production in Section 5.6 of RFC 3339.

    The date-and-time type is compatible with the dateTime XML
    schema type with the following notable exceptions:

    (a) The date-and-time type does not allow negative years.

    (b) The date-and-time time-offset -00:00 indicates an unknown
        time zone (see RFC 3339) while -00:00 and +00:00 and Z all
        represent the same time zone in dateTime.

    (c) The canonical format (see below) of data-and-time values
        differs from the canonical format used by the dateTime XML
        schema type, which requires all times to be in UTC using
        the time-offset 'Z'.

    This type is not equivalent to the DateAndTime textual
    convention of the SMIV2 since RFC 3339 uses a different
    separator between full-date and full-time and provides
    higher resolution of time-secfrac."
```

The canonical format for date-and-time values with a known time zone uses a numeric time zone offset that is calculated using the device's configured known offset to UTC time. A change of the device's offset to UTC time will cause date-and-time values to change accordingly. Such changes might happen periodically in case a server follows automatically daylight saving time (DST) time zone offset changes. The canonical format for date-and-time values with an unknown time zone (usually referring to the notion of local time) uses the time-offset -00:00.";

reference

"RFC 3339: Date and Time on the Internet: Timestamps

RFC 2579: Textual Conventions for SMIV2

XSD-TYPES: XML Schema Part 2: Datatypes Second Edition";

}

typedef timeticks {

type uint32;

description

"The timeticks type represents a non-negative integer that represents the time, modulo 2³² (4294967296 decimal), in hundredths of a second between two epochs. When a schema node is defined that uses this type, the description of the schema node identifies both of the reference epochs.

In the value set and its semantics, this type is equivalent to the TimeTicks type of the SMIV2.";

reference

"RFC 2578: Structure of Management Information Version 2 (SMIV2)";

}

typedef timestamp {

type yang:timeticks;

description

"The timestamp type represents the value of an associated timeticks schema node at which a specific occurrence happened. The specific occurrence must be defined in the description of any schema node defined using this type. When the specific occurrence occurred prior to the last time the associated timeticks attribute was zero, then the timestamp value is zero. Note that this requires all timestamp values to be reset to zero when the value of the associated timeticks attribute reaches 497+ days and wraps around to zero.

The associated timeticks schema node must be specified in the description of any schema node using this type.

```

    In the value set and its semantics, this type is equivalent
    to the TimeStamp textual convention of the SMIV2.";
reference
    "RFC 2579: Textual Conventions for SMIV2";
}

/*** collection of generic address types ***/

typedef phys-address {
    type string {
        pattern '([0-9a-fA-F]{2}(:[0-9a-fA-F]{2})*)?';
    }
    description
        "Represents media- or physical-level addresses represented
        as a sequence octets, each octet represented by two hexadecimal
        numbers. Octets are separated by colons. The canonical
        representation uses lowercase characters.

        In the value set and its semantics, this type is equivalent
        to the PhysAddress textual convention of the SMIV2.";
reference
    "RFC 2579: Textual Conventions for SMIV2";
}

typedef mac-address {
    type string {
        pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){5}';
    }
    description
        "The mac-address type represents an IEEE 802 MAC address.
        The canonical representation uses lowercase characters.

        In the value set and its semantics, this type is equivalent
        to the MacAddress textual convention of the SMIV2.";
reference
    "IEEE 802: IEEE Standard for Local and Metropolitan Area
        Networks: Overview and Architecture
        RFC 2579: Textual Conventions for SMIV2";
}

/*** collection of XML specific types ***/

typedef xpath1.0 {
    type string;
    description
        "This type represents an XPATH 1.0 expression.

        When a schema node is defined that uses this type, the
```



```
        description of the schema node MUST specify the XPath
        context in which the XPath expression is evaluated.";
    reference
        "XPATh: XML Path Language (XPath) Version 1.0";
}

/*** collection of string types ***/

typedef hex-string {
    type string {
        pattern '[0-9a-fA-F]{2}(:[0-9a-fA-F]{2})*';
    }
    description
        "A hexadecimal string with octets represented as hex digits
        separated by colons. The canonical representation uses
        lowercase characters.";
}

typedef uuid {
    type string {
        pattern '[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-'
            + '[0-9a-fA-F]{4}-[0-9a-fA-F]{12}';
    }
    description
        "A Universally Unique IDentifier in the string representation
        defined in RFC 4122. The canonical representation uses
        lowercase characters.

        The following is an example of a UUID in string representation:
        f81d4fae-7dec-11d0-a765-00a0c91e6bf6
        ";
    reference
        "RFC 4122: A Universally Unique IDentifier (UUID) URN
        Namespace";
}

typedef dotted-quad {
    type string {
        pattern
            '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
            + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
    }
    description
        "An unsigned 32-bit number expressed in the dotted-quad
        notation, i.e., four octets written as decimal numbers
        and separated with the '.' (full stop) character.";
}
}
```

<CODE ENDS>

4. Internet-Specific Derived Types

The ietf-inet-types YANG module references [RFC0768], [RFC0791], [RFC0793], [RFC0952], [RFC1034], [RFC1123], [RFC1930], [RFC2460], [RFC2474], [RFC2780], [RFC2782], [RFC3289], [RFC3305], [RFC3492], [RFC3595], [RFC3986], [RFC4001], [RFC4007], [RFC4271], [RFC4291], [RFC4340], [RFC4960], [RFC5017], [RFC5891], [RFC5952], and [RFC6793].

```
<CODE BEGINS> file "ietf-inet-types@2013-01-20.yang"
```

```
module ietf-inet-types {

  namespace "urn:ietf:params:xml:ns:yang: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:     Juergen Schoenwaelder
                <mailto:j.schoenwaelder@jacobs-university.de>";

  description
    "This module contains a collection of generally useful derived
    YANG data types for Internet addresses and related things.

    Copyright (c) 2013 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 2013-01-20 {
  description
    "This revision adds the following new data types:
    - ip-address-no-zone
    - ipv4-address-no-zone
    - ipv6-address-no-zone";
  reference
    "RFC XXXX: Common YANG Data Types";
}

revision 2010-09-24 {
  description
    "Initial revision.";
  reference
    "RFC 6021: Common YANG Data Types";
}

/*** collection of protocol field related types ***/

typedef ip-version {
  type enumeration {
    enum unknown {
      value "0";
      description
        "An unknown or unspecified version of the Internet
        protocol.";
    }
    enum ipv4 {
      value "1";
      description
        "The IPv4 protocol as defined in RFC 791.";
    }
    enum ipv6 {
      value "2";
      description
        "The IPv6 protocol as defined in RFC 2460.";
    }
  }
  description
    "This value represents the version of the IP protocol.

    In the value set and its semantics, this type is equivalent
    to the InetVersion textual convention of the SMIV2.";
  reference
    "RFC 791: Internet Protocol
    RFC 2460: Internet Protocol, Version 6 (IPv6) Specification
    RFC 4001: Textual Conventions for Internet Network Addresses";
}
```

```
typedef dscp {  
  type uint8 {  
    range "0..63";  
  }  
  description  
    "The dscp type represents a Differentiated Services Code-Point  
    that may be used for marking packets in a traffic stream.  
  
    In the value set and its semantics, this type is equivalent  
    to the Dscp textual convention of the SMIV2.";  
  reference  
    "RFC 3289: Management Information Base for the Differentiated  
      Services Architecture  
    RFC 2474: Definition of the Differentiated Services Field  
      (DS Field) in the IPv4 and IPv6 Headers  
    RFC 2780: IANA Allocation Guidelines For Values In  
      the Internet Protocol and Related Headers";  
}  
  
typedef ipv6-flow-label {  
  type uint32 {  
    range "0..1048575";  
  }  
  description  
    "The flow-label type represents flow identifier or Flow Label  
    in an IPv6 packet header that may be used to discriminate  
    traffic flows.  
  
    In the value set and its semantics, this type is equivalent  
    to the IPv6FlowLabel textual convention of the SMIV2.";  
  reference  
    "RFC 3595: Textual Conventions for IPv6 Flow Label  
    RFC 2460: Internet Protocol, Version 6 (IPv6) Specification";  
}  
  
typedef port-number {  
  type uint16 {  
    range "0..65535";  
  }  
  description  
    "The port-number type represents a 16-bit port number of an  
    Internet transport layer protocol such as UDP, TCP, DCCP, or  
    SCTP. Port numbers are assigned by IANA. A current list of  
    all assignments is available from <http://www.iana.org/>.  
  
    Note that the port number value zero is reserved by IANA. In  
    situations where the value zero does not make sense, it can  
    be excluded by subtyping the port-number type.
```

```

    In the value set and its semantics, this type is equivalent
    to the InetPortNumber textual convention of the SMIV2.";
reference
  "RFC 768: User Datagram Protocol
   RFC 793: Transmission Control Protocol
   RFC 4960: Stream Control Transmission Protocol
   RFC 4340: Datagram Congestion Control Protocol (DCCP)
   RFC 4001: Textual Conventions for Internet Network Addresses";
}

/*** collection of autonomous system related types ***/

typedef as-number {
  type uint32;
  description
    "The as-number type represents autonomous system numbers
    which identify an Autonomous System (AS).  An AS is a set
    of routers under a single technical administration, using
    an interior gateway protocol and common metrics to route
    packets within the AS, and using an exterior gateway
    protocol to route packets to other ASs'.  IANA maintains
    the AS number space and has delegated large parts to the
    regional registries.

    Autonomous system numbers were originally limited to 16
    bits.  BGP extensions have enlarged the autonomous system
    number space to 32 bits.  This type therefore uses an uint32
    base type without a range restriction in order to support
    a larger autonomous system number space.

    In the value set and its semantics, this type is equivalent
    to the InetAutonomousSystemNumber textual convention of
    the SMIV2.";
  reference
    "RFC 1930: Guidelines for creation, selection, and registration
      of an Autonomous System (AS)
     RFC 4271: A Border Gateway Protocol 4 (BGP-4)
     RFC 4001: Textual Conventions for Internet Network Addresses
     RFC 6793: BGP Support for Four-octet AS Number Space";
}

/*** collection of IP address and hostname related types ***/

typedef ip-address {
  type union {
    type inet:ipv4-address;
    type inet:ipv6-address;
  }
}
```

```

description
  "The ip-address type represents an IP address and is IP
  version neutral. The format of the textual representations
  implies the IP version.";
}

typedef ipv4-address {
  type string {
    pattern
      '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
      + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
      + '(%[\p{N}\p{L}]+)?';
  }
  description
    "The ipv4-address type represents an IPv4 address in
    dotted-quad notation. The IPv4 address may include a zone
    index, separated by a % sign.

    The zone index is used to disambiguate identical address
    values. For link-local addresses, the zone index will
    typically be the interface index number or the name of an
    interface. If the zone index is not present, the default
    zone of the device will be used.

    The canonical format for the zone index is the numerical
    format";
}

typedef ipv6-address {
  type string {
    pattern '((:|0-9a-fA-F){0,4}):([0-9a-fA-F]{0,4}):{0,5}'
      + '(((0-9a-fA-F){0,4}):)?(:|0-9a-fA-F){0,4})|'
      + '(((25[0-5]|2[0-4][0-9]|01?[0-9]?[0-9])\.){3}'
      + '(25[0-5]|2[0-4][0-9]|01?[0-9]?[0-9]))'
      + '(%[\p{N}\p{L}]+)?';
    pattern '([[:^:]]+){6}([[:^:]]+:[[:^:]]+)|(.*\..*)|'
      + '([[:^:]]+:[[:^:]]+)*[[:^:]]+?::([[:^:]]+:[[:^:]]+)*[[:^:]]+?'
      + '(%.*+)?';
  }
  description
    "The ipv6-address type represents an IPv6 address in full,
    mixed, shortened, and shortened-mixed notation. The IPv6
    address may include a zone index, separated by a % sign.

    The zone index is used to disambiguate identical address
    values. For link-local addresses, the zone index will
    typically be the interface index number or the name of an
    interface. If the zone index is not present, the default

```

zone of the device will be used.

The canonical format of IPv6 addresses uses the compressed format described in RFC 4291, Section 2.2, item 2 with the following additional rules: the :: substitution must be applied to the longest sequence of all-zero 16-bit chunks in an IPv6 address. If there is a tie, the first sequence of all-zero 16-bit chunks is replaced by ::. Single all-zero 16-bit chunks are not compressed. The canonical format uses lowercase characters and leading zeros are not allowed. The canonical format for the zone index is the numerical format as described in RFC 4007, Section 11.2.";

reference

"RFC 4291: IP Version 6 Addressing Architecture
RFC 4007: IPv6 Scoped Address Architecture
RFC 5952: A Recommendation for IPv6 Address Text
Representation";

}

typedef ip-address-no-zone {

type union {

type inet:ipv4-address-no-zone;

type inet:ipv6-address-no-zone;

}

description

"The ip-address-no-zone type represents an IP address without zone information in an IP version neutral way. The format of the textual representations implies the IP version.";

}

typedef ipv4-address-no-zone {

type inet:ipv4-address {

pattern "[\.\0-9]*";

}

description

"An IPv4 address without a zone index. This type may be used in situations where the zone is known from the context and hence no zone index is needed.";

}

typedef ipv6-address-no-zone {

type inet:ipv6-address {

pattern '[0-9a-fA-F:]*';

}

description

"An IPv6 address without a zone index. This type may be used in situations where the zone is known from the context and


```

        hence no zone index is needed.";
    }

typedef ip-prefix {
    type union {
        type inet:ipv4-prefix;
        type inet:ipv6-prefix;
    }
    description
        "The ip-prefix type represents an IP prefix and is IP
        version neutral. The format of the textual representations
        implies the IP version.";
}

typedef ipv4-prefix {
    type string {
        pattern
            '(((0-9)|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.){3}'
            + '([0-9]|1[0-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])'
            + '/(((0-9)|([1-2][0-9])|(3[0-2])))';
    }
    description
        "The ipv4-prefix type represents an IPv4 address prefix.
        The prefix length is given by the number following the
        slash character and must be less than or equal to 32.

        A prefix length value of n corresponds to an IP address
        mask that has n contiguous 1-bits from the most
        significant bit (MSB) and all other bits set to 0.

        The canonical format of an IPv4 prefix has all bits of
        the IPv4 address set to zero that are not part of the
        IPv4 prefix.";
}

typedef ipv6-prefix {
    type string {
        pattern '(:|0-9a-fA-F){0,4}):([0-9a-fA-F]{0,4}):{0,5}'
            + '(((0-9a-fA-F){0,4}):?(:|0-9a-fA-F){0,4}))|'
            + '(((25[0-5]|2[0-4][0-9]|01?[0-9]?[0-9])\.){3}'
            + '(25[0-5]|2[0-4][0-9]|01?[0-9]?[0-9]))|'
            + '(/(((0-9)|([0-9]{2})|(1[0-1][0-9])|(12[0-8]))))';
        pattern '([[:^:]]+){6}([[:^:]]+:[[:^:]]+)|(.*\..*)|'
            + '([[:^:]]+)*[[:^:]]+?:([[:^:]]+)*[[:^:]]+?'
            + '(/.+)';
    }
    description
        "The ipv6-prefix type represents an IPv6 address prefix.

```

The prefix length is given by the number following the slash character and must be less than or equal 128.

A prefix length value of *n* corresponds to an IP address mask that has *n* contiguous 1-bits from the most significant bit (MSB) and all other bits set to 0.

The IPv6 address should have all bits that do not belong to the prefix set to zero.

The canonical format of an IPv6 prefix has all bits of the IPv6 address set to zero that are not part of the IPv6 prefix. Furthermore, IPv6 address is represented in the compressed format described in RFC 4291, Section 2.2, item 2 with the following additional rules: the :: substitution must be applied to the longest sequence of all-zero 16-bit chunks in an IPv6 address. If there is a tie, the first sequence of all-zero 16-bit chunks is replaced by ::. Single all-zero 16-bit chunks are not compressed. The canonical format uses lowercase characters and leading zeros are not allowed."

reference

"RFC 4291: IP Version 6 Addressing Architecture";

}

/** collection of domain name and URI types */

```
typedef domain-name {
  type string {
    pattern
      '((([a-zA-Z0-9_]([a-zA-Z0-9\_-]){0,61})?[a-zA-Z0-9]\.)*'
      + '([a-zA-Z0-9_]([a-zA-Z0-9\_-]){0,61})?[a-zA-Z0-9]\.?)'
      + '|\.';
    length "1..253";
  }
  description
    "The domain-name type represents a DNS domain name. The
    name SHOULD be fully qualified whenever possible.
```

Internet domain names are only loosely specified. Section 3.5 of RFC 1034 recommends a syntax (modified in Section 2.1 of RFC 1123). The pattern above is intended to allow for current practice in domain name use, and some possible future expansion. It is designed to hold various types of domain names, including names used for A or AAAA records (host names) and other records, such as SRV records. Note that Internet host names have a stricter syntax (described in RFC 952) than the DNS recommendations in RFCs 1034 and

1123, and that systems that want to store host names in schema nodes using the domain-name type are recommended to adhere to this stricter standard to ensure interoperability.

The encoding of DNS names in the DNS protocol is limited to 255 characters. Since the encoding consists of labels prefixed by a length bytes and there is a trailing NULL byte, only 253 characters can appear in the textual dotted notation.

The description clause of schema nodes using the domain-name type MUST describe when and how these names are resolved to IP addresses. Note that the resolution of a domain-name value may require to query multiple DNS records (e.g., A for IPv4 and AAAA for IPv6). The order of the resolution process and which DNS record takes precedence can either be defined explicitly or it may depend on the configuration of the resolver.

Domain-name values use the US-ASCII encoding. Their canonical format uses lowercase US-ASCII characters. Internationalized domain names MUST be encoded in punycode as described in RFC 3492";

reference

- "RFC 952: DoD Internet Host Table Specification
- RFC 1034: Domain Names - Concepts and Facilities
- RFC 1123: Requirements for Internet Hosts -- Application and Support
- RFC 2782: A DNS RR for specifying the location of services (DNS SRV)
- RFC 3492: Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)
- RFC 5891: Internationalizing Domain Names in Applications (IDNA): Protocol";

```
}  
  
typedef host {  
  type union {  
    type inet:ip-address;  
    type inet:domain-name;  
  }  
  description  
    "The host type represents either an IP address or a DNS  
    domain name."  
}  
  
typedef uri {
```

```
type string;
description
  "The uri type represents a Uniform Resource Identifier
  (URI) as defined by STD 66.

  Objects using the uri type MUST be in US-ASCII encoding,
  and MUST be normalized as described by RFC 3986 Sections
  6.2.1, 6.2.2.1, and 6.2.2.2. All unnecessary
  percent-encoding is removed, and all case-insensitive
  characters are set to lowercase except for hexadecimal
  digits, which are normalized to uppercase as described in
  Section 6.2.2.1.

  The purpose of this normalization is to help provide
  unique URIs. Note that this normalization is not
  sufficient to provide uniqueness. Two URIs that are
  textually distinct after this normalization may still be
  equivalent.

  Objects using the uri type may restrict the schemes that
  they permit. For example, 'data:' and 'urn:' schemes
  might not be appropriate.

  A zero-length URI is not a valid URI. This can be used to
  express 'URI absent' where required.

  In the value set and its semantics, this type is equivalent
  to the Uri SMIV2 textual convention defined in RFC 5017.";
reference
  "RFC 3986: Uniform Resource Identifier (URI): Generic Syntax
  RFC 3305: Report from the Joint W3C/IETF URI Planning Interest
  Group: Uniform Resource Identifiers (URIs), URLs,
  and Uniform Resource Names (URNs): Clarifications
  and Recommendations
  RFC 5017: MIB Textual Conventions for Uniform Resource
  Identifiers (URIs)";
}
}
<CODE ENDS>
```

5. IANA Considerations

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

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

Registrant Contact: The NETMOD WG of the IETF.

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

URI: urn:ietf:params:xml:ns:yang:ietf-inet-types

Registrant Contact: The NETMOD WG of the IETF.

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

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

name:	ietf-yang-types
namespace:	urn:ietf:params:xml:ns:yang:ietf-yang-types
prefix:	yang
reference:	RFC 6021

name:	ietf-inet-types
namespace:	urn:ietf:params:xml:ns:yang:ietf-inet-types
prefix:	inet
reference:	RFC 6021

6. Security Considerations

This document defines common data types using the YANG data modeling language. The definitions themselves have no security impact on the Internet but the usage of these definitions in concrete YANG modules might have. The security considerations spelled out in the YANG specification [RFC6020] apply for this document as well.

7. Contributors

The following people contributed significantly to the initial version of this document:

- Andy Bierman (Brocade)
- Martin Bjorklund (Tail-f Systems)
- Balazs Lengyel (Ericsson)
- David Partain (Ericsson)
- Phil Shafer (Juniper Networks)

8. Acknowledgments

The editor wishes to thank the following individuals for providing helpful comments on various versions of this document: Andy Bierman, Martin Bjorklund, Ladislav Lhotka, Lars-Johan Liman, and Dan Romascanu.

Juergen Schoenwaelder was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.
- [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [XPath] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <<http://www.w3.org/TR/1999/REC-xpath-19991116>>.

9.2. Informative References

- [IEEE802] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture", IEEE Std. 802-2001.
- [ISO9834-1] ISO/IEC, "Information technology -- Open Systems

Interconnection -- Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the ASN.1 Object Identifier tree", ISO/IEC 9834-1:2008, 2008.

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, October 1985.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.
- [RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)", BCP 6, RFC 1930, March 1996.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIV2)", STD 58, RFC 2578, April 1999.
- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIV2", STD 58, RFC 2579, April 1999.
- [RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, March 2000.
- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for

specifying the location of services (DNS SRV)", RFC 2782, February 2000.

- [RFC2856] Bierman, A., McCloghrie, K., and R. Presuhn, "Textual Conventions for Additional High Capacity Data Types", RFC 2856, June 2000.
- [RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information Base for the Differentiated Services Architecture", RFC 3289, May 2002.
- [RFC3305] Mealling, M. and R. Denenberg, "Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations", RFC 3305, August 2002.
- [RFC3595] Wijnen, B., "Textual Conventions for IPv6 Flow Label", RFC 3595, September 2003.
- [RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4502] Waldbusser, S., "Remote Network Monitoring Management Information Base Version 2", RFC 4502, May 2006.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5017] McWalter, D., "MIB Textual Conventions for Uniform Resource Identifiers (URIs)", RFC 5017, September 2007.
- [RFC5891] Klensin, J., "Internationalizing Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.
- [RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "NETCONF Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, December 2012.
- [XSD-TYPES] Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

Appendix A. Changes from RFC 6021

This version adds new type definitions to the YANG modules. For the further details, see the revision statement of the YANG modules.

Author's Address

Juergen Schoenwaelder (editor)
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: August 27, 2013

L. Lhotka
CZ.NIC
February 23, 2013

A YANG Data Model for Routing Management
draft-ietf-netmod-routing-cfg-09

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 and managing a routing subsystem. It is 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 extensions - router instances, routes, routing tables, routing protocols and route filters.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 27, 2013.

Copyright Notice

Copyright (c) 2013 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	4
2. Terminology and Notation	5
2.1. Glossary of New Terms	5
2.2. Tree Diagrams	6
2.3. Prefixes in Data Node Names	6
3. Objectives	8
4. The Design of the Core Routing Data Model	9
4.1. Router	12
4.1.1. Configuration of IPv6 Router Interfaces	12
4.2. Routes	14
4.3. Routing Tables	14
4.4. Routing Protocols	16
4.4.1. Routing Pseudo-Protocols	16
4.4.2. Defining New Routing Protocols	17
4.5. Route Filters	18
4.6. RPC Operations	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	36
8. IPv6 Unicast Routing YANG Module	40
9. IANA Considerations	50
10. Security Considerations	52
11. Acknowledgments	53
12. References	54
12.1. Normative References	54
12.2. Informative References	54
Appendix A. The Complete Data Tree	55
Appendix B. Example: Adding a New Routing Protocol	57
Appendix C. Example: NETCONF <get> Reply	60
Appendix D. Change Log	65
D.1. Changes Between Versions -08 and -09	65
D.2. Changes Between Versions -07 and -08	65
D.3. Changes Between Versions -06 and -07	65
D.4. Changes Between Versions -05 and -06	65
D.5. Changes Between Versions -04 and -05	66
D.6. Changes Between Versions -03 and -04	67
D.7. Changes Between Versions -02 and -03	67
D.8. Changes Between Versions -01 and -02	68
D.9. Changes Between Versions -00 and -01	68

Author's Address	69
----------------------------	----

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 configuration and management of more sophisticated routing systems. 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 data model
- o data node
- o mandatory node
- o module
- o state data
- o RPC operation

2.1. Glossary of New Terms

active route: a route that 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. Tree Diagrams

A simplified graphical representation of the complete data tree is presented in Appendix A, and similar diagrams of its various subtrees appear in the main text. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2.3. 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]
rt	ietf-routing	Section 6
v4ur	ietf-ipv4-unicast-routing	Section 7
v6ur	ietf-ipv6-unicast-routing	Section 8
yang	ietf-yang-types	[RFC6021bis]
inet	ietf-inet-types	[RFC6021bis]

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. An abridged view of the data hierarchy is shown in Figure 1. See Appendix A for the complete data tree.


```

+--rw routing
  +--rw router [name]
    |   +--rw name
    |   +--rw type?
    |   +--rw enabled?
    |   +--rw router-id?
    |   +--rw description?
    |   +--rw main-routing-tables
    |   |   +--rw main-routing-table [address-family safi]
    |   |   |   +--rw address-family
    |   |   |   +--rw safi
    |   |   |   +--rw name?
    |   +--rw interfaces
    |   |   +--rw interface [name]
    |   |   |   +--rw name
    |   |   |   +--rw v6ur:ipv6-router-advertisements
    |   |   |   ...
    |   +--rw routing-protocols
    |   |   +--rw routing-protocol [name]
    |   |   |   +--rw name
    |   |   |   +--rw description?
    |   |   |   +--rw enabled?
    |   |   |   +--rw type
    |   |   |   +--rw connected-routing-tables
    |   |   |   |   ...
    |   |   |   +--rw static-routes
    |   |   |   ...
    +--rw routing-tables
    |   +--rw routing-table [name]
    |   |   +--rw name
    |   |   +--rw address-family
    |   |   +--rw safi
    |   |   +--rw description?
    |   |   +--ro routes
    |   |   |   +--ro route
    |   |   |   ...
    |   +--rw recipient-routing-tables
    |   |   +--rw recipient-routing-table [name]
    |   |   ...
    +--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 lists of 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.

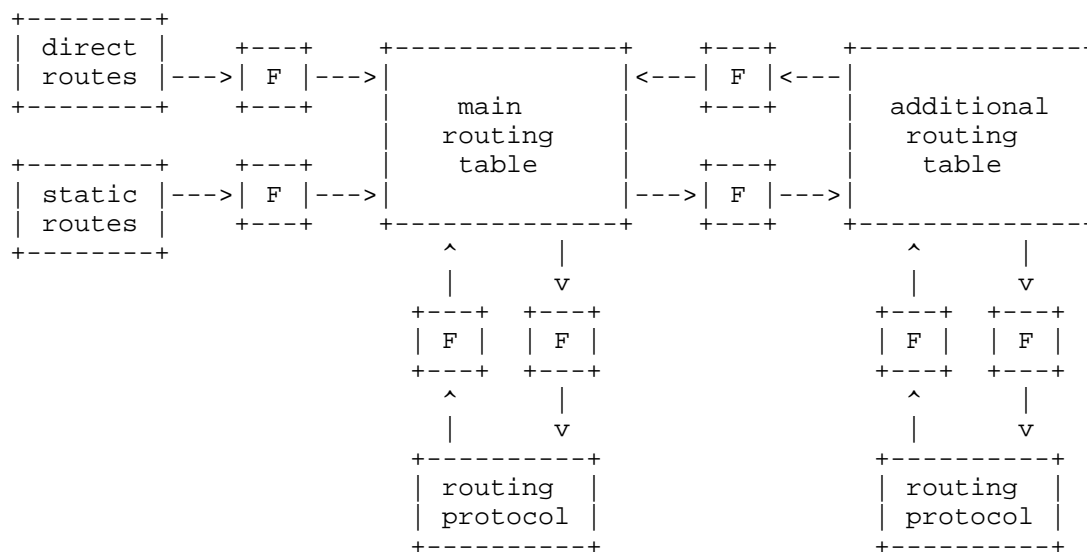


Figure 2: Example setup of a routing system

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.

An implementation MAY support multiple types of logical routers simultaneously. Instances of all router types are organized as entries of the same flat "router" list. In order to discriminate router instances belonging to different types, the "type" leaf is defined as a child of the "router" node.

An implementation MAY pose restrictions on allowed router types and on the number of supported instances for each type. For example, a simple router implementation may support only one router instance of the default type "standard-router".

Each network layer interface has to 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 is 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].

In YANG terms, the list of router interfaces is modeled as the "list" node rather than "leaf-list" in order to allow for adding, via augmentation, other configuration or state data related to the corresponding router interface.

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.

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: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. Routes

Routes are basic elements of information in a routing system. The core routing data model defines only the following minimal set of route attributes:

- o "dest-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 "dest-prefix" should be sent.
- o "outgoing-interface": network interface that should be used for sending packets with destination addresses belonging to "dest-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 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 "last-updated": the date and time when the route was last updated, or inserted into the routing table.

Each routing table must 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 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.

In order to activate an address family for use within a router instance, a client configures an entry of the list `/routing/router/main-routing-tables/main-routing-table`. This entry contains a reference to a routing table which henceforth serves as the so-called main routing table for the router instance and address family. Section 4.4 explains the role of main routing tables.

Routing tables are global, which means that a configured routing table may be used by any or all router instances.

Server implementations MAY pose restrictions regarding the number of supported routing tables, and rules for configuration and use of routing tables. For example:

- o A server may support no more than one routing table per address family.
- o Router instances (of a certain type) may not be allowed to share routing tables, i.e., each routing table is used by no more than one router instance.

For servers supporting multiple routing tables per address family, additional tables can 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 way how a routing system uses information from routing tables for actual packet forwarding is outside the scope of this document.

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. Optionally, a route filter may be configured for any or all recipient routing tables. Such a route filter then selects and/or manipulates the routes that are passed between the source and recipient routing table.

A routing table MUST NOT appear among its own recipient routing tables. A recipient routing table also MUST be of the same address family as its source routing table.

4.4. Routing Protocols

The core routing data model provides an open-ended framework for defining multiple routing protocol instances within each router instance. Each routing protocol instance **MUST** be assigned a type, which is 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. Routes learned from the network by a routing protocol are normally installed into the connected routing table(s) and, conversely, routes from the connected routing table(s) are normally injected into the routing protocol. However, routing protocol implementations **MAY** specify rules that restrict this exchange of routes in either direction (or both directions).

A routing table is connected to a routing protocol instance by creating a corresponding entry in the "connected-routing-table" list. If such an entry is not configured for an address family, then the main routing table **MUST** be used as the connected routing table for this address family.

In addition, two independent route filters (see Section 4.5) may be configured for each connected routing table to apply client-defined policies controlling 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 the routing protocol instance to the connected routing table,
- o export filter controls which routes the routing protocol instance receives from the connected routing table.

Note that 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 implement 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. The "direct" pseudo-protocol MUST always be connected to the main routing tables of all supported address families. Unlike other routing protocol types, this connection cannot be changed in the configuration. Direct routes MAY be filtered before they appear in the main 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 configuration will have exactly one instance per logical router.

Static routes are configured within the "static-routes" container, see Figure 3.

```
+--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?
```

Figure 3: Structure of "static-routes" subtree.

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 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 state data by augmenting the definitions of the nodes

/rt:routing-tables/rt:routing-table/rt:route

and

/rt:active-route/rt:output/rt:route,

and possibly other places in the configuration, state data and RPC input or output.

- o Configuration parameters and state data for the new protocol can be defined by augmenting the "routing-protocol" data node.
- o Per-interface configuration, including activation of the routing protocol on individual interfaces, can use references to entries in the list of router interfaces (rt:interface).

By using the "when" statement, the augmented configuration parameters and state data specific to the new protocol SHOULD be made conditional and valid only if the value of "rt:type" is equal to the new protocol's identity. It is also RECOMMENDED that the protocol-specific data be encapsulated in appropriately named containers.

The above steps are implemented by the example YANG module for the RIP routing protocol in Appendix B.

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 two 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.

4.6. RPC Operations

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

- o active-route: query the routing system for the active route(s) that are currently used for sending datagrams to a destination host whose address is passed as an input parameter.
- o route-count: retrieve the total number of entries in a routing table.

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.

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: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: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 network prefixes or masks on network layer interfaces. Configuration of these parameters on an enabled

interface MUST result in an immediate creation of the corresponding direct route. The destination prefix of this route is set according to the configured IP address and network prefix/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@2013-02-23.yang"

```
module ietf-routing {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-routing";  
  
    prefix "rt";  
  
    import ietf-yang-types {  
        prefix "yang";  
    }  
  
    import ietf-interfaces {  
        prefix "if";  
    }  
  
    import iana-afn-safi {  
        prefix "ianaaf";  
    }  
  
    organization  
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
  
    contact  
        "WG Web: <http://tools.ietf.org/wg/netmod/>  
        WG List: <mailto:netmod@ietf.org>  
  
        WG Chair: David Kessens  
        <mailto:david.kessens@nsn.com>  
  
        WG Chair: Juergen Schoenwaelder  
        <mailto:j.schoenwaelder@jacobs-university.de>  
  
        Editor: Ladislav Lhotka  
        <mailto:lhotka@nic.cz>  
        ";  
  
    description  
        "This 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 2013-02-23 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Management";
}

/* Identities */

identity router-type {
  description
    "Base identity from which router type identities are derived.

    It is primarily intended for discriminating among different
    types of logical routers or router virtualization.
    ";
}

identity standard-router {
  base router-type;
  description
    "This identity represents a standard router.";
}

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.";
    }

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

    typedef route-filter-ref {
        type leafref {
            path "/rt:routing/rt:route-filters/rt:route-filter/rt:name";
        }
    }
```

```
    description
      "This type is used for leafs that reference a route filter.";
  }

  /* Groupings */

  grouping afn-safi {
    description
      "This grouping provides two parameters specifying address
      family and subsequent address family.";
    leaf address-family {
      type ianaaf:address-family;
      mandatory "true";
      description
        "Address family.";
    }
    leaf safi {
      type ianaaf:subsequent-address-family;
      mandatory "true";
      description
        "Subsequent address family.";
    }
  }

  grouping route-content {
    description
      "Generic parameters of routes.";
    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 router instance with 'router-name' doesn't exist, then
```


this operation SHALL fail with error-tag 'data-missing' and error-app-tag 'router-not-found'.

If no active route for 'destination-address' exists, no output is returned - the server SHALL send an <rpc-reply> containing a single element <ok>.

```
";  
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 {  
    description  
      "Network layer destination address.  
  
      Address family specific modules MUST augment this  
      container with a leaf named 'address'.  
";  
    uses afn-safi;  
  }  
}  
output {  
  list route {  
    description  
      "List of active routes.  
  
      Route contents specific for each address family is  
      expected be defined through augmenting.  
";  
    uses afn-safi;  
    uses route-content;  
  }  
}  
}  
  
rpc route-count {  
  description  
    "Return the current number of routes in a routing table.  
  
    Parameters:  
  
    1. 'routing-table-name'.  
  
    If the routing table with the name specified in
```

```
        'routing-table-name' doesn't exist, then this operation SHALL
        fail with error-tag 'data-missing' and error-app-tag
        'routing-table-not-found'.
    ";
    input {
        leaf routing-table {
            type routing-table-ref;
            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";
        description
            "Each list entry is a container for configuration and state
            data of a single (logical) router instance.
            ";
        leaf name {
            type string;
            description
                "An arbitrary name of the router instance.";
        }
        leaf type {
            type identityref {
                base router-type;
            }
            default "rt:standard-router";
            description
                "This leaf specifies the router type.

                It is primarily intended as a means for discriminating
                among different types of logical routers, route
```

```
        virtualization, master-slave arrangements etc., while
        keeping all such router instances in the same flat list.
    ";
}
leaf enabled {
    type boolean;
    default "true";
    description
        "Enable/disable the router instance.

        If this parameter is false, the parent router instance is
        disabled, despite any other configuration that might be
        present.
    ";
}
leaf router-id {
    type yang:dotted-quad;
    description
        "Global router ID - 32-bit number in the form of a dotted
        quad.

        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.";
}
container main-routing-tables {
    description
        "Main routing tables used by the router instance.";
    list main-routing-table {
        must "address-family=/routing/routing-tables/"
            + "routing-table[name=current()/name]/"
            + "address-family and safi=/routing/routing-tables/"
            + "routing-table[name=current()/name]/safi" {
            error-message "Address family mismatch.";
            description
                "The entry's address family MUST match that of the
                referenced routing table.";
        }
        key "address-family safi";
        description

```

"Each list entry specifies the main routing table for one address family.

The main routing table is operationally connected to all routing protocols for which a connected routing table has not been explicitly configured.

The 'direct' pseudo-protocol is always connected to the main routing table.

Address families that don't have their entry in this list MUST NOT be used in the rest of the router instance configuration.

```

";
uses afn-safi;
leaf name {
  type routing-table-ref;
  description
    "Name of an existing routing table to be used as the
    main routing table for the given router instance and
    address family.";
}
}
}
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
    "An arbitrary name of the routing protocol instance.";
}
leaf description {
  type string;
  description
    "Textual description of the routing protocol
    instance.";
}
leaf enabled {
  type boolean;
  default "true";
  description
    "Enable/disable the routing protocol instance.

    If this parameter is false, the parent routing
    protocol instance is disabled, despite any other
    configuration that might be present.
    ";
}
leaf type {
  type identityref {
    base routing-protocol;
  }
  mandatory "true";
  description
    "Type of the routing protocol - an identity derived
    from the 'routing-protocol' base identity.";
}
container connected-routing-tables {
  description
    "Container for connected routing tables.
    ";
  list connected-routing-table {
    must "not(/routing/routing-tables/"
      + "routing-table[name=current()/"
      + "preceding-sibling::connected-routing-table/"
      + "name and address-family=/routing/routing-tables/"
      + "routing-table[name=current()/name]/"
      + "address-family and safi=/routing/routing-tables/"
      + "routing-table[name=current()/name]/safi])" {
      error-message "Duplicate address family for "
        + "connected routing tables.";
    }
    description
      "For each AFN/SAFI pair there MUST NOT be more than
      one connected routing table.";
  }
}
```

```
}
key "name";
description
  "List of routing tables to which the routing protocol
   instance is connected (at most one routing table per
   address family).

   If no connected routing table is configured for an
   address family, the routing protocol MUST be
   operationally connected to the main routing table
   for that address family.

   ";
leaf name {
  type routing-table-ref;
  must "../.../type != 'rt:direct' or "
    + "../.../.../main-routing-tables/ "
    + "main-routing-table/name=." {
    error-message "The 'direct' protocol can be "
      + "connected only to a main routing "
      + "table.";
    description
      "For the 'direct' pseudo-protocol, the connected
       routing table must always be a main routing
       table.";
  }
  description
    "Name of an existing routing table.";
}
leaf import-filter {
  type route-filter-ref;
  description
    "Reference to a route filter that is used for
     filtering routes passed from this routing protocol
     instance to the routing table specified by the
     '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 route-filter-ref;
  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.

The 'direct' and 'static' pseudo-protocols accept no routes from any routing table.

```
    ";
  }
}
}
container static-routes {
  when "../type='rt:static'" {
    description
      "This container is only valid for the 'static'
      routing protocol.";
  }
  description
    "Configuration of 'static' pseudo-protocol.

    Address family specific modules augment this node with
    their lists of routes.
    ";
}
}
}
}
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 belong to
      the same address family.";
    leaf name {
      type string;
      description
        "An arbitrary 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 (state data).";
list route {
  description
    "A routing table entry. This data node MUST be
    augmented with information specific for routes of each
    address family.";
  uses route-content;
  leaf source-protocol {
    type string;
    mandatory "true";
    description
      'Routing protocol instance from which the route
      originated.

      It must be either "direct" or the name of a
      configured routing protocol instance.
      ';
```

```
  }
  leaf last-updated {
    type yang:date-and-time;
    description
      "Time stamp of the last modification of the route. If
      the route was never modified, it is the time when
      the route was inserted into the routing table.";
  }
}
}
container recipient-routing-tables {
  description
    "Container for recipient routing tables.";
  list recipient-routing-table {
    must "name != ../../name" {
      error-message "Source and recipient routing tables "
        + "are identical.";
    }
    description
      "A routing table MUST NOT appear among its recipient
      routing tables.";
  }
  must "/routing/routing-tables/"
    + "routing-table[name=current()/name]/"
    + "address-family=../../address-family and /routing/"
    + "routing-tables/routing-table[name=current()/name]/"
    + "safi=../../safi" {
    error-message "Address family mismatch.";
    description
      "Address family of the recipient routing table MUST
      match the source table.";
```



```
    }
    key "name";
    description
        "List of routing tables that receive routes from this
        routing table.";
    leaf name {
        type routing-table-ref;
        description
            "The name of the recipient routing table.";
    }
    leaf filter {
        type route-filter-ref;
        description
            "A route filter which is applied to the routes passed
            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 and 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
                "An arbitrary name of the route filter.";
        }
        leaf description {
            type string;
            description
                "Textual description of the route filter.";
        }
        leaf type {
            type identityref {
                base route-filter;
            }
        }
    }
}
```

```
        mandatory "true";
        description
            "Type of the route-filter - an identity derived from the
             'route-filter' base identity.";
    }
}
}
```

<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@2013-02-23.yang"

```
module ietf-ipv4-unicast-routing {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing";  
  
    prefix "v4ur";  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    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 state data for IPv4 unicast routing.  
  
        Copyright (c) 2012 IETF Trust and the persons identified as  
        authors of the code. All rights reserved.  
  
        Redistribution and use in source and binary forms, with or  
        without modification, is permitted pursuant to, and subject to
```

the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

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

```
";

revision 2013-02-23 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Management";
}

/* 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 "rt:address-family='ipv4' and rt: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 "rt:address-family='ipv4' and rt: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 by 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: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@2013-02-23.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 state data for IPv6 unicast routing.

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

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

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

";

```
revision 2013-02-23 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Management";
}
```

/* 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 "rt:address-family='ipv6' and rt: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 "rt:address-family='ipv6' and rt: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[if:name=current()/rt:name]/"
    + "ip:ipv6/ip:enabled='true'" {
    description
      "This augment is only valid for router interfaces with
      enabled IPv6.";
  }
  description
    "IPv6-specific parameters of router interfaces.";
  container ipv6-router-advertisements {
    description
      "Parameters of IPv6 Router Advertisements.";
    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.";
      reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvSendAdvertisements.";
    }
    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.";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        MaxRtrAdvInterval.";
}
leaf min-rtr-adv-interval {
    type uint16 {
        range "3..1350";
    }
    units "seconds";
    must ". <= 0.75 * ../max-rtr-adv-interval" {
        description
            "The value MUST NOT be greater than 75 % of
            'max-rtr-adv-interval'.";
    }
    description
        "The minimum time allowed between sending unsolicited
        multicast Router Advertisements from the interface.

        The default value to be used operationally if this leaf is
        not configured is determined as follows:

        - 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.
        ";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        MinRtrAdvInterval.";
}
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.";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvManagedFlag.";
}
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.";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvOtherConfigFlag.";
}
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.";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvLinkMTU.";
}
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).";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvReachableTime.";
}
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).";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvRetransTimer.";
}
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

"RFC 4861: Neighbor Discovery for IP version 6 (IPv6) - AdvCurHopLimit.

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.

 If this parameter is not configured, a value of 3 * max-rtr-adv-interval SHOULD be used.

 ";

 reference

 "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) - AdvDefaultLifeTime.";

}

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.

Prefixes that do not have their entries in the child 'prefix' list are advertised with the default values of all parameters.

The link-local prefix SHOULD NOT be included in the list of advertised prefixes.

```

";
reference
  "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
  AdvPrefixList.";
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 can 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.
        ";
      reference
        "RFC 4861: Neighbor Discovery for IP version 6
        (IPv6) - AdvValidLifetime.";
    }
  }
}
```



```

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 by 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: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

Configuration and state data conforming to the core routing data model (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 belonging to the core routing data model 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:

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

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

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

/routing/routing-tables/routing-table This list specifies the configured routing tables used by the device.

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, Wes Hardaker, Andrew McGregor, Thomas Morin, Tom Petch, Bruno Rijsman, Juergen Schoenwaelder, Phil Shafer, 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-04 (work in progress), June 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.
- [RFC6021bis] Schoenwaelder, J., Ed., "Common YANG Data Types", draft-ietf-netmod-rfc6021-bis-00 (work in progress), February 2013.
- [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-09 (work in progress), February 2013.
- [YANG-IP] Bjorklund, M., "A YANG Data Model for IP Configuration", draft-ietf-netmod-ip-cfg-09 (work in progress), February 2013.

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. The Complete Data Tree

This appendix presents the complete data tree of the core routing data model. See Section 2.2 for an explanation of symbols. Data type of every leaf node is shown near the right end of the corresponding line.

```

+--rw routing
  +--rw router [name]
    +--rw name string
    +--rw type? identityref
    +--rw enabled? boolean
    +--rw router-id? yang:dotted-quad
    +--rw description? string
    +--rw main-routing-tables
      +--rw main-routing-table [address-family safi]
        +--rw address-family ianaaf:address-family
        +--rw safi ianaaf:subsequent-address-family
        +--rw name? routing-table-ref
    +--rw interfaces
      +--rw interface [name]
        +--rw name if:interface-ref
        +--rw v6ur:ipv6-router-advertisements
          +--rw v6ur:send-advertisements? boolean
          +--rw v6ur:max-rtr-adv-interval? uint16
          +--rw v6ur:min-rtr-adv-interval? uint16
          +--rw v6ur:managed-flag? boolean
          +--rw v6ur:other-config-flag? boolean
          +--rw v6ur:link-mtu? uint32
          +--rw v6ur:reachable-time? uint32
          +--rw v6ur:retrans-timer? uint32
          +--rw v6ur:cur-hop-limit? uint8
          +--rw v6ur:default-lifetime? uint16
          +--rw v6ur:prefix-list
            +--rw v6ur:prefix [prefix-spec]
              +--rw v6ur:prefix-spec inet:ipv6-prefix
              +--rw (control-adv-prefixes)?
                +--:(no-advertise)
                  +--rw v6ur:no-advertise? empty
                +--:(advertise)
                  +--rw v6ur:valid-lifetime? uint32
                  +--rw v6ur:on-link-flag? boolean
                  +--rw v6ur:preferred-lifetime? uint32
                  +--rw v6ur:autonomous-flag? boolean
        +--rw routing-protocols
          +--rw routing-protocol [name]
            +--rw name string
            +--rw description? string

```

```

    +---rw enabled?                boolean
    +---rw type                    identityref
    +---rw connected-routing-tables
    |   +---rw connected-routing-table [name]
    |   |   +---rw name            routing-table-ref
    |   |   +---rw import-filter?  route-filter-ref
    |   |   +---rw export-filter?  route-filter-ref
    +---rw static-routes
    |   +---rw v4ur:ipv4
    |   |   +---rw v4ur:route [id]
    |   |   |   +---rw v4ur:id            uint32
    |   |   |   +---rw v4ur:description?  string
    |   |   |   +---rw v4ur:outgoing-interface?  if:interface-ref
    |   |   |   +---rw v4ur:dest-prefix    inet:ipv4-prefix
    |   |   |   +---rw v4ur:next-hop?      inet:ipv4-address
    |   +---rw v6ur:ipv6
    |   |   +---rw v6ur:route [id]
    |   |   |   +---rw v6ur:id            uint32
    |   |   |   +---rw v6ur:description?  string
    |   |   |   +---rw v6ur:outgoing-interface?  if:interface-ref
    |   |   |   +---rw v6ur:dest-prefix    inet:ipv6-prefix
    |   |   |   +---rw v6ur:next-hop?      inet:ipv6-address
    +---rw routing-tables
    |   +---rw routing-table [name]
    |   |   +---rw name            string
    |   |   +---rw address-family  ianaaf:address-family
    |   |   +---rw safi            ianaaf:subsequent-address-family
    |   |   +---rw description?    string
    |   +---ro routes
    |   |   +---ro route
    |   |   |   +---ro outgoing-interface?  if:interface-ref
    |   |   |   +---ro source-protocol      string
    |   |   |   +---ro last-updated?        yang:date-and-time
    |   |   |   +---ro v4ur:dest-prefix?    inet:ipv4-prefix
    |   |   |   +---ro v4ur:next-hop?       inet:ipv4-address
    |   |   |   +---ro v6ur:dest-prefix?    inet:ipv6-prefix
    |   |   |   +---ro v6ur:next-hop?       inet:ipv6-address
    +---rw recipient-routing-tables
    |   +---rw recipient-routing-table [name]
    |   |   +---rw name            routing-table-ref
    |   |   +---rw filter?        route-filter-ref
    +---rw route-filters
    |   +---rw route-filter [name]
    |   |   +---rw name            string
    |   |   +---rw description?    string
    |   |   +---rw type            identityref

```

Appendix B. 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.

```
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  
            "This grouping defines RIP-specific route attributes.";  
        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:routing-tables/rt:routing-table/rt:routes/"  
        + "rt:route" {
```



```
    description
      "RIP-specific route attributes.";
    uses route-content;
  }

  augment "/rt:active-route/rt:output/rt:route" {
    description
      "RIP-specific route attributes in the output of 'active-route'
      RPC.";
    uses route-content;
  }

  augment "/rt:routing/rt:router/rt:routing-protocols/"
    + "rt:routing-protocol" {
    when "rt:type = 'rip:rip'" {
      description
        'This augment is only valid for a routing protocol instance
        of type "rip".';
    }
    container rip {
      description
        "RIP instance configuration.";
      container interfaces {
        description
          "Per-interface RIP configuration.";
        list interface {
          key "name";
          description
            "RIP is enabled on interfaces that have an entry in this
            list, unless 'enabled' is set to 'false' for that
            entry.";
          leaf name {
            type leafref {
              path "../..../rt:interfaces/rt:interface/"
                + "rt:name";
            }
          }
          leaf enabled {
            type boolean;
            default "true";
          }
          leaf metric {
            type rip-metric;
            default "1";
          }
        }
      }
      leaf update-interval {
```

```
        type uint8 {  
            range "10..60";  
        }  
        units "seconds";  
        default "30";  
        description  
            "Time interval between periodic updates.";  
    }  
}  
}
```

Appendix C. Example: NETCONF <get> Reply

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 4: 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.

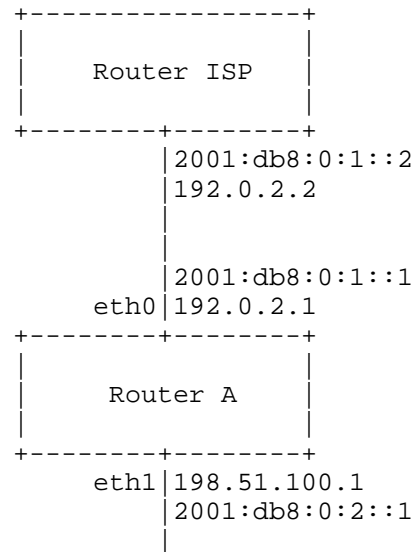


Figure 4: Example network configuration

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>eth0</if:location>
      <ip:ipv4>
        <ip:address>
          <ip:ip>192.0.2.1</ip:ip>
          <ip:prefix-length>24</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
      </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:forwarding>true</ip:forwarding>
        <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>eth1</if:location>
      <ip:ipv4>
        <ip:address>
          <ip:ip>198.51.100.1</ip:ip>
          <ip:prefix-length>24</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
      </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:forwarding>true</ip:forwarding>
      </ip:ipv6>
    </if:interface>
  </if:interfaces>
</data>
```

```
<ip:autoconf>
  <ip:create-global-addresses>>false</ip:create-global-addresses>
</ip:autoconf>
</ip:ipv6>
</if:interface>
</if:interfaces>
<rt:routing>
  <rt:router>
    <rt:name>rtr0</rt:name>
    <rt:router-id>192.0.2.1</rt:router-id>
    <rt:description>Router A</rt:description>
    <rt:main-routing-tables>
      <rt:main-routing-table>
        <rt:address-family>ipv4</rt:address-family>
        <rt:safi>nlri-unicast</rt:safi>
        <rt:name>ipv4-unicast</rt:name>
      </rt:main-routing-table>
      <rt:main-routing-table>
        <rt:address-family>ipv6</rt:address-family>
        <rt:safi>nlri-unicast</rt:safi>
        <rt:name>ipv6-unicast</rt:name>
      </rt:main-routing-table>
    </rt:main-routing-tables>
    <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>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-tables>
  <rt:routing-table>
    <rt:name>ipv4-unicast</rt:name>
    <rt:address-family>ipv4</rt:address-family>
    <rt:safi>nlri-unicast</rt:safi>
    <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:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
      </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:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
      </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:last-updated>2012-10-02T18:02:45+01:00</rt:last-updated>
      </rt:route>
    </rt:routes>
  </rt:routing-table>
  <rt:routing-table>
    <rt:name>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:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
</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:last-updated>2012-10-02T17:11:27+01:00</rt:last-updated>
</rt:route>
<rt:route>
  <v6ur:dest-prefix>::/0</v6ur:dest-prefix>
  <v6ur:next-hop>2001:db8:0:1::2</v6ur:next-hop>
  <rt:source-protocol>st0</rt:source-protocol>
  <rt:last-updated>2012-10-02T18:02:45+01:00</rt:last-updated>
</rt:route>
</rt:routes>
</rt:routing-table>
</rt:routing-tables>
</rt:routing>
</data>
</rpc-reply>
```

Appendix D. Change Log

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

D.1. Changes Between Versions -08 and -09

- o Fixed "must" expresion for "connected-routing-table".
- o Simplified "must" expression for "main-routing-table".
- o Moved per-interface configuration of a new routing protocol under 'routing-protocol'. This also affects the 'example-rip' module.

D.2. Changes Between Versions -07 and -08

- o Changed reference from RFC6021 to RFC6021bis.

D.3. Changes Between Versions -06 and -07

- o The contents of <get-reply> in Appendix C was updated: "eth[01]" is used as the value of "location", and "forwarding" is on for both interfaces and both IPv4 and IPv6.
- o The "must" expression for "main-routing-table" was modified to avoid redundant error messages reporting address family mismatch when "name" points to a non-existent routing table.
- o The default behavior for IPv6 RA prefix advertisements was clarified.
- o Changed type of "rt:router-id" to "ip:dotted-quad".
- o Type of "rt:router-id" changed to "yang:dotted-quad".
- o Fixed missing prefixes in XPath expressions.

D.4. Changes Between Versions -05 and -06

- o Document title changed: "Configuration" was replaced by "Management".
- o New typedefs "routing-table-ref" and "route-filter-ref".
- o Double slashes "/" were removed from XPath expressions and replaced with the single "/".
- o Removed uniqueness requirement for "router-id".

- o Complete data tree is now in Appendix A.
- o Changed type of "source-protocol" from "leafref" to "string".
- o Clarified the relationship between routing protocol instances and connected routing tables.
- o Added a must constraint saying that a routing table connected to the direct pseudo-protocol must not be a main routing table.

D.5. Changes Between Versions -04 and -05

- o Routing tables are now global, i.e., "routing-tables" is a child of "routing" rather than "router".
- o "must" statement for "static-routes" changed to "when".
- o Added "main-routing-tables" containing references to main routing tables for each address family.
- o Removed the defaults for "address-family" and "safi" and made them mandatory.
- o Removed the default for route-filter/type and made this leaf mandatory.
- o If there is no active route for a given destination, the "active-route" RPC returns no output.
- o Added "enabled" switch under "routing-protocol".
- o Added "router-type" identity and "type" leaf under "router".
- o Route attribute "age" changed to "last-updated", its type is "yang:date-and-time".
- o The "direct" pseudo-protocol is always connected to main routing tables.
- o Entries in the list of connected routing tables renamed from "routing-table" to "connected-routing-table".
- o Added "must" constraint saying that a routing table must not be its own recipient.

D.6. 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".

D.7. 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".

D.8. Changes Between Versions -01 and -02

- o Added module "ietf-ipv6-unicast-routing".
- o The example in Appendix C now uses IP addresses from blocks reserved for documentation.
- o Direct routes appear by default in the forwarding 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.

D.9. 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

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

M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
February 11, 2013

A YANG Data Model for SNMP Configuration
draft-ietf-netmod-snmp-cfg-01

Abstract

This document defines a collection of YANG definitions for configuring SNMP engines.

Status of this Memo

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

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

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

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

Copyright Notice

Copyright (c) 2013 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. Data Model	4
2.1. General Considerations	4
2.2. Common Definitions	4
2.3. Engine Configuration	4
2.4. Target Configuration	5
2.5. Notification Configuration	6
2.6. Proxy Configuration	7
2.7. Community Configuration	7
2.8. View-based Access Control Model Configuration	9
2.9. User-based Security Model Configuration	9
2.10. Transport Security Model Configuration	11
2.11. Transport Layer Security Transport Model Configuration . .	12
2.12. Secure Shell Transport Model Configuration	13
3. Definitions	14
3.1. Module 'ietf-snmp'	14
3.2. Submodule 'ietf-snmp-common'	16
3.3. Submodule 'ietf-snmp-engine'	20
3.4. Submodule 'ietf-snmp-target'	23
3.5. Submodule 'ietf-snmp-notification'	27
3.6. Submodule 'ietf-snmp-proxy'	31
3.7. Submodule 'ietf-snmp-community'	33
3.8. Submodule 'ietf-snmp-vacm'	38
3.9. Submodule 'ietf-snmp-usm'	44
3.10. Submodule 'ietf-snmp-tsm'	48
3.11. Submodule 'ietf-snmp-tls'	50
3.12. Submodule 'ietf-snmp-ssh'	56
4. IANA Considerations	59
5. Security Considerations	61
6. Acknowledgments	62
7. References	63
7.1. Normative References	63
7.2. Informative References	63
Appendix A. Example configurations	65
A.1. Engine Configuration Example	65
A.2. Community Configuration Example	65
A.3. User-based Security Model Configuration Example	66
A.4. Target and Notification Configuration Example	67
A.5. Proxy Configuration Example	69
A.6. View-based Access Control Model Configuration Example . .	71
A.7. Transport Layer Security Transport Model Configuration Example	73
Authors' Addresses	75

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of SNMP engines. The configuration model is consistent with the MIB modules defined in [RFC3411], [RFC3412], [RFC3413], [RFC3414], [RFC3415], [RFC3418], [RFC3584], [RFC5591], [RFC5592], and [RFC6353] but takes advantage of YANG's ability to define hierarchical configuration data models. The structure of the model has been derived from existing proprietary configuration models implemented as command line interfaces.

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].

2. Data Model

In order to preserve the modularity of SNMP, the YANG configuration data model is organized in a set of YANG submodules, all sharing the same module namespace. This allows to add configuration support for additional SNMP features while keeping the number of namespaces that have to be dealt with down to a minimum.

2.1. General Considerations

Most YANG nodes are mapped 1-1 to the corresponding MIB object. The "reference" statement is used to indicate which corresponding MIB object the YANG node is mapped to. When there is not a simple 1-1 mapping, the "description" statement explains the mapping.

2.2. Common Definitions

The submodule "ietf-snmp-common" defines a set of common typedefs, features, and the top-level container "snmp". All configuration parameters defined in the other submodules are organized under this top-level container.

This submodule defines five YANG features:

proxy: A server implements this feature if it can act as an SNMP Proxy.

notification-filter: A server implements this feature if it supports SNMP notification filtering.

tsm: A server implements this feature if it supports the Transport Security Model (tsm) [RFC5591].

sshtm: A server implements this feature if it supports the Secure Shell (SSH) Transport Model (sshtm) [RFC5592].

tlstm: A server implements this feature if it supports the Transport Layer Security (TLS) Transport Model (tlstm) [RFC6353].

2.3. Engine Configuration

The submodule "ietf-snmp-engine", which defines configuration parameters that are specific to SNMP engines, has the following structure:

```

+--rw snmp
  +--rw engine
    +--rw enabled?      boolean
    +--rw listen
      | +--rw udp [ip port]
      |   +--rw ip      inet:ip-address
      |   +--rw port    inet:port-number
    +--rw version
      | +--rw v1?      empty
      | +--rw v2c?     empty
      | +--rw v3?     empty
    +--rw engine-id?   snmp:engine-id

```

The leaf `"/snmp/engine/enabled"` can be used to enable/disable an SNMP engine.

The container `"/snmp/engine/listen"` provides configuration of the transport endpoints the engine is listening to. In this submodule, SNMP over UDP is defined. TLS and Datagram Transport Layer Security (DTLS) are also supported, defined in `"ietf-snmp-tls"` (Section 2.11). The `"listen"` container is expected to be augmented for other transports.

The `"/snmp/engine/version"` container can be used to enable/disable the different message processing models.

2.4. Target Configuration

The submodule `"ietf-snmp-target"`, which defines configuration parameters that correspond to the objects in SNMP-TARGET-MIB, has the following structure:

```

+--rw snmp
  +--rw target [name]
    +--rw name      snmp:identifier
    +--rw (transport)
      | +--:(udp)
      |   +--rw udp
      |     +--rw ip      inet:ip-address
      |     +--rw port?   inet:port-number
      |     +--rw prefix-length? uint8
    +--rw tag*      snmp:identifier
    +--rw timeout?  uint32
    +--rw retries?  uint8
    +--rw (params)?

```

An entry in the list `"/snmp/target"` corresponds to an `"snmpTargetAddrEntry"`.

The "snmpTargetAddrTDomain" and "snmpTargetAddrTAddress" objects are mapped to transport-specific YANG nodes. Each transport is configured as a separate case in the "transport" choice. In this submodule, SNMP over UDP is defined. TLS and DTLS are also supported, defined in "ietf-snmp-tls" (Section 2.11). The "transport" choice is expected to be augmented for other transports.

In order to provide a simpler configuration model with less cross-references, the "target" list also inlines the "snmpTargetParamsEntry" pointed to by "snmpTargetAddrParams". This is accomplished with a choice "params", which is augmented by security model specific submodules, currently "ietf-snmp-community" (Section 2.7), "ietf-snmp-usm" (Section 2.9), and "ietf-snmp-tls" (Section 2.11).

The YANG model does not define a separate list that maps directly to "snmpTargetParamsTable". Since "snmpProxyTable" also has a reference to this table, "snmpProxyTable" also has a choice "params" which is augmented by security model specific submodules (Section 2.6).

2.5. Notification Configuration

The submodule "ietf-snmp-notification", which defines configuration parameters that correspond to the objects in SNMP-NOTIFICATION-MIB, has the following structure:

```
+--rw snmp
  +--rw notify [name]
    |   +--rw name      snmp:identifier
    |   +--rw tag       snmp:identifier
    |   +--rw type?     enumeration
  +--rw notify-filter-profile [name]
    |   +--rw name      snmp:identifier
    |   +--rw include*   wildcard-object-identifier
    |   +--rw exclude*   wildcard-object-identifier
  +--rw enable-authen-traps?   boolean
```

It also augments the "target" list defined in the "ietf-snmp-target" submodule (Section 2.4) with one leaf:

```
+--rw snmp
  +--rw target [name]
    ...
    +--rw notify-filter-profile?   leafref
```

An entry in the list "/snmp/notify" corresponds to an "snmpNotifyEntry".

An entry in the list `"/snmp/notify-filter-profile"` corresponds to an `"snmpNotifyFilterProfileEntry"`. In the MIB, there is a sparse relationship between `"snmpTargetParamsTable"` and `"snmpNotifyFilterProfileTable"`. In the YANG model, this sparse relationship is represented with a leafref leaf `"notify-filter-profile"` in the `"/snmp/target"` list, which refers to an entry in the `"/snmp/notify-filter-profile"` list.

The `"snmpNotifyFilterTable"` is represented as a list `"filter"` within the `"/snmp/notify-filter-profile"` list.

2.6. Proxy Configuration

The submodule `"ietf-snmp-proxy"`, which defines configuration parameters that correspond to the objects in `SNMP-PROXY-MIB`, has the following structure:

```

+--rw snmp
  +--rw proxy [name]
    +--rw name                snmp:identifier
    +--rw type                enumeration
    +--rw context-engine-id   snmp:engine-id
    +--rw context-name?      snmp:context-name
    +--rw params-in
      | +--rw (params)
      +--rw single-target-out? snmp:identifier
      +--rw multiple-target-out? snmp:identifier

```

An entry in the list `"/snmp/proxy"` corresponds to an `"snmpProxyEntry"`.

Like the `"target"` list (Section 2.4), the `"proxy"` list inlines the `"snmpTargetParamsEntry"` pointed to by `"snmpProxyTargetParamsIn"`. This is accomplished with a choice `"params"`, which is augmented by security model specific submodules, currently `"ietf-snmp-community"` (Section 2.7), `"ietf-snmp-usm"` (Section 2.9), and `"ietf-snmp-tls"` (Section 2.11).

2.7. Community Configuration

The submodule `"ietf-snmp-community"`, which defines configuration parameters that correspond to the objects in `SNMP-COMMUNITY-MIB`, has the following structure:

```

+--rw snmp
  +--rw community [index]
    +--rw index          snmp:identifier
    +--rw (name)?
      | +--:(text-name)
      | | +--rw text-name?      string
      | +--:(binary-name)
      | | +--rw binary-name?    binary
    +--rw security-name   snmp:security-name
    +--rw engine-id?     snmp:engine-id
    +--rw context?       snmp:context-name
    +--rw target-tag?    snmp:identifier

```

It also augments the `"/snmp/target/params"` and `"/snmp/proxy/params-in/params"` choices with nodes for the Community-Based Security Model used by SNMPv1 and SNMPv2c:

```

+--rw snmp
  +--rw target [name]
    | ...
    | +--rw (params)?
    | | +--:(v1)
    | | | +--rw v1
    | | | | +--rw security-name   snmp:security-name
    | | +--:(v2c)
    | | | +--rw v2c
    | | | | +--rw security-name   snmp:security-name
    | +--rw mms?      union
  +--rw proxy
    +--rw params-in
      +--rw params
        +--:(v1)
        | +--rw v1
        | | +--rw security-name   snmp:security-name
        +--:(v2c)
        | +--rw v2c
        | | +--rw security-name   snmp:security-name

```

An entry in the list `"/snmp/community"` corresponds to an `"snmpCommunityEntry"`.

When a case `"v1"` or `"v2c"` is chosen, it implies a `snmpTargetParamsMpmModel` 0 (SNMPv1) or 1 (SNMPv2), and a `snmpTargetParamsSecurityModel` 1 (SNMPv1) or 2 (SNMPv2), respectively. Both cases implies a `snmpTargetParamsSecurityLevel` of `noAuthNoPriv`.

2.8. View-based Access Control Model Configuration

The submodule "ietf-snmp-vacm", which defines configuration parameters that correspond to the objects in SNMP-VIEW-BASED-ACM-MIB, has the following structure:

```

+--rw snmp
  +--rw vacm
    +--rw group [name]
      +--rw name          group-name
      +--rw member [security-name]
        +--rw security-name    snmp:security-name
        +--rw security-model*  snmp:security-model
      +--rw access [context security-model security-level]
        +--rw context          snmp:context-name
        +--rw context-match?   enumeration
        +--rw security-model    snmp:security-model-or-any
        +--rw security-level    snmp:security-level
        +--rw read-view?       view-name
        +--rw write-view?      view-name
        +--rw notify-view?     view-name
    +--rw view [name]
      +--rw name          view-name
      +--rw include*      snmp:wildcard-object-identifier
      +--rw exclude*      snmp:wildcard-object-identifier

```

The "vacmSecurityToGroupTable" and "vacmAccessTable" are mapped to a structure of nested lists in the YANG model. Groups are defined in the list "/snmp/vacm/group" and for each group there is a sublist "member" that maps to "vacmSecurityToGroupTable", and a sublist "access" that maps to "vacmAccessTable".

MIB views are defined in the list "/snmp/vacm/view" and for each MIB view there is a leaf-list of included subtree families and a leaf-list of excluded subtree families. This is more compact and thus a more readable representation of the "vacmViewTreeFamilyTable".

2.9. User-based Security Model Configuration

The submodule "ietf-snmp-usm", which defines configuration parameters that correspond to the objects in SNMP-USER-BASED-SM-MIB, has the following structure:

```

+--rw snmp
  +--rw usm
    +--rw local
      | +--rw user [name]
      |   +-- {common user params}
    +--rw remote [engine-id]
      +--rw engine-id snmp:engine-id
      +--rw user [name]
        +-- {common user params}

```

The "{common user params}" are:

```

+--rw name snmp:identifier
+--rw auth?
  | +--rw (protocol)
  |   +--:(md5)
  |     | +--rw md5
  |     |   +-- rw key string
  |     +--:(sha)
  |       +--rw sha
  |       +-- rw key string
+--rw priv?
  +--rw (protocol)
  +--:(des)
  | +--rw des
  |   +-- rw key string
  +--:(aes)
  +--rw aes
  +-- rw key string

```

It also augments the "/snmp/target/params" and "/snmp/proxy/params-in/params" choices with nodes for the SNMP User-based Security Model.

```

+--rw snmp
  +--rw target [name]
    ...
    | +--rw (params)?
    | | +--:(usm)
    | | | +--rw usm
    | | | | +--rw user-name          snmp:security-name
    | | | | +--rw security-level      security-level
    +--rw proxy [name]
      ...
      +--rw params-in
        +--rw (params)
          +--:(usm)
            +--rw usm
              +--rw user-name          snmp:security-name
              +--rw security-level      security-level

```

In the MIB, there is a single table with local and remote users, indexed by the engine id and user name. In the YANG model, there is one list of local users, and a nested list of remote users.

In the MIB, there are several objects related to changing the authentication and privacy keys. These objects are not present in the YANG model. However, the localized key can be changed. This implies that if the engine id is changed, all users keys need to be changed as well.

2.10. Transport Security Model Configuration

The submodule "ietf-snmp-tsm", which defines configuration parameters that correspond to the objects in SNMP-TSM-MIB, has the following structure:

```

+--rw snmp
  +--rw tsm
    +--rw use-prefix?  boolean

```

It also augments the "/snmp/target/params" and "/snmp/proxy/params-in/params" choices with nodes for the SNMP Transport Security Model.


```

+--rw snmp
  +--rw target [name]
    ...
    | +--rw (params)?
    |   +--:(tsm)
    |     +--rw tsm
    |       +--rw security-name      snmp:security-name
    |       +--rw security-level     security-level
  +--rw proxy [name]
    ...
    +--rw params-in
      +--rw (params)
        +--:(tsm)
          +--rw tsm
            +--rw security-name      snmp:security-name
            +--rw security-level     security-level

```

2.11. Transport Layer Security Transport Model Configuration

The submodule "ietf-snmp-tls", which defines configuration parameters that correspond to the objects in SNMP-TLS-TM-MIB, has the following structure:

```

+--rw snmp
  ...
  +--rw target [name]
    ...
    | +--rw (transport)
    |   ...
    |   +--:(tls)
    |     | +--rw tls
    |     |   +-- {common (d)tls transport params}
    |   +--:(dtls)
    |     | +--rw dtls
    |     |   +-- {common (d)tls transport params}
  +--rw tlstm
    +--rw cert-to-tm-security-name [id]
      +--rw id                               uint32
      +--rw fingerprint?                    tls-fingerprint
      +--rw map-type?                        identityref
      +--rw cert-specified-tm-security-name? admin-string

```

The "{common (d)tls transport params}" are:

```

+--rw ip?                inet:ip-address
+--rw port?              inet:port-number
+--rw client-fingerprint?  tls-fingerprint
+--rw server-fingerprint?  tls-fingerprint

```

```

+--rw server-identity?      admin-string

```

It also augments the `"/snmp/engine/listen"` container with objects for the D(TLS) transport endpoints:

```

+--rw snmp
  +--rw engine
    ...
    +--rw listen
      ...
      +--rw tls [ip port]
        | +--rw ip      inet:ip-address
        | +--rw port    inet:port-number
      +--rw dtls [ip port]
        +--rw ip      inet:ip-address
        +--rw port    inet:port-number

```

2.12. Secure Shell Transport Model Configuration

The submodule `"ietf-snmp-ssh"`, which defines configuration parameters that correspond to the objects in SNMP-SSH-TM-MIB, has the following structure:

```

+--rw snmp
  ...
  +--rw target [name]
    ...
    +--rw (transport)
      ...
      +--:(ssh)
        +--rw ssh
          +--rw ip      inet:host
          +--rw port?   inet:port-number
          +--rw username? string

```

It also augments the `"/snmp/engine/listen"` container with objects for the SSH transport endpoints:

```

+--rw snmp
  +--rw engine
    ...
    +--rw listen
      ...
      +--rw ssh [ip port]

```

3. Definitions

3.1. Module 'ietf-snmp'

<CODE BEGINS> file "ietf-snmp.yang"

```
module ietf-snmp {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-snmp";  
    prefix snmp;  
  
    include ietf-snmp-common {  
        revision-date 2013-02-11;  
    }  
    include ietf-snmp-engine {  
        revision-date 2012-06-05;  
    }  
    include ietf-snmp-target {  
        revision-date 2012-06-05;  
    }  
    include ietf-snmp-notification {  
        revision-date 2012-06-05;  
    }  
    include ietf-snmp-proxy {  
        revision-date 2012-06-05;  
    }  
    include ietf-snmp-community {  
        revision-date 2012-06-05;  
    }  
    include ietf-snmp-usm {  
        revision-date 2013-02-11;  
    }  
    include ietf-snmp-tsm {  
        revision-date 2012-06-05;  
    }  
    include ietf-snmp-vacm {  
        revision-date 2012-06-05;  
    }  
    include ietf-snmp-tls {  
        revision-date 2013-02-11;  
    }  
    include ietf-snmp-ssh {  
        revision-date 2012-11-26;  
    }  
  
    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>

  Editor: Juergen Schoenwaelder
          <mailto:j.schoenwaelder@jacobs-university.de>";

description
  "This module contains a collection of YANG definitions for
  configuring SNMP engines.

  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 2012-11-26 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}
```

<CODE ENDS>

3.2. Submodule 'ietf-snmp-common'

<CODE BEGINS> file "ietf-snmp-common.yang"

```
submodule ietf-snmp-common {  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
  
  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>  
  
    Editor: Juergen Schoenwaelder  
            <mailto:j.schoenwaelder@jacobs-university.de>";  
  
  description  
    "This submodule contains a collection of common YANG definitions  
    for configuring SNMP engines.  
  
    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 2013-02-11 {
    description
        "Initial revision.";
    reference
        "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

/* Collection of SNMP features */

feature proxy {
    description
        "A server implements this feature if it can act as an
        SNMP Proxy";
}

feature notification-filter {
    description
        "A server implements this feature if it supports SNMP
        notification filtering.";
}

feature tsm {
    description
        "A server implements this feature if it supports the
        Transport Security Model for SNMP.";
    reference
        "RFC5591: Transport Security Model for the
        Simple Network Management Protocol (SNMP)";
}

feature sshtm {
    description
        "A server implements this feature if it supports the
        Secure Shell Transport Model for SNMP.";
    reference
        "RFC5592: Secure Shell Transport Model for the
        Simple Network Management Protocol (SNMP)";
}
```

```

feature tlstm {
  description
    "A server implements this feature if it supports the
    Transport Layer Security Transport Model for SNMP.";
  reference
    "RFC6353: Transport Layer Security (TLS) Transport Model for
    the Simple Network Management Protocol (SNMP)";
}

/* Collection of SNMP specific data types */

typedef admin-string {
  type string {
    length "0..255";
  }
  description
    "Represents and SnmpAdminString as defined in RFC 3411.

    Note that the size of an SnmpAdminString is measured in
    octets, not characters.";
  reference "SNMP-FRAMEWORK-MIB.SnmpAdminString";
}

typedef identifier {
  type admin-string {
    length "1..32";
  }
  description
    "Identifiers are used to name items in the SNMP configuration
    data store.";
}

typedef context-name {
  type admin-string {
    length "0..32";
  }
  description
    "The context type represents an SNMP context name.";
  reference
    "RFC3411: An Architecture for Describing SNMP Management
    Frameworks";
}

typedef security-name {
  type admin-string {
    length "1..32";
  }
  description

```

```
    "The security-name type represents an SNMP security name.";
    reference
      "RFC3411: An Architecture for Describing SNMP Management
        Frameworks";
  }

  typedef security-model {
    type union {
      type enumeration {
        enum v1 { value 1; }
        enum v2c { value 2; }
        enum usm { value 3; }
        enum tsm { value 4; }
      }
      type int32 {
        range "1..2147483647";
      }
    }
    reference
      "RFC3411: An Architecture for Describing SNMP Management
        Frameworks";
  }

  typedef security-model-or-any {
    type union {
      type enumeration {
        enum any { value 0; }
      }
      type security-model;
    }
    reference
      "RFC3411: An Architecture for Describing SNMP Management
        Frameworks";
  }

  typedef security-level {
    type enumeration {
      enum no-auth-no-priv { value 1; }
      enum auth-no-priv { value 2; }
      enum auth-priv { value 3; }
    }
    reference
      "RFC3411: An Architecture for Describing SNMP Management
        Frameworks";
  }

  typedef engine-id {
    type yang:hex-string {
```



```
    pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2}){4,31}';
  }
  description
    "The Engine ID specified as a list of colon-specified hexa-
    decimal octets, e.g., '80:00:02:b8:04:61:62:63'.";
  reference
    "RFC3411: An Architecture for Describing SNMP Management
    Frameworks";
}

typedef wildcard-object-identifier {
  type string;
  description
    "The wildcard-object-identifier type represents an SNMP object
    identifier where subidentifiers can be given either as a label,
    in numeric form, or a wildcard, represented by a *.";
}

container snmp {
  description
    "Top-level container for SNMP related configuration and
    status objects.";
}
}
```

<CODE ENDS>

3.3. Submodule 'ietf-snmp-engine'

```
<CODE BEGINS> file "ietf-snmp-engine.yang"

submodule ietf-snmp-engine {

  belongs-to ietf-snmp {
    prefix snmp;
  }

  import ietf-inet-types {
    prefix inet;
  }

  include ietf-snmp-common;

  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>>

Editor: Juergen Schoenwaelder
<<mailto:j.schoenwaelder@jacobs-university.de>>;

description

"This submodule contains a collection of YANG definitions
for configuring SNMP engines.

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: A YANG Data Model for SNMP Configuration";  
}
```

```
augment /snmp:snmp {  
  container engine {
```

```
description
  "Configuration of the SNMP engine.";

leaf enabled {
  type boolean;
  default "false";
  description
    "Enables the SNMP engine.";
}

container listen {
  description
    "Configuration of the transport endpoints on which the
    engine listens. Submodules providing configuration for
    additional transports are expected to augment this
    container.";

  list udp {
    key "ip port";
    description
      "A list of IPv4 and IPv6 addresses and ports to which the
      engine listens.";

    leaf ip {
      type inet:ip-address;
      description
        "The IPv4 or IPv6 address on which the engine
        listens.";
    }
    leaf port {
      type inet:port-number;
      description
        "The UDP port on which the engine listens.";
    }
  }
}

container version {
  description
    "SNMP version used by the engine";
  leaf v1 {
    type empty;
  }
  leaf v2c {
    type empty;
  }
  leaf v3 {
    type empty;
  }
}
```

```
    }  
  }  
  
  leaf engine-id {  
    type snmp:engine-id;  
    description  
      "The local SNMP engine's administratively-assigned unique  
       identifier.  
  
       If this leaf is not set, the device automatically  
       calculates an engine id, as described in RFC 3411. A  
       server MAY initialize this leaf with the automatically  
       created value.";  
    reference "SNMP-FRAMEWORK-MIB.snmpEngineID";  
  }  
}  
}
```

<CODE ENDS>

3.4. Submodule 'ietf-snmp-target'

<CODE BEGINS> file "ietf-snmp-target.yang"

```
submodule ietf-snmp-target {  
  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
  
  import ietf-inet-types {  
    prefix inet;  
  }  
  
  include ietf-snmp-common;  
  
  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>

Editor: Juergen Schoenwaelder
<mailto:j.schoenwaelder@jacobs-university.de>;

description

"This submodule contains a collection of YANG definitions
for configuring SNMP targets.

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.

reference

"RFC3413: Simple Network Management Protocol (SNMP)
Applications";

// 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: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {

list target {
 key name;
 description
 "List of targets.";

```

reference "SNMP-TARGET-MIB.snmpTargetAddrTable";

leaf name {
  type snmp:identifier;
  description
    "Identifies the target.";
  reference "SNMP-TARGET-MIB.snmpTargetAddrName";
}
choice transport {
  mandatory true;
  description
    "Transport address of the target.

    The snmpTargetAddrTDomain and snmpTargetAddrTAddress
    objects are mapped to transport-specific YANG nodes. Each
    transport is configured as a separate case in this
    choice. Submodules providing configuration for additional
    transports are expected to augment this choice.";
  reference "SNMP-TARGET-MIB.snmpTargetAddrTDomain
    SNMP-TARGET-MIB.snmpTargetAddrTAddress";
  case udp {
    reference "SNMPv2-TM.snmpUDPDomain
      TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv4
      TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv4z
      TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv6
      TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv6z";
    container udp {
      leaf ip {
        type inet:ip-address;
        mandatory true;
        reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress";
      }
      leaf port {
        type inet:port-number;
        default 162;
        description
          "UDP port number";
        reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress";
      }
      leaf prefix-length {
        type uint8;
        description
          "The value of this leaf must match the value of
          ../snmp:ip. If ../snmp:ip contains an ipv4 address,
          this leaf must be less than or equal to 32. If it
          contains an ipv6 address, it must be less than or
          equal to 128."
      }
    }
  }
}

```

```

        Note that the prefix-length is currently only used
        by the Community-based Security Model to filter
        incoming messages. Furthermore, the prefix-length
        filtering does not cover all possible filters
        supported by the corresponding MIB object.";
        reference "SNMP-COMMUNITY-MIB.snmpTargetAddrTMask";
    }
}
}
}
leaf-list tag {
    type snmp:identifier;
    description
        "List of tag values used to select target address.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrTagList";
}
leaf timeout {
    type uint32;
    units "0.01 seconds";
    default 1500;
    description
        "Needed only if this target can receive
        InformRequest-PDUs.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrTimeout";
}
leaf retries {
    type uint8;
    default 3;
    description
        "Needed only if this target can receive
        InformRequest-PDUs.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrRetryCount";
}
choice params {
    description
        "This choice is augmented with case nodes containing
        security model specific configuration parameters. Each
        such case represents one entry in the
        snmpTargetParamsTable.

        When the snmpTargetAddrParams object contains a reference
        to a non-existing snmpTargetParamsEntry, this choice does
        not contain any case, and vice versa.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrParams
        SNMP-TARGET-MIB.snmpTargetParamsTable";
}
}
}

```

```
}
```

```
<CODE ENDS>
```

3.5. Submodule 'ietf-snmp-notification'

```
<CODE BEGINS> file "ietf-snmp-notification.yang"
```

```
submodule ietf-snmp-notification {
```

```
  belongs-to ietf-snmp {  
    prefix snmp;  
  }
```

```
  include ietf-snmp-common;  
  include ietf-snmp-target;
```

```
  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>
```

```
    Editor: Juergen Schoenwaelder  
            <mailto:j.schoenwaelder@jacobs-university.de>;
```

```
  description
```

```
    "This submodule contains a collection of YANG definitions  
    for configuring SNMP notifications.
```

```
    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.

reference
  "RFC3413: Simple Network Management Protocol (SNMP)
    Applications";

// 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: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {

  list notify {
    key name;
    description
      "Targets that will receive notifications.

      Entries in this lists are mapped 1-1 to entries in
      snmpNotifyTable, except that if an entry in snmpNotifyTable
      has a snmpNotifyTag for which no snmpTargetAddrEntry exists,
      then the snmpNotifyTable entry is not mapped to an entry in
      this list.";
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyTable";

    leaf name {
      type snmp:identifier;
      description
        "An arbitrary name for the list entry.";
      reference "SNMP-NOTIFICATION-MIB.snmpNotifyName";
    }
    leaf tag {
      type snmp:identifier;
      mandatory true;
      description
        "Target tag, selects a set of notification targets.
```

```

        Implementations MAY restrict the values of this leaf
        to be one of the available values of /snmp/target/tag in
        a valid configuration.";
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyTag";
}
leaf type {
    type enumeration {
        enum trap { value 1; }
        enum inform { value 2; }
    }
    default trap;
    description
        "Defines the notification type to be generated.";
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyType";
}
}

list notify-filter-profile {
    if-feature snmp:notification-filter;
    key name;

    description
        "Notification filter profiles.

        The leaf /snmp/target/notify-filter-profile is used
        to associate a filter profile with a target.

        If an entry in this list is referred to by one or more
        /snmp/target/notify-filter-profile, each such
        notify-filter-profile is represented by one
        snmpNotifyFilterProfileEntry.

        If an entry in this list is not referred to by any
        /snmp/target/notify-filter-profile, the entry is not mapped
        to snmpNotifyFilterProfileTable.";
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileTable
        SNMP-NOTIFICATION-MIB.snmpNotifyFilterTable";

    leaf name {
        type snmp:identifier;
        description
            "Name of the filter profile";
        reference
            "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileName";
    }

    leaf-list include {
        type snmp:wildcard-object-identifier;

```

```

        description
            "A family of subtrees included in this filter.";
        reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterSubtree
                SNMP-NOTIFICATION-MIB.snmpNotifyFilterMask
                SNMP-NOTIFICATION-MIB.snmpNotifyFilterType";
    }

    leaf-list exclude {
        type snmp:wildcard-object-identifier;
        description
            "A family of subtrees excluded from this filter.";
        reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterSubtree
                SNMP-NOTIFICATION-MIB.snmpNotifyFilterMask
                SNMP-NOTIFICATION-MIB.snmpNotifyFilterType";
    }
}

leaf enable-authen-traps {
    type boolean;
    description
        "Indicates whether the SNMP entity is permitted to
        generate authenticationFailure traps.";
    reference "SNMPv2-MIB.snmpEnableAuthenTraps";
}

augment /snmp:snmp/snmp:target {
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileTable";
    leaf notify-filter-profile {
        if-feature snmp:notification-filter;
        type leafref {
            path "/snmp/notify-filter-profile/name";
        }
        description
            "This leafref leaf is used to represent the sparse
            relationship between the /snmp/target list and the
            /snmp/notify-filter-profile list.";
        reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileName";
    }
}

}

<CODE ENDS>

```

3.6. Submodule 'ietf-snmp-proxy'

```
<CODE BEGINS> file "ietf-snmp-proxy.yang"

submodule ietf-snmp-proxy {

  belongs-to ietf-snmp {
    prefix snmp;
  }

  include ietf-snmp-common;
  include ietf-snmp-target;

  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>

    Editor: Juergen Schoenwaelder
            <mailto:j.schoenwaelder@jacobs-university.de>";

  description
    "This submodule contains a collection of YANG definitions
    for configuring SNMP proxies.

    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.

reference
  "RFC3413: Simple Network Management Protocol (SNMP)
    Applications";

// 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: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {
  if-feature snmp:proxy;

  list proxy {
    key name;

    description
      "List of proxy parameters.";
    reference "SNMP-PROXY-MIB.snmpProxyTable";

    leaf name {
      type snmp:identifier;
      description
        "Identifies the proxy parameter entry.";
      reference "SNMP-PROXY-MIB.snmpProxyName";
    }
    leaf type {
      type enumeration {
        enum read;
        enum write;
        enum trap;
        enum inform;
      }
      mandatory true;
      reference "SNMP-PROXY-MIB.snmpProxyType";
    }
    leaf context-engine-id {
      type snmp:engine-id;
      mandatory true;
      reference "SNMP-PROXY-MIB.snmpProxyContextEngineID";
    }
  }
}
```

```

    leaf context-name {
      type snmp:context-name;
      reference "SNMP-PROXY-MIB.snmpProxyContextName";
    }
    container params-in {
      choice params {
        mandatory true;
        description
          "This choice is augmented with case nodes containing
          security model specific configuration parameters. Each
          such case represents one entry in the
          snmpTargetParamsTable.

          When the snmpProxyTargetParamsIn object contains a
          reference to a non-existing snmpTargetParamsEntry, this
          choice does not contain any case, and vice versa.";
      }
      reference "SNMP-PROXY-MIB.snmpProxyTargetParamsIn";
    }
    leaf single-target-out {
      when "../type = 'read' or ../type = 'write'";
      type snmp:identifier;
      description
        "Implementations MAY restrict the values of this leaf
        to be one of the available values of /snmp/target/name in
        a valid configuration.";
      reference "SNMP-PROXY-MIB.snmpProxySingleTargetOut";
    }
    leaf multiple-target-out {
      when "../type = 'trap' or ../type = 'inform'";
      type snmp:identifier;
      description
        "Implementations MAY restrict the values of this leaf
        to be one of the available values of /snmp/target/tag in
        a valid configuration.";
      reference "SNMP-PROXY-MIB.snmpProxyMultipleTargetOut";
    }
  }
}

```

<CODE ENDS>

3.7. Submodule 'ietf-snmp-community'

<CODE BEGINS> file "ietf-snmp-community.yang"

```

submodule ietf-snmp-community {

```

```
belongs-to ietf-snmp {
  prefix snmp;
}

include ietf-snmp-common;
include ietf-snmp-target;
include ietf-snmp-proxy;

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>

  Editor: Juergen Schoenwaelder
          <mailto:j.schoenwaelder@jacobs-university.de>";

description
  "This submodule contains a collection of YANG definitions
  for configuring community-based SNMP.

  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.

reference
```

```
"RFC3584: Coexistence between Version 1, Version 2, and Version 3
  of the Internet-standard Network Management Framework";

// 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: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {

  list community {
    key index;

    description
      "List of communities";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityTable";

    leaf index {
      type snmp:identifier;
      description
        "Index into the community list.";
      reference "SNMP-COMMUNITY-MIB.snmpCommunityIndex";
    }
    choice name {
      description
        "The community name, either specified as a string
        or as a binary. The binary name is used when the
        community name contains characters that are not legal
        in a string.

        If not set, the value of 'security-name' is operationally
        used as the snmpCommunityName.";
      reference "SNMP-COMMUNITY-MIB.snmpCommunityName";
      leaf text-name {
        type string;
        description
          "A community name that can be represented as a
          YANG string.";
      }
      leaf binary-name {
        type binary;
        description
          "A community name represented as a binary value.";
      }
    }
  }
}
```



```

    }
  }
  leaf security-name {
    type snmp:security-name;
    mandatory true;
    description
      "The snmpCommunitySecurityName of this entry.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunitySecurityName";
  }
  leaf engine-id {
    if-feature snmp:proxy;
    type snmp:engine-id;
    description
      "If not set, the value of the local SNMP engine is
       operationally used by the device.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityContextEngineID";
  }
  leaf context {
    type snmp:context-name;
    default "";
    description
      "The context in which management information is accessed
       when using the community string specified by this entry.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityContextName";
  }
  leaf target-tag {
    type snmp:identifier;
    description
      "Used to limit access for this community to the specified
       targets.

       Implementations MAY restrict the values of this leaf
       to be one of the available values of /snmp/target/tag in
       a valid configuration.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityTransportTag";
  }
}
}

grouping vl-target-params {
  container vl {
    description
      "SNMPv1 parameters type.
       Represents snmpTargetParamsMPModel '0',
       snmpTargetParamsSecurityModel '1', and
       snmpTargetParamsSecurityLevel 'noAuthNoPriv'.";
    leaf security-name {
      type snmp:security-name;
    }
  }
}

```

```

        mandatory true;
        description
            "Implementations MAY restrict the values of this leaf
             to be one of the available values of
             /snmp/community/security-name in a valid configuration.";
        reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
    }
}

grouping v2c-target-params {
    container v2c {
        description
            "SNMPv2 community parameters type.
             Represents snmpTargetParamsMPModel '1',
             snmpTargetParamsSecurityModel '2', and
             snmpTargetParamsSecurityLevel 'noAuthNoPriv'.";
        leaf security-name {
            type snmp:security-name;
            mandatory true;
            description
                "Implementations MAY restrict the values of this leaf
                 to be one of the available values of
                 /snmp/community/security-name in a valid configuration.";
            reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
        }
    }
}

augment /snmp:snmp/snmp:target/snmp:params {
    case v1 {
        uses v1-target-params;
    }
    case v2c {
        uses v2c-target-params;
    }
}

augment /snmp:snmp/snmp:proxy/snmp:params-in/snmp:params {
    case v1 {
        uses v1-target-params;
    }
    case v2c {
        uses v2c-target-params;
    }
}

augment /snmp:snmp/snmp:target {

```

```
when "snmp:v1 or snmp:v2c";
leaf mms {
  type union {
    type enumeration {
      enum "unknown";
    }
    type int32 {
      range "484..max";
    }
  }
  default "484";
  reference
    "SNMP-COMMUNITY-MIB.snmpTargetAddrMMS";
}
}
}

<CODE ENDS>
```

3.8. Submodule 'ietf-snmp-vacm'

```
<CODE BEGINS> file "ietf-snmp-vacm.yang"

submodule ietf-snmp-vacm {

  belongs-to ietf-snmp {
    prefix snmp;
  }

  include ietf-snmp-common;

  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>
```

Editor: Juergen Schoenwaelder
<mailto:j.schoenwaelder@jacobs-university.de>;

description

"This submodule contains a collection of YANG definitions for configuring the View-based Access Control Model (VACM) of SNMP.

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.

reference

"RFC3415: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)";

// 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: A YANG Data Model for SNMP Configuration";  
}
```

```
typedef view-name {  
  type snmp:identifier;  
  description  
    "The view-name type represents an SNMP VACM view name."  
}
```

```
typedef group-name {  
  type snmp:identifier;  
  description  
    "The group-name type represents an SNMP VACM group name.";
```

```

}

augment /snmp:snmp {

  container vacm {
    description
      "Configuration of the View-based Access Control Model";

    list group {
      key name;
      description
        "VACM Groups.

        This data model has a different structure than the MIB.
        Groups are explicitly defined in this list, and group
        members are defined in the 'member' list (mapped to
        vacmSecurityToGroupTable), and access for the group is
        defined in the 'access' list (mapped to
        vacmAccessTable).";
      reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityToGroupTable
        SNMP-VIEW-BASED-ACM-MIB.vacmAccessTable";

      leaf name {
        type group-name;
        description
          "The name of this VACM group.";
        reference "SNMP-VIEW-BASED-ACM-MIB.vacmGroupName";
      }

      list member {
        key "security-name";
        min-elements 1;
        description
          "A member of this VACM group. According to VACM, every
          group must have at least one member.

          A certain combination of security-name and
          security-model MUST NOT be present in more than
          one group.";
        reference
          "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityToGroupTable";

        leaf security-name {
          type snmp:security-name;
          description
            "The securityName of a group member.";
          reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityName";
        }
      }
    }
  }
}

```

```

    leaf-list security-model {
        type snmp:security-model;
        min-elements 1;
        description
            "The security models under which this security-name
             is a member of this group.";
        reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityModel";
    }
}

list access {
    key "context security-model security-level";
    description
        "Definition of access right for groups";
    reference "SNMP-VIEW-BASED-ACM-MIB.vacmAccessTable";

    leaf context {
        type snmp:context-name;
        description
            "The context (prefix) under which the access rights
             apply.";
        reference
            "SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextPrefix";
    }

    leaf context-match {
        type enumeration {
            enum exact;
            enum prefix;
        }
        default exact;
        reference
            "SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextMatch";
    }

    leaf security-model {
        type snmp:security-model-or-any;
        description
            "The security model under which the access rights
             apply.";
        reference
            "SNMP-VIEW-BASED-ACM-MIB.vacmAccessSecurityModel";
    }

    leaf security-level {
        type snmp:security-level;
        description
            "The minimum security level under which the access

```

```

        rights apply.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmAccessSecurityLevel";
}

leaf read-view {
    type view-name;
    description
        "The name of the MIB view of the SNMP context
        authorizing read access. If this leaf does not
        exist in a configuration, it maps to a zero-length
        vacmAccessReadViewName.

        Implementations MAY restrict the values of this
        leaf to be one of the available values of
        /snmp/vacm/view/name in a valid configuration.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmAccessReadViewName";
}

leaf write-view {
    type view-name;
    description
        "The name of the MIB view of the SNMP context
        authorizing write access. If this leaf does not
        exist in a configuration, it maps to a zero-length
        vacmAccessWriteViewName.

        Implementations MAY restrict the values of this
        leaf to be one of the available values of
        /snmp/vacm/view/name in a valid configuration.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmAccessWriteViewName";
}

leaf notify-view {
    type view-name;
    description
        "The name of the MIB view of the SNMP context
        authorizing notify access. If this leaf does not
        exist in a configuration, it maps to a zero-length
        vacmAccessNotifyViewName.

        Implementations MAY restrict the values of this
        leaf to be one of the available values of
        /snmp/vacm/view/name in a valid configuration.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmAccessNotifyViewName";
}

```

```

    }
  }
}

list view {
  key name;
  description
    "Definition of MIB views.";
  reference
    "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyTable";

  leaf name {
    type view-name;
    description
      "The name of this VACM MIB view.";
    reference
      "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyName";
  }

  leaf-list include {
    type snmp:wildcard-object-identifier;
    description
      "A family of subtrees included in this MIB view.";
    reference
      "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree
      SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyMask
      SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyType";
  }

  leaf-list exclude {
    type snmp:wildcard-object-identifier;
    description
      "A family of subtrees excluded from this MIB view.";
    reference
      "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree
      SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyMask
      SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyType";
  }
}
}
}

<CODE ENDS>

```


3.9. Submodule 'ietf-snmp-usm'

```
<CODE BEGINS> file "ietf-snmp-usm.yang"

submodule ietf-snmp-usm {

  belongs-to ietf-snmp {
    prefix snmp;
  }

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-netconf-acm {
    prefix nacm;
  }

  include ietf-snmp-common;
  include ietf-snmp-target;
  include ietf-snmp-proxy;

  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>

    Editor:   Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>";

  description
    "This submodule contains a collection of YANG definitions for
    configuring the User-based Security Model (USM) of SNMP.

    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.

reference

"RFC3414: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3).";

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

revision 2013-02-11 {
 description
 "Initial revision."
 reference
 "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

grouping key {
 leaf key {
 type yang:hex-string;
 mandatory true;
 nacm:default-deny-all;
 description
 "Localized key specified as a list of colon-specified
 hexa-decimal octets";
 }
}

grouping user-list {
 list user {
 key "name";

 reference "SNMP-USER-BASED-SM-MIB.usmUserTable";

 leaf name {
 type snmp:identifier;
 reference "SNMP-USER-BASED-SM-MIB.usmUserName";
 }
 }
 container auth {

```

    presence "enables authentication";
    description
        "Enables authentication of the user";
    choice protocol {
        mandatory true;
        reference "SNMP-USER-BASED-SM-MIB.usmUserAuthProtocol";
        container md5 {
            uses key;
            reference
                "SNMP-USER-BASED-SM-MIB.usmHMACMD5AuthProtocol";
        }
        container sha {
            uses key;
            reference
                "SNMP-USER-BASED-SM-MIB.usmHMACSHAAuthProtocol";
        }
    }
}
}
container priv {
    must "../auth" {
        error-message
            "when privacy is used, authentication must also be used";
    }
    presence "enables encryption";
    description
        "Enables encryption of SNMP messages.";

    choice protocol {
        mandatory true;
        reference "SNMP-USER-BASED-SM-MIB.usmUserPrivProtocol";
        container des {
            uses key;
            reference "SNMP-USER-BASED-SM-MIB.usmDESPrivProtocol";
        }
        container aes {
            uses key;
            reference "SNMP-USM-AES-MIB.usmAesCfb128Protocol";
        }
    }
}
}
}

augment /snmp:snmp {

    container usm {
        description
            "Configuration of the User-based Security Model";
    }
}

```

```

    container local {
        uses user-list;
    }

    list remote {
        key "engine-id";

        leaf engine-id {
            type snmp:engine-id;
            reference "SNMP-USER-BASED-SM-MIB.usmUserEngineID";
        }

        uses user-list;
    }
}

grouping usm-target-params {
    container usm {
        description
            "User based SNMPv3 parameters type.

            Represents snmpTargetParamsMModel '3' and
            snmpTargetParamsSecurityModel '3'";

        leaf user-name {
            type snmp:security-name;
            mandatory true;
            reference
                "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
        }

        leaf security-level {
            type snmp:security-level;
            mandatory true;
            reference
                "SNMP-TARGET-MIB.snmpTargetParamsSecurityLevel";
        }
    }
}

augment /snmp:snmp/snmp:target/snmp:params {
    case usm {
        uses usm-target-params;
    }
}

augment /snmp:snmp/snmp:proxy/snmp:params-in/snmp:params {
    case usm {
        uses usm-target-params;
    }
}

```

```
    }  
  }  
}  
  
<CODE ENDS>
```

3.10. Submodule 'ietf-snmp-tsm'

```
<CODE BEGINS> file "ietf-snmp-tsm.yang"  
  
submodule ietf-snmp-tsm {  
  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
  
  include ietf-snmp-common;  
  include ietf-snmp-target;  
  include ietf-snmp-proxy;  
  
  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>  
  
    Editor:   Juergen Schoenwaelder  
              <mailto:j.schoenwaelder@jacobs-university.de>";  
  
  description  
    "This submodule contains a collection of YANG definitions for  
    configuring the Transport Security Model (TSM) of SNMP.  
  
    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.

reference

"RFC5591: Transport Security Model for the
Simple Network Management Protocol (SNMP)";

// 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: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {
 if-feature tsm;
 container tsm {
 description
 "Configuration of the Transport-based Security Model";

 leaf use-prefix {
 type boolean;
 default false;
 reference
 "SNMP-TSM-MIB.snmpTsmConfigurationUsePrefix";
 }
 }
}

grouping tsm-target-params {
 container tsm {
 description
 "Transport based security SNMPv3 parameters type.

 Represents snmpTargetParamsMPModel '3' and
 snmpTargetParamsSecurityModel '4'";

```

        leaf security-name {
            type snmp:security-name;
            mandatory true;
            reference
                "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
        }
        leaf security-level {
            type snmp:security-level;
            mandatory true;
            reference
                "SNMP-TARGET-MIB.snmpTargetParamsSecurityLevel";
        }
    }
}

augment /snmp:snmp/snmp:target/snmp:params {
    if-feature tsm;
    case tsm {
        uses tsm-target-params;
    }
}

augment /snmp:snmp/snmp:proxy/snmp:params-in/snmp:params {
    if-feature tsm;
    case tsm {
        uses tsm-target-params;
    }
}
}

<CODE ENDS>

```

3.11. Submodule 'ietf-snmp-tls'

```

<CODE BEGINS> file "ietf-snmp-tls.yang"

submodule ietf-snmp-tls {

    belongs-to ietf-snmp {
        prefix snmp;
    }

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

```

```
include ietf-snmp-common;  
include ietf-snmp-engine;  
include ietf-snmp-target;
```

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>>

Editor: Juergen Schoenwaelder
<<mailto:j.schoenwaelder@jacobs-university.de>>;

description

"This submodule contains a collection of YANG definitions for configuring the Transport Layer Security Transport Model (TLSTM) of SNMP.

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.

reference

"RFC6353: Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)";


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

```
revision 2013-02-11 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}
```

```
/* Typedefs */
```

```
typedef tls-fingerprint {
  type yang:hex-string {
    pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2}){0,254}';
  }
  description
    "A fingerprint value that can be used to uniquely reference
    other data of potentially arbitrary length.

    An tls-fingerprint value is composed of a 1-octet hashing
    algorithm identifier followed by the fingerprint value. The
    octet value encoded is taken from the IANA TLS HashAlgorithm
    Registry (RFC 5246). The remaining octets are filled using
    the results of the hashing algorithm.

    The corresponding TEXTUAL-CONVENTION allows a zero-length
    value to be used for objects that are optional. In the YANG
    data models, such objects are represented as optional leafs.";
  reference "SNMP-TLS-TM-MIB.SnmpTLSTmCertFingerprint";
}
```

```
/* Identities */
```

```
identity cert-to-tm-security-name {
}
```

```
identity specified {
  base cert-to-tm-security-name;
  reference "SNMP-TLS-TM-MIB.snmpTlstmCertSpecified";
}
```

```
identity san-rfc822-name {
  base cert-to-tm-security-name;
  reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANRFC822Name";
}
```

```
identity san-dns-name {
```

```

    base cert-to-tm-security-name;
    reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANDNSName";
}

identity san-ip-address {
    base cert-to-tm-security-name;
    reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANIpAddress";
}

identity san-any {
    base cert-to-tm-security-name;
    reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANAny";
}

identity common-name {
    base cert-to-tm-security-name;
    reference "SNMP-TLS-TM-MIB.snmpTlstmCertCommonName";
}

augment /snmp:snmp/snmp:engine/snmp:listen {
    if-feature tlstm;
    list tls {
        key "ip port";
        description
            "A list of IPv4 and IPv6 addresses and ports to which the
            engine listens for SNMP messages over TLS.";

        leaf ip {
            type inet:ip-address;
            description
                "The IPv4 or IPv6 address on which the engine listens
                for SNMP messages over TLS.";
        }
        leaf port {
            type inet:port-number;
            description
                "The TCP port on which the engine listens for SNMP
                messages over TLS.";
        }
    }
    list dtls {
        key "ip port";
        description
            "A list of IPv4 and IPv6 addresses and ports to which the
            engine listens for SNMP messages over DTLS.";

        leaf ip {
            type inet:ip-address;

```

```

        description
            "The IPv4 or IPv6 address on which the engine listens
            for SNMP messages over DTLS.";
    }
    leaf port {
        type inet:port-number;
        description
            "The UDP port on which the engine listens for SNMP messages
            over DTLS.";
    }
}

augment /snmp:snmp {
    if-feature tlstm;
    container tlstm {
        list cert-to-tm-security-name {
            key id;
            reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNEntry";

            leaf id {
                type uint32;
                reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNID";
            }
            leaf fingerprint {
                type tls-fingerprint;
                reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNFingerprint";
            }
            leaf map-type {
                type identityref {
                    base cert-to-tm-security-name;
                }
                description
                    "Mappings that use the snmpTlstmCertToTSNData column
                    need to augment the 'cert-to-tm-security-name' list
                    with additional configuration objects corresponding
                    to the snmpTlstmCertToTSNData value. Such objects
                    should use the 'when' statement to make them
                    conditional based on the 'map-type'.";
                reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNMapType";
            }
            leaf cert-specified-tm-security-name {
                when "../map-type = 'snmp:specified'";
                type snmp:admin-string;
                description
                    "Maps to snmpTlstmCertToTSNData when 'map-type' is
                    'specified'.";
                reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNData";
            }
        }
    }
}

```

```

    }
  }
}

grouping tls-transport {
  leaf ip {
    type inet:host;
    mandatory true;
    reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
              SNMP-TLS-TM-MIB.SnmpTLSAddress";
  }
  leaf port {
    type inet:port-number;
    default 10161;
    reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
              SNMP-TLS-TM-MIB.SnmpTLSAddress";
  }
  leaf client-fingerprint {
    type tls-fingerprint;
    reference "SNMP-TLS-TM-MIB.snmpTlstmParamsClientFingerprint";
  }
  leaf server-fingerprint {
    type tls-fingerprint;
    reference "SNMP-TLS-TM-MIB.snmpTlstmAddrServerFingerprint";
  }
  leaf server-identity {
    type snmp:admin-string;
    reference "SNMP-TLS-TM-MIB.snmpTlstmAddrServerIdentity";
  }
}

augment /snmp:snmp/snmp:target/snmp:transport {
  if-feature tlstm;
  case tls {
    reference "SNMP-TLS-TM-MIB.snmpTLSTCPDomain";
    container tls {
      uses tls-transport;
    }
  }
}

augment /snmp:snmp/snmp:target/snmp:transport {
  if-feature dtls;
  case dtls {
    reference "SNMP-TLS-TM-MIB.snmpDTLSUDPDDomain";
    container dtls {
      uses tls-transport;
    }
  }
}

```

```
    }  
  }  
}
```

<CODE ENDS>

3.12. Submodule 'ietf-snmp-ssh'

<CODE BEGINS> file "ietf-snmp-ssh.yang"

```
submodule ietf-snmp-ssh {  
  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
  
  import ietf-inet-types {  
    prefix inet;  
  }  
  
  include ietf-snmp-common;  
  include ietf-snmp-engine;  
  include ietf-snmp-target;  
  
  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>  
  
    Editor:   Juergen Schoenwaelder  
              <mailto:j.schoenwaelder@jacobs-university.de>";  
  
  description  
    "This submodule contains a collection of YANG definitions for  
    configuring the Secure Shell Transport Model (SSHTM)  
    of SNMP."
```

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.

reference

"RFC5592: Secure Shell Transport Model for the
Simple Network Management Protocol (SNMP)";

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

revision 2012-11-26 {
 description
 "Initial revision.";
 reference
 "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp/snmp:engine/snmp:listen {
 if-feature sshtm;
 list ssh {
 key "ip port";
 description
 "A list of IPv4 and IPv6 addresses and ports to which the
 engine listens for SNMP messages over SSH.";

 leaf ip {
 type inet:ip-address;
 description
 "The IPv4 or IPv6 address on which the engine listens
 for SNMP messages over SSH.";
 }
 leaf port {
 type inet:port-number;
 description
 "The TCP port on which the engine listens for SNMP

```

        messages over SSH.";
    }
}

augment /snmp:snmp/snmp:target/snmp:transport {
  if-feature sshtm;
  case ssh {
    reference "SNMP-SSH-TM-MIB.snmpSSHDomain";
    container ssh {
      leaf ip {
        type inet:host;
        mandatory true;
        reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
                  SNMP-SSH-TM-MIB.SnmpSSHAddress";
      }
      leaf port {
        type inet:port-number;
        default 5161;
        reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
                  SNMP-SSH-TM-MIB.SnmpSSHAddress";
      }
      leaf username {
        type string;
        reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
                  SNMP-SSH-TM-MIB.SnmpSSHAddress";
      }
    }
  }
}
}
}
}

```

<CODE ENDS>

4. 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-snmp

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

The document registers the following YANG submodules in the YANG Module Names registry [RFC6020].

name:	ietf-snmp-common
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-engine
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-community
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-notification
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-target
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-vacm
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-usm
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-tsm
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-tls
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-ssh
parent:	ietf-snmp
reference:	RFC XXXX

5. Security Considerations

The YANG module and submodules defined in this memo 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].

There are a number of data nodes defined in the YANG module and submodules 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:

<list subtrees and data nodes and state why they are sensitive>

Some of the readable data nodes in the YANG module and submodules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

6. Acknowledgments

The authors want to thank Wes Hardaker and David Spakes for their reviews and valuable comments.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [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.

7.2. Informative References

- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.
- [RFC3412] Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, December 2002.
- [RFC3413] Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, December 2002.
- [RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.
- [RFC3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, December 2002.
- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.
- [RFC3584] Frye, R., Levi, D., Routhier, S., and B. Wijnen,

"Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework", BCP 74, RFC 3584, August 2003.

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", RFC 5591, June 2009.
- [RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP)", RFC 5592, June 2009.
- [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.

Appendix A. Example configurations

A.1. Engine Configuration Example

Below is an XML instance document showing a configuration of an SNMP engine listening on UDP port 161 on IPv4 and IPv6 endpoints and accepting SNMPv2c and SNMPv3 messages.

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <engine>
    <enabled>true</enabled>
    <listen>
      <udp>
        <ip>0.0.0.0</ip>
        <port>161</port>
      </udp>
      <udp>
        <ip>:::</ip>
        <port>161</port>
      </udp>
    </listen>
    <version>
      <v2c/>
      <v3/>
    </version>
    <engine-id>80:00:02:b8:04:61:62:63</engine-id>
  </engine>
</snmp>
```

A.2. Community Configuration Example

Below is an XML instance document showing a configuration that maps the community name "public" to the security-name "community-public" on the local engine with the default context name. The target tag "community-public-access" filters the access to this community name.

```

<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <community>
    <index>1</index>
    <text-name>public</text-name>
    <security-name>community-public</security-name>
    <target-tag>community-public-access</target-tag>
  </community>
  <target>
    <name>bluebox</name>
    <udp>
      <ip>2001:db8::abcd</ip>
      <port>161</port>
    </udp>
    <tag>blue</tag>
    <v2c>
      <security-name>community-public</security-name>
    </v2c>
  </target>
</snmp>

```

A.3. User-based Security Model Configuration Example

Below is an XML instance document showing the configuration of a local user "joey" who has no authentication or privacy keys. For the remote SNMP engine identified by the snmpEngineID '800002b804616263'H, two users are configured. The user "matt" has a localized SHA authentication key and the user "russ" has a localized SHA authentication key and an AES encryption key.

```

<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <usm>
    <local>
      <user>
        <name>joey</name>
      </user>
    </local>
    <remote>
      <engine-id>00:00:00:00:00:00:00:00:00:00:02</engine-id>
      <user>
        <name>matt</name>
        <auth>
          <sha>
            <!--
              The 'key' value is split into two lines to match
              the RFC formatting rules.
            -->
            <key>66:95:fe:bc:92:88:e3:62:82:23:
              5f:c7:15:1f:12:84:97:b3:8f:3f</key>
          </sha>
        </auth>
      </user>
    </remote>
  </usm>
</snmp>

```

```

        </sha>
      </auth>
    </user>
    <user>
      <name>russ</name>
      <auth>
        <sha>
          <!--
            The 'key' value is split into two lines to match
            the RFC formatting rules.
          -->
          <key>66:95:fe:bc:92:88:e3:62:82:23:
            5f:c7:15:1f:12:84:97:b3:8f:3f</key>
        </sha>
      </auth>
      <priv>
        <aes>
          <!--
            The 'key' value is split into two lines to match
            the RFC formatting rules.
          -->
          <key>66:95:fe:bc:92:88:e3:62:82:23:
            5f:c7:15:1f:12:84</key>
        </aes>
      </priv>
    </user>
  </remote>
</usm>
<target>
  <name>bluebox</name>
  <udp>
    <ip>2001:db8::abcd</ip>
    <port>161</port>
  </udp>
  <tag>blue</tag>
  <usm>
    <user-name>matt</user-name>
    <security-level>auth-no-priv</security-level>
  </usm>
</target>
</snmp>

```

A.4. Target and Notification Configuration Example

Below is an XML instance document showing the configuration of a notification generator application (see Appendix A of [RFC3413]). Note that the USM specific objects are defined in the `ietf-snmp-usm.yang` submodule.


```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <target>
    <name>addr1</name>
    <udp>
      <ip>192.0.2.3</ip>
      <port>162</port>
    </udp>
    <tag>group1</tag>
    <usm>
      <user-name>joe</user-name>
      <security-level>auth-no-priv</security-level>
    </usm>
  </target>
  <target>
    <name>addr2</name>
    <udp>
      <ip>192.0.2.6</ip>
      <port>162</port>
    </udp>
    <tag>group1</tag>
    <usm>
      <user-name>joe</user-name>
      <security-level>auth-no-priv</security-level>
    </usm>
  </target>
  <target>
    <name>addr3</name>
    <udp>
      <ip>192.0.2.9</ip>
      <port>162</port>
    </udp>
    <tag>group2</tag>
    <usm>
      <user-name>bob</user-name>
      <security-level>auth-priv</security-level>
    </usm>
  </target>
  <notify>
    <name>group1</name>
    <tag>group1</tag>
    <type>trap</type>
  </notify>
  <notify>
    <name>group2</name>
    <tag>group2</tag>
    <type>trap</type>
  </notify>
</snmp>
```

A.5. Proxy Configuration Example

Below is an XML instance document showing the configuration of a proxy forwarder application. It proxies SNMPv2c messages from command generators to a file server running a SNMPv1 agent that recognizes two community strings, "private" and "public", with different associated read views. The fileserver is represented as two "target" instances, one for each community string.

If the proxy receives a SNMPv2c message with the community string "public" from a device in the "Office Network" or "Home Office Network", it gets tagged as "trusted", and the proxy uses the "private" community string when sending the message to the file server. Other SNMPv2c messages with the community string "public" get tagged as "non-trusted", and the proxy uses the "public" community string for these messages. There is also a special "backdoor" community string that can be used from any location to get "trusted" access.

The "Office Network" and "Home Office Network" are represented as two "target" instances.

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <target>
    <name>File Server (private)</name>
    <udp>
      <ip>192.0.2.1</ip>
    </udp>
    <v1>
      <security-name>private</security-name>
    </v1>
  </target>
  <target>
    <name>File Server (public)</name>
    <udp>
      <ip>192.0.2.1</ip>
    </udp>
    <v1>
      <security-name>public</security-name>
    </v1>
  </target>
  <target>
    <name>Office Network</name>
    <udp>
      <ip>192.0.2.0</ip>
      <prefix-length>24</prefix-length>
    </udp>
    <tag>office</tag>
  </target>
</snmp>
```

```

</target>
<target>
  <name>Home Office Network</name>
  <udp>
    <ip>203.0.113.0</ip>
    <prefix-length>24</prefix-length>
  </udp>
  <tag>home-office</tag>
</target>

<!--
  Communities c1,c2,c3, and c4 are used for incoming messages
  that should be forwarded.

  Communities c3 and c5 are used for outgoing messages to the
  file server.
-->
<community>
  <index>c1</index>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>trusted</context>
  <target-tag>office</target-tag>
</community>
<community>
  <index>c2</index>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>trusted</context>
  <target-tag>home-office</target-tag>
</community>
<community>
  <index>c3</index>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>not-trusted</context>
</community>
<community>
  <index>c4</index>
  <text-name>backdoor</text-name>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>trusted</context>
</community>
<community>
  <index>c5</index>
  <security-name>private</security-name>
  <engine-id>80:00:61:81:c8</engine-id>

```

```

    <context>trusted</context>
  </community>

  <proxy>
    <name>p1</name>
    <type>read</type>
    <context-engine-id>80:00:61:81:c8</context-engine-id>
    <context-name>trusted</context-name>
    <params-in>
      <v2c>
        <security-name>public</security-name>
      </v2c>
    </params-in>
    <single-target-out>File Server (private)</single-target-out>
  </proxy>
  <proxy>
    <name>p2</name>
    <type>read</type>
    <context-engine-id>80:00:61:81:c8</context-engine-id>
    <context-name>not-trusted</context-name>
    <params-in>
      <v2c>
        <security-name>public</security-name>
      </v2c>
    </params-in>
    <single-target-out>File Server (public)</single-target-out>
  </proxy>
</snmp>

```

If an SNMPv2c Get request with community string "public" is received from an IP address tagged as "office" or "home-office", or if the request is received from anywhere else with community string "backdoor", the implied context is "trusted" and so proxy entry "p1" matches. The request is forwarded to the file server as SNMPv1 with community "private" using community table entry "c5" for outbound params lookup.

If an SNMPv2c Get request with community string "public" is received from any other IP address, the implied context is "not-trusted" so proxy entry "p2" matches, and the request is forwarded to the file server as SNMPv1 with community "public".

A.6. View-based Access Control Model Configuration Example

Below is an XML instance document showing the minimum-secure VACM configuration (see Appendix A of [RFC3415]).

```

<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <vacm>
    <group>
      <name>initial</name>
      <member>
        <security-name>initial</security-name>
        <security-model>usm</security-model>
      </member>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>no-auth-no-priv</security-level>
        <read-view>restricted</read-view>
        <notify-view>restricted</notify-view>
      </access>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>auth-no-priv</security-level>
        <read-view>internet</read-view>
        <write-view>internet</write-view>
        <notify-view>internet</notify-view>
      </access>
    </group>
    <view>
      <name>initial</name>
      <include>1.3.6.1</include>
    </view>
    <view>
      <name>restricted</name>
      <include>1.3.6.1</include>
    </view>
  </vacm>
</snmp>

```

The following XML instance document shows the semi-secure VACM configuration (only the view configuration is different).

```

<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <vacm>
    <group>
      <name>initial</name>
      <member>
        <security-name>initial</security-name>
        <security-model>usm</security-model>
      </member>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>no-auth-no-priv</security-level>
        <read-view>restricted</read-view>
        <notify-view>restricted</notify-view>
      </access>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>auth-no-priv</security-level>
        <read-view>internet</read-view>
        <write-view>internet</write-view>
        <notify-view>internet</notify-view>
      </access>
    </group>
    <view>
      <name>initial</name>
      <include>1.3.6.1</include>
    </view>
    <view>
      <name>restricted</name>
      <include>1.3.6.1.2.1.1</include>
      <include>1.3.6.1.2.1.11</include>
      <include>1.3.6.1.6.3.10.2.1</include>
      <include>1.3.6.1.6.3.11.2.1</include>
      <include>1.3.6.1.6.3.15.1.1</include>
    </view>
  </vacm>
</snmp>

```

A.7. Transport Layer Security Transport Model Configuration Example

Below is an XML instance document showing the configuration of the certificate to security name mapping (see Appendix A.2 and A.3 of [RFC6353]).

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <tlstm>
    <cert-to-tm-security-name>
      <id>1</id>
      <fingerprint>11:0A:05:11:00</fingerprint>
      <map-type>san-any</map-type>
    </cert-to-tm-security-name>
    <cert-to-tm-security-name>
      <id>2</id>
      <fingerprint>11:0A:05:11:00</fingerprint>
      <map-type>specified</map-type>
      <cert-specified-tm-security-name>
        Joe Cool
      </cert-specified-tm-security-name>
    </cert-to-tm-security-name>
  </tlstm>
</snmp>
```

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
February 25, 2013

YANG Data Model for System Management
draft-ietf-netmod-system-mgmt-05

Abstract

This document defines a YANG data model for the configuration and identification of the management system of a device.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 29, 2013.

Copyright Notice

Copyright (c) 2013 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
1.1.1. Terms	3
1.2. Tree Diagrams	3
2. Objectives	5
2.1. System Identification	5
2.2. System Time Management	5
2.3. User Authentication	5
3. System Data Model	6
3.1. System Identification	6
3.2. System Time Management	6
3.3. DNS Resolver Model	6
3.4. RADIUS Client Model	7
3.5. User Authentication Model	7
3.5.1. SSH Public Key Authentication	8
3.5.2. Local User Password Authentication	8
3.5.3. RADIUS Password Authentication	8
3.6. System Control	9
4. System YANG module	10
5. IANA Considerations	26
6. Security Considerations	27
7. Change Log	29
7.1. 00-01	29
7.2. 01-02	29
7.3. 02-03	29
7.4. 03-04	29
7.5. 04-05	29
8. Normative References	30
Authors' Addresses	32

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration and identification of the management system of a device.

Devices that are managed by NETCONF and perhaps other mechanisms have common properties that need to be configured and monitored in a standard way.

The "ietf-system" YANG module defined in this document provides the following features:

- o system administrative data configuration
- o system identification monitoring
- o system time-of-day configuration and monitoring
- o user authentication configuration
- o local users configuration

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].

1.1.1. Terms

The following terms are used within this document:

- o system: This term refers to the embodiment of the entire set of management interfaces that a single NETCONF server is supporting at a given moment. The set of physical entities managed by a single NETCONF server can be static or it can change dynamically.

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.

- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Objectives

2.1. System Identification

There are many common properties used to identify devices, operating systems, software versions, etc. that need to be supported in the system data module. These objects are defined as operational data and intended to be specific to the device vendor.

Some user-configurable administrative strings are also provided such as the system location and description.

2.2. System Time Management

The management of the date and time used by the system need to be supported. Use of one or more NTP servers to automatically set the system date and time need to be possible. Utilization of the Timezone database [RFC6557] also need to be supported.

2.3. User Authentication

The authentication mechanism need to support password authentication over RADIUS, to support deployment scenarios with centralized authentication servers. Additionally, local users need to be supported, for scenarios when no centralized authentication server exists, or for situations where the centralized authentication server cannot be reached from the device.

Since the mandatory transport protocol for NETCONF is SSH [RFC6242] the authentication model need to support SSH's "publickey" and "password" authentication methods [RFC4252].

The model for authentication configuration should be flexible enough to support authentication methods defined by other standard documents or by vendors.

3. System Data Model

3.1. System Identification

The data model for system identification has the following structure:

```
+--rw system
  +--rw contact?          string
  +--rw name?             string
  +--rw location?         string
  +--ro platform
    +--ro os-name?        string
    +--ro os-release?     string
    +--ro os-version?     string
    +--ro machine?        string
    +--ro nodename?       string
```

3.2. System Time Management

The data model for system time management has the following structure:

```
+--rw system
  +--rw clock
    | +--ro current-datetime?    yang:date-and-time
    | +--ro boot-datetime?      yang:date-and-time
    | +--rw (timezone)?
    |   +--:(timezone-location)
    |   | +--rw timezone-location?  string
    |   +--:(timezone-utc-offset)
    |   | +--rw timezone-utc-offset? int16
    +--rw ntp
      +--rw use-ntp?            boolean
      +--rw ntp-server [address]
        +--rw association-type? enumeration
        +--rw address          inet:host
        +--rw enabled?         boolean
        +--rw iburst?          boolean
        +--rw prefer?          boolean
```

3.3. DNS Resolver Model

The data model for configuration of the DNS resolver has the following structure:

```
+--rw system
  +--rw dns
    +--rw search*      inet:host
    +--rw server*      inet:ip-address
    +--rw options
      +--rw timeout?   uint8
      +--rw attempts? uint8
```

3.4. RADIUS Client Model

The data model for configuration of the RADIUS client has the following structure:

```
+--rw system
  +--rw radius
    +--rw server [address]
      | +--rw address          inet:host
      | +--rw authentication-port? inet:port-number
      | +--rw shared-secret?    string
    +--rw options
      +--rw timeout?   uint8
      +--rw attempts? uint8
```

3.5. User Authentication Model

This document defines three authentication methods for use with NETCONF:

- o publickey for local users over SSH
- o password for local users over any transport
- o password for RADIUS users over any transport

Additional methods can be defined by other standard documents or by vendors.

This document defines two optional YANG features, "local-users" and "radius-authentication", which the server advertises to indicate support for configuring local users on the device, and support for using RADIUS for authentication, respectively.

The authentication parameters defined in this document are primarily used to configure authentication of NETCONF users, but MAY also be used by other interfaces, e.g., a Command Line Interface or a Web-based User Interface.

The data model for user authentication has the following structure:


```
+--rw system
  +--rw authentication
    +--rw user-authentication-order*  identityref
    +--rw user [name]
      +--rw name          string
      +--rw password?     crypt-hash
      +--rw ssh-key [name]
        +--rw name        string
        +--rw algorithm?  string
        +--rw key-data?   binary
```

3.5.1. SSH Public Key Authentication

If the NETCONF server advertises the "local-users" feature, configuration of local users and their SSH public keys is supported in the /system/authentication/user list.

Public key authentication is requested by the SSH client. If the "local-users" feature is supported, then when a NETCONF client starts an SSH session towards the server using the "publickey" authentication "method name" [RFC4252], the SSH server looks up the user name given in the SSH authentication request in the /system/authentication/user list, and verifies the key as described in [RFC4253].

3.5.2. Local User Password Authentication

If the NETCONF server advertises the "local-users" feature, configuration of local users and their passwords is supported in the /system/authentication/user list.

For NETCONF transport protocols that support password authentication, the leaf-list "user-authentication-order" is used to control if local user password authentication should be used.

In SSH, password authentication is requested by the client. Other NETCONF transport protocols MAY also support password authentication.

When local user password authentication is requested, the NETCONF transport looks up the user name provided by the client in the /system/ authentication/user list, and verifies the password.

3.5.3. RADIUS Password Authentication

If the NETCONF server advertises the "radius-authentication" feature, the device supports user authentication using RADIUS.

For NETCONF transport protocols that support password authentication,

the leaf-list "user-authentication-order" is used to control if RADIUS password authentication should be used.

In SSH, password authentication is requested by the client. Other NETCONF transport protocols MAY also support password authentication.

3.6. System Control

Two protocol operations are included to restart or shutdown the system. The 'system-restart' operation can be used to restart the entire system (not just the NETCONF server). The 'system-shutdown' operation can be used to power off the entire system.

4. System YANG module

This YANG module imports YANG extensions from [RFC6536], and imports YANG types from [RFC6021] and [I-D.lange-netmod-iana-timezones]. It also references [RFC1321], [RFC2865], [RFC3418], [RFC5607], [IEEE-1003.1-2008], and [FIPS.180-3.2008].

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

<CODE BEGINS> file "ietf-system@2013-02-25.yang"

```
module ietf-system {
  namespace "urn:ietf:params:xml:ns:yang:ietf-system";
  prefix "sys";

  import ietf-yang-types {
    prefix yang;
  }

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-netconf-acm {
    prefix nacm;
  }

  import iana-timezones {
    prefix ianatz;
  }

  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:    Andy Bierman
               <mailto:andy@yumaworks.com>
```

Editor: Martin Bjorklund
<mailto:mbj@tail-f.com>;

description

"This module contains a collection of YANG definitions for the configuration and identification of the management system of a device.

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.: remove this note
// Note: extracted from draft-ietf-netmod-system-mgmt-05.txt

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
revision "2013-02-25" {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for System Management";
}

/*
 * Typedefs
 */

typedef crypt-hash {
  type string {
    pattern "$0$.*|$(1|5|6)$[a-zA-Z0-9./]{2,16}$.*";
  }
  description
    "The crypt-hash type is used to store passwords using
    a hash function. This type is implemented in various UNIX
    systems as the function crypt(3).
```

When a clear text value is set to a leaf of this type, the server calculates a password hash, and stores the result in the datastore. Thus, the password is never stored in clear text.

When a leaf of this type is read, the stored password hash is returned.

A value of this type matches one of the forms:

```
$0$<clear text password>
$id>$<salt>$<password hash>
```

The '\$0\$' prefix signals that the value is clear text. When such a value is received by the server, a hash value is calculated, and the string '\$<id>\$<salt>\$' is prepended to the result, where <salt> is a random 2-16 characters long salt used to generate the digest. This value is stored in the configuration data store.

If a value starting with '\$<id>\$<salt>\$' is received, the server knows that the value already represents a hashed value, and stores it as is in the data store.

When a server needs to verify a password given by a user, it finds the stored password hash string for that user, extracts the salt, and calculates the hash with the salt and given password as input. If the calculated hash value is the same as the stored value, the password given by the client is correct.

This type defines the following hash functions:

id	hash function	feature
1	MD5	crypt-hash-md5
5	SHA-256	crypt-hash-sha-256
6	SHA-512	crypt-hash-sha-512

The server indicates support for the different hash functions by advertising the corresponding feature."

reference

```
"IEEE Std 1003.1-2008 - crypt() function
Wikipedia: http://en.wikipedia.org/wiki/Crypt_(Unix)
RFC 1321: The MD5 Message-Digest Algorithm
FIPS.180-3.2008: Secure Hash Standard";
```

```
}
```

```
/*
 * Features
 */

feature radius {
  description
    "Indicates that the device can be configured as a RADIUS
    client.";
  reference
    "RFC 2865: Remote Authentication Dial In User Service "
    + "(RADIUS)";
}

feature authentication {
  description
    "Indicates that the device can be configured
    to do authentication of users.";
}

feature local-users {
  if-feature authentication;
  description
    "Indicates that the device supports
    local user authentication.";
}

feature radius-authentication {
  if-feature radius;
  if-feature authentication;
  description
    "Indicates that the device supports user authentication over
    RADIUS.";
  reference
    "RFC 2865: Remote Authentication Dial In User Service (RADIUS)
    RFC 5607: Remote Authentication Dial-In User Service (RADIUS)
    Authorization for Network Access Server (NAS)
    Management";
}

feature crypt-hash-md5 {
  description
    "Indicates that the device supports the MD5
    hash function in 'crypt-hash' values";
  reference "RFC 1321: The MD5 Message-Digest Algorithm";
}

feature crypt-hash-sha-256 {
  description
```

```
        "Indicates that the device supports the SHA-256
        hash function in 'crypt-hash' values";
        reference "FIPS.180-3.2008: Secure Hash Standard";
    }

    feature crypt-hash-sha-512 {
        description
            "Indicates that the device supports the SHA-512
            hash function in 'crypt-hash' values";
        reference "FIPS.180-3.2008: Secure Hash Standard";
    }

    feature ntp {
        description
            "Indicates that the device can be configured
            to use one or more NTP servers to set the
            system date and time.";
    }

    feature timezone-location {
        description
            "Indicates that the local timezone on the device
            can be configured to use the TZ database
            to set the timezone and manage daylight savings time.";
        reference
            "TZ Database http://www.twinsun.com/tz/tz-link.htm
            Maintaining the Timezone Database
            RFC 6557 (BCP 175)";
    }

    /*
    * Identities
    */

    identity authentication-method {
        description
            "Base identity for user authentication methods.";
    }

    identity radius {
        base authentication-method;
        description
            "Indicates user authentication using RADIUS.";
        reference
            "RFC 2865: Remote Authentication Dial In User Service (RADIUS)
            RFC 5607: Remote Authentication Dial-In User Service (RADIUS)
            Authorization for Network Access Server (NAS)
            Management";
    }
```

```
    }

    identity local-users {
      base authentication-method;
      description
        "Indicates password-based authentication of locally
        configured users.";
    }

    identity radius-authentication-type {
      description
        "Base identity for RADIUS authentication types.";
    }

    identity radius-pap {
      base radius-authentication-type;
      description
        "The device requests PAP authentication from the RADIUS
        server.";
      reference
        "RFC 2865: Remote Authentication Dial In User Service";
    }

    identity radius-chap {
      base radius-authentication-type;
      description
        "The device requests CHAP authentication from the RADIUS
        server.";
      reference
        "RFC 2865: Remote Authentication Dial In User Service";
    }

    /*
     * Top-level container
     */

    container system {
      description
        "System group configuration.";

      leaf contact {
        type string {
          length "0..255";
        }
        description
          "The administrator contact information for the system.";
        reference
          "RFC 3418 - Management Information Base (MIB) for the
```



```
        Simple Network Management Protocol (SNMP)
        SNMPv2-MIB.sysContact";
    }

    leaf name {
        type string {
            length "0..255";
        }
        description
            "The administratively assigned system name.";
        reference
            "RFC 3418 - Management Information Base (MIB) for the
             Simple Network Management Protocol (SNMP)
             SNMPv2-MIB.sysName";
    }

    leaf location {
        type string {
            length "0..255";
        }
        description
            "The system location";
        reference
            "RFC 3418 - Management Information Base (MIB) for the
             Simple Network Management Protocol (SNMP)
             SNMPv2-MIB.sysLocation";
    }

    container platform {
        config false;
        description
            "Contains vendor-specific information for
             identifying the system platform and operating system.";
        reference
            "IEEE Std 1003.1-2008 - sys/utsname.h";

        leaf os-name {
            type string;
            description
                "The name of the operating system in use,
                 for example 'Linux'";
            reference
                "IEEE Std 1003.1-2008 - utsname.sysname";
        }

        leaf os-release {
            type string;
            description
```

```
        "The current release level of the operating
        system in use. This string MAY indicate
        the OS source code revision.";
    reference
        "IEEE Std 1003.1-2008 - utsname.release";
}

leaf os-version {
    type string;
    description
        "The current version level of the operating
        system in use. This string MAY indicate
        the specific OS build date and target variant
        information.";
    reference
        "IEEE Std 1003.1-2008 - utsname.version";
}

leaf machine {
    type string;
    description
        "A vendor-specific identifier string representing
        the hardware in use.";
    reference
        "IEEE Std 1003.1-2008 - utsname.machine";
}

leaf nodename {
    type string;
    description
        "The host name of this system.";
    reference
        "IEEE Std 1003.1-2008 - utsname.nodename";
}
}

container clock {
    description
        "Configuration and monitoring of the system
        date and time properties.";

    leaf current-datetime {
        type yang:date-and-time;
        config false;
        description
            "The current system date and time.";
    }
}
```

```
leaf boot-datetime {
  type yang:date-and-time;
  config false;
  description
    "The system date and time when the NETCONF
    server last restarted.";
}

choice timezone {
  description
    "The system timezone information.";

  leaf timezone-location {
    if-feature timezone-location;
    type ianatz:iana-timezone;
    description
      "The TZ database location identifier string
      to use for the system, such as 'Europe/Stockholm'.";
  }

  leaf timezone-utc-offset {
    type intl6 {
      range "-1500 .. 1500";
    }
    units "minutes";
    description
      "The number of minutes to add to UTC time to
      identify the timezone for this system.
      For example, 'UTC - 8:00 hours' would be
      represented as '-480'. Note that automatic
      daylight savings time adjustment is not provided,
      if this object is used.";
  }
}

container ntp {
  if-feature ntp;

  description
    "Configuration of the NTP client.";

  leaf use-ntp {
    type boolean;
    default true;
    description
      "Indicates that the system should attempt
      to synchronize the system clock with an
```

```
        NTP server from the 'ntp-server' list.";
    }

    list ntp-server {
        key address;
        description
            "List of NTP servers to use for
            system clock synchronization.  If 'use-ntp'
            is 'true', then the system will attempt to
            contact and utilize the specified NTP servers.";

        leaf association-type {
            type enumeration {
                enum server {
                    description
                        "Use server association mode.  This device
                        is not expected to synchronize with the
                        configured NTP server.";
                }
                enum peer {
                    description
                        "Use peer association mode.  This device
                        may be expected to synchronize with the
                        configured NTP server.";
                }
                enum pool {
                    description
                        "Use pool association mode.  This device
                        is not expected to synchronize with the
                        configured NTP server.";
                }
            }
        }
        default server;
        description
            "The desired association type for this NTP server.";
    }
    leaf address {
        type inet:host;
        description
            "The IP address or domain name of the NTP server.";
    }
    leaf enabled {
        type boolean;
        default true;
        description
            "Indicates whether this server is enabled for use or
            not.";
    }
}
```

```
    leaf iburst {
      type boolean;
      default false;
      description
        "Indicates whether this server should enable burst
        synchronization or not.";
    }
    leaf prefer {
      type boolean;
      default false;
      description
        "Indicates whether this server should be preferred
        or not.";
    }
  }
}

container dns {
  description
    "Configuration of the DNS resolver.";

  leaf-list search {
    type inet:host;
    ordered-by user;
    description
      "An ordered list of domains to search when resolving
      a host name.";
  }
  leaf-list server {
    type inet:ip-address;
    ordered-by user;
    description
      "Addresses of the name servers that the resolver should
      query.

      Implementations MAY limit the number of entries in this
      leaf list.";
  }
  container options {
    description
      "Resolver options. The set of available options has been
      limited to those that are generally available across
      different resolver implementations, and generally
      useful.";
    leaf timeout {
      type uint8 {
        range "1..max";
      }
    }
  }
}
```

```
        units "seconds";
        default "5";
        description
            "The amount of time the resolver will wait for a
             response from a remote name server before
             retrying the query via a different name server.";
    }
    leaf attempts {
        type uint8 {
            range "1..max";
        }
        default "2";
        description
            "The number of times the resolver will send a query to
             its name servers before giving up and returning an
             error to the calling application.";
    }
}

container radius {
    if-feature radius;

    description
        "Configuration of the RADIUS client.";

    list server {
        key address;
        ordered-by user;
        description
            "List of RADIUS servers used by the device.";

        leaf address {
            type inet:host;
            description
                "The address of the RADIUS server.";
        }
        leaf authentication-port {
            type inet:port-number;
            default "1812";
            description
                "The port number of the RADIUS server.";
        }
        leaf shared-secret {
            type string;
            nacm:default-deny-all;
            description
                "The shared secret which is known to both the RADIUS
```

```
        client and server.";
    reference
        "RFC 2865: Remote Authentication Dial In User Service";
}
leaf authentication-type {
    type identityref {
        base radius-authentication-type;
    }
    default radius-pap;
    description
        "The authentication type requested from the RADIUS
        server.";
}
}
container options {
    description
        "RADIUS client options.";

    leaf timeout {
        type uint8 {
            range "1..max";
        }
        units "seconds";
        default "5";
        description
            "The number of seconds the device will wait for a
            response from a RADIUS server before trying with a
            different server.";
    }
    leaf attempts {
        type uint8 {
            range "1..max";
        }
        default "2";
        description
            "The number of times the device will send a query to
            the RADIUS servers before giving up.";
    }
}
}

container authentication {
    nacm:default-deny-write;
    if-feature authentication;

    description
        "The authentication configuration subtree.";
```

```
leaf-list user-authentication-order {
  type identityref {
    base authentication-method;
  }
  must '(. = "sys:radius" and ../../radius/server) or'
    + '(. != "sys:radius")' {
    error-message
      "When 'radius' is used, a radius server"
      + " must be configured.";
  }
  ordered-by user;

  description
    "When the device authenticates a user with
    a password, it tries the authentication methods in this
    leaf-list in order.  If authentication with one method
    fails, the next method is used.  If no method succeeds,
    the user is denied access.

    If the 'radius-authentication' feature is advertised by
    the NETCONF server, the 'radius' identity can be added to
    this list.

    If the 'local-users' feature is advertised by the
    NETCONF server, the 'local-users' identity can be
    added to this list.";
}

list user {
  if-feature local-users;
  key name;
  description
    "The list of local users configured on this device.";

  leaf name {
    type string;
    description
      "The user name string identifying this entry.";
  }
  leaf password {
    type crypt-hash;
    description
      "The password for this entry.";
  }
  list ssh-key {
    key name;
    description
      "A list of public SSH keys for this user.";
  }
}
```



```

        reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer
              Protocol";

        leaf name {
            type string;
            description
                "An arbitrary name for the ssh key.";
        }
        leaf algorithm {
            type string;
            description
                "The public key algorithm name for this ssh key.

                Valid values are the values in the IANA Secure Shell
                (SSH) Protocol Parameters registry, Public Key
                Algorithm Names";
            reference
                "IANA Secure Shell (SSH) Protocol Parameters registry,
                Public Key Algorithm Names";
        }
        leaf key-data {
            type binary;
            description
                "The binary key data for this ssh key.";
        }
    }
}

rpc set-current-datetime {
    nacm:default-deny-all;
    description
        "Set the /system/clock/current-datetime leaf
        to the specified value.

        If the system is using NTP (e.g., /system/ntp/use-ntp
        is set to 'true'), then this operation will
        fail with error-tag 'operation-failed',
        and error-app-tag value of 'ntp-active'";
    input {
        leaf current-datetime {
            type yang:date-and-time;
            mandatory true;
            description
                "The current system date and time.";
        }
    }
}

```

```
    }  
  }  
  
  rpc system-restart {  
    nacm:default-deny-all;  
    description  
      "Request that the entire system be restarted immediately.  
      A server SHOULD send an rpc reply to the client before  
      restarting the system.";  
  }  
  
  rpc system-shutdown {  
    nacm:default-deny-all;  
    description  
      "Request that the entire system be shut down immediately.  
      A server SHOULD send an rpc reply to the client before  
      shutting down the system.";  
  }  
}  
  
<CODE ENDS>
```

5. IANA Considerations

This document registers one 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-system
Registrant Contact: The NETMOD WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

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

name: ietf-system
namespace: urn:ietf:params:xml:ns:yang:ietf-system
prefix: sys
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 this 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:

- o /system/clock/timezone: This choice contains the objects used to control the timezone used by the device.
- o /system/ntp: This container contains the objects used to control the Network Time Protocol servers used by the device.
- o /system/dns: This container contains the objects used to control the Domain Name System servers used by the device.
- o /system/radius: This container contains the objects used to control the Remote Authentication Dial-In User Service servers used by the device.
- o /system/authentication/user-authentication-order: This leaf controls how user login attempts are authenticated by the device.
- o /system/authentication/user: This list contains the local users enabled on the system.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- o /system/platform: This container has objects which may help identify the specific NETCONF server and/or operating system implementation used on the device.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

- o set-current-datetime: Changes the current date and time on the device.
- o system-restart: Reboots the device.
- o system-shutdown: Shuts down the device.

7. Change Log

-- RFC Ed.: remove this section before publication.

7.1. 00-01

- o added configuration-source identities
- o added configuration-source leaf to ntp and dns (via grouping) to choose configuration source
- o added association-type, iburst, prefer, and true leafs to the ntp-server list
- o extended the ssh keys for a user to a list of keys. support all defined key algorithms, not just dsa and rsa
- o clarified timezone-utc-offset description-stmt
- o removed '/system/ntp/server/true' leaf from data model

7.2. 01-02

- o added default-stmts to ntp-server/iburst and ntp-server/prefer leafs
- o changed timezone-location leaf to use iana-timezone typedef instead of a string

7.3. 02-03

- o removed configuration-source identities and leafs

7.4. 03-04

- o removed ndots dns resolver option
- o added radius-authentication-type identity, and identities for pap and chap, and a leaf to control which authentication type to use when communicating with the radius server
- o made 0 an invalid value for timeouts and attempts

7.5. 04-05

- o updated tree diagram explanation text

8. Normative References

- [FIPS.180-3.2008]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-3, October 2008, <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>.
- [I-D.lange-netmod-iana-timezones]
Lange, J., "IANA Timezone Database YANG Module", draft-lange-netmod-iana-timezones-01 (work in progress), June 2012.
- [IEEE-1003.1-2008]
Institute of Electrical and Electronics Engineers, "POSIX.1-2008", IEEE Standard 1003.1, March 2008.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC4252] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.
- [RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.
- [RFC5607] Nelson, D. and G. Weber, "Remote Authentication Dial-In User Service (RADIUS) Authorization for Network Access Server (NAS) Management", RFC 5607, July 2009.
- [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.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.
- [RFC6557] Lear, E. and P. Eggert, "Procedures for Maintaining the Time Zone Database", BCP 175, RFC 6557, February 2012.

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: April 8, 2013

L. Lhotka
CZ.NIC
October 05, 2012

Modeling JSON Text with YANG
draft-lhotka-netmod-yang-json-00

Abstract

This document defines rules for mapping data models expressed in YANG to configuration and operational state data encoded as JSON text. It does so by specifying a procedure for translating the subset of YANG-compatible XML documents to JSON text, and vice versa.

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 April 8, 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
3. Specification of the Translation Procedure	5
3.1. Names and Namespaces	6
3.2. Mapping XML Elements to JSON Objects	6
3.3. Mapping YANG Datatypes to JSON Values	7
3.3.1. Numeric Types	7
3.3.2. The "string" Type	7
3.3.3. The "boolean" Type	7
3.3.4. The "enumeration" Type	7
3.3.5. The "bits" Type	8
3.3.6. The "binary" Type	8
3.3.7. The "leafref" Type	8
3.3.8. The "identityref" Type	8
3.3.9. The "empty" Type	8
3.3.10. The "union" Type	8
3.3.11. The "instance-identifier" Type	8
3.4. Example	9
3.5. IANA Considerations	11
3.6. Security Considerations	11
3.7. Acknowledgments	11
4. References	12
4.1. Normative References	12
4.2. Informative References	12
Author's Address	13

1. Introduction

The aim of this document is define rules for mapping data models expressed in the YANG data modeling language [RFC6020] to configuration and operational state data encoded as JavaScript Object Notation (JSON) text [RFC4627]. The result can be potentially applied in two different ways:

1. JSON may be used instead of the standard XML [XML] encoding in the context of the NETCONF protocol [RFC6241] and/or with existing data models expressed in YANG. An example application is the YANG-API Protocol [YANG-API].
2. Other documents that choose JSON to represent structured data can use YANG for defining the data model, i.e., both syntactic and semantic constraints that the data have to satisfy.

JSON mapping rules could be specified in a similar way as the XML mapping rules in [RFC6020]. This would however require solving several problems. To begin with, YANG uses XPath [XPath] quite extensively, but XPath is not defined for JSON and such a definition would be far from straightforward.

In order to avoid these technical difficulties, this document employs an alternative approach: it defines a relatively simple procedure which allows to translate the subset of XML that can be modeled using YANG to JSON, and vice versa. Consequently, validation of a JSON text against a data model can be done by translating the JSON text to XML, which is then validated according to the rules stated in [RFC6020].

The translation procedure is adapted to YANG specifics and requirements, namely:

1. The translation is driven by a concrete YANG data model and uses information about data types to achieve better results than generic XML-JSON translation procedures.
2. Various document types are supported, namely configuration data, configuration + state data, RPC input and output parameters, and notifications.
3. XML namespaces specified in the data model are mapped to namespaces of JSON objects. However, explicit namespace identifiers are rarely needed in JSON text.
4. Translation of XML attributes, mixed content, comments and processing instructions is not supported.

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 [RFC6020]:

- o anyxml
- o augment
- o container
- o data model
- o data node
- o data tree
- o datatype
- o feature
- o identity
- o instance identifier
- o leaf
- o leaf-list
- o list
- o module
- o submodule

The following terms are defined in [XMLNS]:

- o local name
- o prefixed name
- o qualified name

3. Specification of the Translation Procedure

The translation procedure defines a 1-1 correspondence between the subset of YANG-compatible XML documents and JSON text. This means that the translation can be applied in both directions and is always invertible.

Any YANG-compatible XML document can be translated, except documents with mixed content. This is only a minor limitation since mixed content is marginal in YANG - it is allowed only in "anyxml" nodes.

The following subsections specify rules mainly for translating XML documents to JSON text. Rules for the inverse translation are stated only where necessary, otherwise they can be easily inferred.

REQUIRED parameters of the translation procedure are:

- o YANG data model,
- o type of the input XML document,
- o optional features (defined via the "feature" statement) that are considered active.

The permissible types of XML documents are listed in Table 1 together with the corresponding part of the data model that is used for the translation.

Document Type	Data Model Section
configuration and state data	main data tree
configuration	main data tree ("config true")
RPC input parameters	"input" nodes under "rpc"
RPC output parameters	"output" nodes under "rpc"
notification	"notification" nodes

Table 1: YANG Document Types

A particular application may decide to use only a subset of document types from Table 1. For instance, YANG-API Protocol [YANG-API] does not use notifications.

XML documents can be translated to JSON text only if they are valid instances of the YANG data model and selected document type, also taking into account the active features, if there are any.

3.1. Names and Namespaces

The local part of a JSON name is always identical to the local name of the corresponding XML element.

Each JSON name lives in a namespace which is uniquely identified by the name of the YANG module where the corresponding data node is defined. If the data node is defined in a submodule, then the namespace identifier is the name of the main module to which the submodule belongs. The translation procedure MUST correctly map YANG namespace URIs to YANG module names and vice versa.

The namespace SHALL be expressed in JSON text by prefixing the local name in the following way:

`<module name>:<local name>`

Figure 1: Encoding a namespace identifier with a local name.

The namespace identifier MUST be used for local names that are ambiguous, i.e., whenever the data model permits a sibling node with the same local name. Otherwise, the namespace identifier is OPTIONAL.

When mapping namespaces from JSON text to XML, the resulting XML document may use default namespace declarations (via the "xmlns" attribute), prefix-based namespace declarations (via attributes beginning with "xmlns:"), or any combination thereof following the rules stated in [XMLNS]. If prefixed names are used, their prefix SHOULD be the one defined by the "prefix" statement in the YANG module where each data node is defined.

3.2. Mapping XML Elements to JSON Objects

XML elements are translated to JSON objects in a straightforward way:

- o An XML element which is modeled as YANG leaf is translated to a name/value pair and the JSON datatype of the value is derived from the YANG datatype of the leaf (see Section 3.3 for the datatype mapping rules).
- o An XML element which is modeled as YANG container is translated to a JSON object.

- o A sequence of one or more sibling XML elements with the same qualified name, which is modeled as YANG list or leaf-list, is translated to a name/array pair. If the sequence is modeled as a leaf-list in YANG, then the array elements are primitive values whose type depends on the datatype of the leaf-list (see Section 3.3). If the sequence is modeled as a list in YANG, then the array elements are JSON objects.

Note that the same XML element may be translated in different ways, depending on the YANG data model. For example,

```
<foo>42</foo>
```

is translated to

```
"foo": 42
```

if the "foo" node is defined as a leaf with the "uint8" datatype, or to

```
"foo": ["42"]
```

if the "foo" node is defined as a leaf-list with the "string" datatype.

3.3. Mapping YANG Datatypes to JSON Values

3.3.1. Numeric Types

A value of one of the YANG numeric types ("int8", "int16", "int32", "int64", "uint8", "uint16", "uint32", "uint64" and "decimal64") is mapped to a JSON number using the same lexical representation.

3.3.2. The "string" Type

A "string" value is mapped to an identical JSON string, subject to JSON encoding rules.

3.3.3. The "boolean" Type

A "boolean" value is mapped to the corresponding JSON value 'true' or 'false'.

3.3.4. The "enumeration" Type

An "enumeration" value is mapped in the same way as a string except that the permitted values are defined by "enum" statements in YANG.

3.3.5. The "bits" Type

A "bits" value is mapped to a string identical to the lexical representation of this value in XML, i.e., space-separated names representing the individual bit values that are set.

3.3.6. The "binary" Type

A "binary" value is mapped to a JSON string identical to the lexical representation of this value in XML, i.e., base64-encoded binary data.

3.3.7. The "leafref" Type

A "leafref" value is mapped according to the same rules as the type of the leaf being referred to.

3.3.8. The "identityref" Type

An "identityref" value is mapped to a string representing the qualified name of the identity. Its namespace MAY be expressed as shown in Figure 1. If the namespace part is not present, the namespace of the name of the JSON object containing the value is assumed.

3.3.9. The "empty" Type

An "empty" value is mapped to '[null]', i.e., an array with the 'null' value being its only element.

This representation was chosen instead of using simply 'null' in order to facilitate the use of empty leafs in common programming languages. When used in a boolean context, the '[null]' value, unlike 'null', evaluates to 'true'.

3.3.10. The "union" Type

YANG "union" type represents a choice among multiple alternative types. The actual type of the XML value MUST be determined using the procedure specified in Sec. 9.12 of [RFC6020] and the mapping rules for that type are used.

3.3.11. The "instance-identifier" Type

An "instance-identifier" value is a string representing a simplified XPath specification. It is mapped to an analogical JSON string in which all occurrences of XML namespace prefixes are either removed or replaced with the corresponding module name according to the rules of

Section 3.1.

When translating such a value from JSON to XML, all components of the instance-identifier MUST be given appropriate XML namespace prefixes. It is RECOMMENDED that these prefixes be those defined via the "prefix" statement in the corresponding YANG modules.

3.4. Example

Consider a simple data model defined by the following YANG module:

```
module ex-json {  
    namespace "http://example.com/ex-json";  
    prefix "ej";  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    container top {  
        list address {  
            key "seqno";  
            leaf seqno {  
                type uint8;  
            }  
            leaf ip {  
                type inet:ip-address;  
                mandatory "true";  
            }  
        }  
        container phases {  
            typedef angle {  
                type decimal64 {  
                    fraction-digits "2";  
                }  
                units "radians";  
            }  
            leaf max-phase {  
                default "6.28";  
                type angle;  
            }  
            leaf-list phase {  
                type angle;  
                must ". <= ../max-phase";  
                min-elements "1";  
            }  
        }  
    }  
}
```

Figure 2: Example YANG module.

By using the translation procedure defined in this document, we can conclude that the following JSON text is valid according to the data model:

```
{
  "top": {
    "address": [
      {
        "seqno": 1,
        "ip": "192.0.2.1"
      },
      {
        "seqno": 2,
        "ip": "2001:db8:0:1::1"
      }
    ],
    "phases": {
      "phase": [
        "0.79",
        "1.04",
        "3.14"
      ]
    }
  }
}
```

Figure 3: Example JSON text.

3.5. IANA Considerations

TBD.

3.6. Security Considerations

TBD.

3.7. Acknowledgments

The author wishes to thank Andy Bierman, Martin Bjorklund and Phil Shafer for their helpful comments and suggestions.

4. References

4.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, September 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "NETCONF Configuration Protocol", RFC 6241, June 2011.
- [XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <<http://www.w3.org/TR/2006/REC-xml-20060816>>.
- [XMLNS] Bray, T., Hollander, D., Layman, A., Tobin, R., and H. Thompson, "Namespaces in XML 1.0 (Third Edition)", World Wide Web Consortium Recommendation REC-xml-names-20091208, December 2009, <<http://www.w3.org/TR/2009/REC-xml-names-20091208>>.

4.2. Informative References

- [XPath] Clark, J., "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <<http://www.w3.org/TR/1999/REC-xpath-19991116>>.
- [YANG-API] Bierman, A. and M. Bjorklund, "YANG-API Protocol", draft-bierman-netconf-yang-api-00 (work in progress), May 2012.

Author's Address

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

