NETMOD
Internet-Draft
Intended status: Informational
Expires: April 16, 2017

I. Chen
Kuatro Technologies

October 13, 2016

Grammar for Enterprise YANG Module Namespace
draft-chen-netmod-enterprise-yang-namespace-03

Abstract

This document defines the grammar to create enterprise YANG module namespaces that are globally unique, as required by the YANG modeling language.

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 16, 2017.

Copyright Notice

Table of Contents

1. Introduction

The use of a standard data modeling language YANG [RFC7950] together with Network Configuration Protocol (NETCONF) [RFC6241] allows for the creation of a standard network configuration interface. YANG further allows vendors to customize standard YANG modules and to create enterprise YANG modules that adapt standard YANG modules to different devices and features. To identify YANG modules, [RFC7950] requires YANG module namespaces of all YANG modules, both standard YANG modules and enterprise YANG modules, be globally unique. [RFC7950] also recommends that enterprise YANG modules have module names that are globally unique.

Based the module naming convention recommended in [RFC7950], this document defines the grammar to create YANG module namespaces for enterprise YANG modules that result in globally unique namespaces.

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].

2. YANG Module Name and Namespace Requirements and Recommendations

[RFC7950] consists of several YANG module name and namespace requirements and recommendations.

[RFC7950] Section 5.3 paragraph 1 states that

Each module is bound to a distinct XML namespace [XML-NAMES], which is a globally unique URI [RFC3986].

[RFC7950] Section 5.3 paragraph 3 states that

XML namespaces for private modules are assigned by the organization owning the module without a central registry. Namespace URIs MUST be chosen so they cannot collide with standard or other enterprise namespaces -- for example, by using the enterprise or organization name in the namespace.

[RFC7950] Section 6.2.1 paragraph 1 states that

Each identifier is valid in a namespace that depends on the type of the YANG item being defined. All identifiers defined in a namespace MUST be unique.

o All module and submodule names share the same global module identifier namespace.

The requirement in [RFC7950] Section 6.2.1 means that even enterprise YANG module names must be globally unique. The recommendation in [RFC7950] Section 5.3 means that it is desirable to define enterprise YANG modules names to use an organization's name as a prefix.

3. Enterprise YANG Module Namespace Grammar

This section defines the grammar to create globally unique enterprise YANG module namespaces. The grammar defines a namespace to be a Uniform Resource Name (URN) under the top-level "rdns" name identifier [rdns], followed by an organization's reverse registered domain name and optional sub-domain hierarchies, and ending with the module name with the organization's name as the prefix.

<namespace> = urn:rdns:<reverse-dns>:<sub-domain><module-name>

<reverse-dns> = An organization's registered domain name in reverse

<sub-domain> = Empty string or additional levels of hierarchy defined within a domain in which each level is delimited by colons

<module-name> = <organization-prefix>-<function>

<function> = A string that describes the function provided by the YANG module

In discussions on the NETMOD working group mailing list, there are suggestions that the top-level name identifier should be "yang" instead of "rdns". While the final decision on the exact name identifier might not be "rdns", the grammar described in this document is expected to remain as described above.

4. Usage Example

Suppose a vendor has a registered domain name "example.com". This vendor has also chosen to place all of its YANG modules under the "yang" sub-domain. Following the enterprise YANG module namespace grammar described in this document, the vendor ends up with the patterns below.

<reverse-dns> = com:example

<sub-domain> = yang

<namespace> = urn:rdns:com:example:yang:<module-name>

<module-name> = example-<function>

As a result, this vendor's OSPF YANG module has the namespace "urn:rdns:com:example:yang:example-ospf".

5. Security Considerations

TBD.

6. IANA Considerations

This document does not register any new names with IANA. The registration of the new "rdns" name is done in [rdns].

7. References

7.1. Normative References

[rdns] Chen, I., "Work in progress, 'draft-chen-rdns-urn-07.txt'", September 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[XML-NAMES]
Bray, T., Hollander, D., Layman, A., Tobin, R., and H. Thompson, "Namespaces in XML 1.0 (Third Edition)", Recommendation REC-xml-names-20091208, December 2009.

7.2. Informative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

Author’s Address

I. Chen
Kuatro Technologies

Email: ichen@kuatrotech.com

Internet Draft
<draft-chen-rdns-urn-07.txt>
Category: Informational
Expires in 6 months

I. Chen
Ericsson

September 19, 2016

A Uniform Resource Name (URN) Namespace for Enterprise YANG Modules
<draft-chen-rdns-urn-07.txt>

Status of this Memo

Distribution of this memo is unlimited.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire in 6 months.

Copyright Notice

Abstract

This document describes the Namespace Identifier (NID) for Uniform Resource Namespace (URN) resources to uniquely identify enterprise YANG modules. This document defines a single top level "rdns" Namespace identifier (NID), with which organizations and enterprises can define Uniform Resource Name (URN) Namespaces to uniquely identify enterprise YANG modules.

Table of Contents

1. Introduction

The use of a standard data modeling language YANG [RFC6020] together with Network Configuration Protocol (NETCONF) [RFC6241] allows for the creation of a standard network management interface. A networking device that supports such a standard network configuration interface supports NETCONF as well as a set of YANG modules, allowing administrators to manage data defined by the supported YANG modules in a single uniform manner, regardless of the make and model of the device.

To identify YANG modules, RFC 6020 Section 5.3 [RFC6020] requires that each YANG module, whether it is a standard YANG module or not, specify an XML namespace [XML-NAMES], and that the XML namespace be a globally unique Uniform Resource Identifier (URI) [RFC3986]. To date, IETF standard YANG modules register their XML namespaces with the IETF XML namespaces [RFC3688] that fall under the "ietf" Namespace Identifier (NID). Various standards governing bodies such as IEEE are also in the process of registering NIDs for their respective standard YANG module XML namespaces.

As a shortcut, this document registers the "rdns" NID for organizations such as commercial companies or open source communities to create globally unique XML namespaces when they create and publish enterprise YANG modules. An organization can use the "rdns" NID and append its registered domain name in reverse, followed by a string that is unique within its organization, to create a globally unique XML namespace for its enterprise YANG module without incurring extra effort to register a new NID.

2. Keywords

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 [RFC2119].

3. URN Specification for Enterprise YANG Module Namespace Identifier

Namespace ID:

Request "rdns"

Registration Information:

Version Number: 1

Date: <when submitted to IANA>

Declared Registrant of the Namespace:

Registering organization: IETF Netmod Working Group

Designated contact: ichen@kuatrotech.com

Declaration of Syntactic Structure:

An "rdns" URN is meant to be XML namespaces, and thus should follow the rules from both [RFC2141] and [XML-NAMES] for its character set and also when evaluating for lexical equivalence. For ease of determining lexical equivalence, all letters MUST be in lowercase letters. Based on these constraints, URNs that use the "rdns" NID shall have the following structures:

```
"rdns" URN       ::= urn:rdns:<reverse-dns>:<dss>

<reverse-dns>    ::= registered domain name in reverse, each label
                     separated by a colon (":") and all letters
                     MUST be written in lowercase letters

<dss>            ::= 1*<rdns-URN-char>

<rdns-URN-char>  ::= <rdns-trans> | "%" <rdns-hex> <rdns-hex>

<rdns-trans>     ::= <lower> | <number> | <other>

<rdns-hex>       ::= <number> | "a" | "b" | "c" | "d" | "e" |
                     "f"
```

<lower>, <number>, and <other> are the same as those defined in [RFC2141].

With the grammar above, a valid "rdns" URN MUST consist of "urn:rdns" in lowercase letters.

The reverse registered domain name <reverse-dns> is a mandatory string that is an organization's complete registered domain name in reverse. The structure of the string is an organization's domain name from the least specific label to the most specific label, using colons (":") to separate labels. The <reverse-dns> string is based on the A-label form [RFC5890] for internationalized labels, i.e., the labels as turned into ASCII by Punycode. All letters in <reverse-dns> MUST be in lowercase letters.

The domain specific string <dss> is a mandatory string with at least one <rdns-URN-char>. The structure of the string <dss> is opaque, because it is defined by the organization encoded in <reverse-dns>. <dss> is a string for the organization to identify the name or hierarchies of names the organization uses to identify its enterprise YANG module.

Relevant Ancillary Documentation:

See [RFC1034] and [RFC1035] for definitions and conventions of registered domain names, and [RFC5890] for the A-label form for internationalized labels within domain names.

Identifier Uniqueness Considerations:

An organization that provides the domain specific string <dss> MUST guarantee the uniqueness of <dss> within its organization. Using a <dss> that is unique within an organization in conjunction with a globally unique registered domain name (albeit in reverse) and the new "rdns" top-level NID, a URN is guaranteed to be globally unique.

Identifier Persistence Considerations:

Persistence of an "rdns" URN is dependent upon the organization that owns the registered domain name encoded in the URN to continue to own the domain name and also to not reassign the URN to a different YANG module. Organizations that change their domain names MUST republish their enterprise YANG modules to use "rdns" URNs with the new domain name.

In practice, an administrator consciously installs YANG modules in a device. Thus, in the unlikely event that there is a collision due to changing domain names, the administrator can detect the collision and rectify the situation by requesting that the offending organization republish its YANG modules with the correct

"rdns" URNs.

Process of Identifier Assignment:

The assignment of an "rdns" URN is delegated to the organization that has registered the domain name encoded in the URN.

For example, Ericsson registers for the domain name ericsson.com and can assign URNs with the prefix "urn:rdns:com:ericsson", where the <reverse-dns> portion of the URN is "com:ericsson". As mentioned above, the <dss> portion of the URN is assigned by the registrant of the domain name ericsson.com.

Process for Identifier Resolution:

"rdns" URNs are not intended to be accessible for global resolution. Rather, they are only intended to uniquely identify enterprise YANG modules (within a networking device). Resolution of an "rdns" URN is delegated to the organization owning the registered domain name encoded in the URN. If an organization that owns the registered domain name wishes for its "rdns" URNs to be resolvable, then the organization must register with the Resolution Discovery System [RFC2276].

Rules for Lexical Equivalence:

Two valid "rdns" URNs are identical if and only if the strings are identical.

Conformance with URN Syntax:

No special considerations.

Validation Mechanism:

Validation mechanism is controlled by the organization that owns the registered domain name. If an "rdns" URN contains an invalid domain name in the <reverse-dns> portion, then the URN is invalid.

In reality, an "rdns" URN is only meaningful in the context of YANG modules installed and supported in a device. Consequently, the "rdns" URNs in use should all be valid.

Scope:

The scope of an "rdns" URN is limited to enterprise YANG modules.

4. Namespace Considerations

[RFC6020] suggests that for enterprise YANG modules to have globally unique XML namespaces, one possibility is to use Uniform Resource Locators (URLs) based on an organization's registered domain name. However, in addition to being a globally unique identifier, a URL is also a resource locator, providing information about the resource's primary access mechanism. Consequently, an enterprise YANG module using a URL as its XML namespace also identifies the location of the resource, which is not necessarily the desired outcome. For example, an enterprise forced to use the URL http://www.example.com/yang/ospf as its YANG module XML namespace might not wish to make the YANG module available via HTTP [RFC2616], even though that is what using a URL implies. Using "rdns" URNs defined in this document yields globally unique XML namespaces which do not have the side effect of URLs that imply how to obtain resources.

5. Community Considerations

The "rdns" NID is intended for organizations such as enterprises and open source communities to easily create globally unique XML namespaces for enterprise YANG modules, without the need for all organizations to register their own NIDs.

6. Security Considerations

This document does not introduce new security considerations beyond those associated with the use and resolution of URNs in general.

7. IANA Considerations

This document defines a new URN NID registration for "rdns" in IANA's "Formal URN Namespace" registry. The completed registration template is in Section 3.

8. References

8.1. Normative References

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[XML-NAMES] Hollander, D., Tobin, R., Thompson, H., Bray, T., and A. Layman, "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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2141] Moats, R., "URN Syntax", RFC 2141, DOI 10.17487/RFC2141, May 1997, <http://www.rfc-editor.org/info/rfc2141>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

8.2. Informative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC2276] Sollins, K., "Architectural Principles of Uniform Resource Name Resolution", RFC 2276, DOI 10.17487/RFC2276, January 1998, <http://www.rfc-editor.org/info/rfc2276>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <http://www.rfc-editor.org/info/rfc2616>.

Author’s Address

I. Chen
ichen@kuatrotech.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
J. Dong
Huawei Technologies
D. Romascanu

October 31, 2016

A YANG Data Model for Hardware Management
draft-ietf-netmod-entity-01

Abstract

This document defines a YANG data model for the management of hardware on a single server.

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 May 4, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [I-D.ietf-netmod-rfc6020bis] data model for the management of hardware on a single server.

The data model includes configuration data and state data (status information and counters for the collection of statistics).

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. 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:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration data (read-write) and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Objectives

This section describes some of the design objectives for the hardware model.

- There are many common properties used to identify hardware components, which need to be supported in the hardware data model.

- There are many important information and states about the components, which needs to be collected from the devices which support the hardware data model.

- The hardware data model SHOULD be suitable for new implementations to use as is.

- The hardware data model defined in this document can be implemented on a system that also implements ENTITY-MIB, thus the mapping between the hardware data model and ENTITY-MIB SHOULD be clear.

3. Hardware Data Model

This document defines the YANG module "ietf-hardware", which has the following structure:

```
module: ietf-hardware
    +--ro hardware-state
    |  +--ro last-change?   yang:date-and-time
    |  +--ro component* [name]
    |     +--ro name              string
    |     +--ro class             identityref
    |     +--ro physical-index?   int32 {entity-mib}?
    |     +--ro description?      string
    |     +--ro contained-in*     -> ../../component/name
    |     +--ro contains-child*   -> ../../component/name
    |     +--ro parent-rel-pos?   int32
    |     +--ro hardware-rev?     string
    |     +--ro firmware-rev?     string
    |     +--ro software-rev?     string
```

```
|     +--ro serial-num?       string
|     +--ro mfg-name?         string
|     +--ro model-name?       string
|     +--ro alias?            string
|     +--ro asset-id?         string
|     +--ro is-fru?           boolean
|     +--ro mfg-date?         yang:date-and-time
|     +--ro uri*              inet:uri
|     +--ro uuid?             yang:uuid
|     +--ro state {entity-state}?
|     |  +--ro state-last-changed?   yang:date-and-time
|     |  +--ro admin-state?          admin-state
|     |  +--ro oper-state?           oper-state
|     |  +--ro usage-state?          usage-state
|     |  +--ro alarm-status?         alarm-status
|     |  +--ro standby-status?       standby-status
|     +--ro sensor-data {entity-sensor}?
|        +--ro data-type?              sensor-data-type
|        +--ro data-scale?             sensor-data-scale
|        +--ro precision?              sensor-precision
|        +--ro value?                  sensor-value
|        +--ro oper-status?            sensor-status
|        +--ro sensor-units-display?   string
|        +--ro value-timestamp?        yang:date-and-time
|        +--ro value-update-rate?      uint32
|  +--rw hardware {hardware-config}?
|     +--rw component* [name]
|        +--rw name           string
|        +--rw serial-num?    string
|        +--rw alias?         string
|        +--rw asset-id?      string
|        +--rw uri*           inet:uri
|        +--rw admin-state?   admin-state {entity-state}?
|
|  notifications:
|    +---n harwdare-state-change
|    +---n hardware-state-oper-enabled {entity-state}?
|    |  +--ro name?           -> /hardware-state/component/name
|    |  +--ro admin-state?
|    |  |       -> /hardware-state/component/state/admin-state
|    |  +--ro alarm-status?
|    |          -> /hardware-state/component/state/alarm-status
|    +---n hardware-state-oper-disabled {entity-state}?
|       +--ro name?           -> /hardware-state/component/name
|       +--ro admin-state?
|       |       -> /hardware-state/component/state/admin-state
|       +--ro alarm-status?
|               -> /hardware-state/component/state/alarm-status
```

3.1. The Components Lists

The data model for hardware presented in this document uses a flat list of components. Each component in the list is identified by its name. Furthermore, each component has a mandatory "class" leaf.

The "iana-entity" module defines YANG identities for the hardware types in the IANA-maintained "IANA-ENTITY-MIB" registry.

The "class" leaf is a YANG identity that describes the type of the hardware. Vendors are encouraged to either directly use one of the common IANA-defined identities, or derive a more specific identity from one of them.

There is one optional list of configured components ("/hardware/component"), and a separate list for the operational state of all components ("/hardware-state/component").

4. Relationship to ENTITY-MIB

If the device implements the ENTITY-MIB [RFC6933], each entry in the /hardware-state/component list is mapped to one EntPhysicalEntry. Objects that are writable in the MIB are mapped to nodes in the /hardware/component list.

The "physical-index" leaf MUST contain the value of the corresponding entPhysicalEntry’s entPhysicalIndex.

The "class" leaf is mapped to both entPhysicalClass and entPhysicalVendorType. If the value of the "class" leaf is an identity that is either derived from or is one of the identities in the "iana-entity" module, then entPhysicalClass contains the corresponding IANAPhysicalClass enumeration value. Otherwise, entPhysicalClass contains the IANAPhysicalClass value "other(1)". Vendors are encouraged to define an identity (derived from an identity in "iana-entity" if possible) for each enterprise-specific registration identifier used for entPhysicalVendorType, and use that identity for the "class" leaf.

The following tables list the YANG data nodes with corresponding objects in the ENTITY-MIB.

YANG data node in /hardware-state/component	ENTITY-MIB object
name	entPhysicalName
class	entPhysicalClass
	entPhysicalVendorType
physical-index	entPhysicalIndex
description	entPhysicalDescr
contained-in	entPhysicalContainedIn
contains-child	entPhysicalChildIndex
parent-rel-pos	entPhysicalParentRelPos
hardware-rev	entPhysicalHardwareRev
firmware-rev	entPhysicalFirmwareRev
software-rev	entPhysicalSoftwareRev
serial-num	entPhysicalSerialNum
mfg-name	entPhysicalMfgName
model-name	entPhysicalModelName
alias	entPhysicalAlias
asset-id	entPhysicalAssetID
is-fru	entPhysicalIsFRU
mfg-date	entPhysicalMfgDate
uri	entPhysicalUris
uuid	entPhysicalUUID

YANG data nodes and related ENTITY-MIB objects

5. Relationship to ENTITY-SENSOR-MIB

TBD relationship to [RFC3433].

6. Relationship to ENTITY-STATE-MIB

TBD relationship to [RFC4268].

7. Hardware YANG Module

<CODE BEGINS> file "ietf-hardware@2016-10-25.yang"

```
module ietf-hardware {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-hardware";
  prefix hw;

  import ietf-inet-types {
    prefix inet;
  }
```

```
  import ietf-yang-types {
    prefix yang;
  }
  import iana-entity {
    prefix ianaent;
  }

  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: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

     Editor:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>

     Editor:   Jie Dong
               <mailto:jie.dong@huawei.com>

     Editor:   Dan Romascanu
               <mailto:dromasca@gmail.com>";

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

  description
    "This module contains a collection of YANG definitions for
     managing hardware.

     Copyright (c) 2016 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.: update the date below with the date of RFC publication
      // and remove this note.
      revision 2016-10-25 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: A YANG Data Model for Hardware Management";
      }

      /*
       * Features
       */

      feature entity-mib {
        description
          "This feature indicates that the device implements
           the ENTITY-MIB.";
        reference "RFC 6933: Entity MIB (Version 4)";
      }

      feature hardware-config {
        description
          "Indicates that the server supports configuration of
           hardware components.";
      }

      feature entity-state {
        description
          "Indicates the ENTITY-STATE-MIB objects are supported";
        reference "RFC 4268: Entity State MIB";
      }

      feature entity-sensor {
        description
          "Indicates the ENTITY-SENSOR-MIB objects are supported";
        reference "RFC 3433: Entity Sensor MIB";
      }

      /*
       * Typedefs
       */

      typedef admin-state {
        type enumeration {
          enum unknown {
```

```
          value 1;
          description
            "The resource is unable to report administrative state.";
        }
        enum locked {
          value 2;
          description
            "The resource is administratively prohibited from use.";
        }
        enum shutting-down {
          value 3;
          description
            "The resource usage is administratively limited to current
             instances of use.";
        }
        enum unlocked {
          value 4;
          description
            "The resource is not administratively prohibited from
             use.";
        }
      }
      description
        "Represents the various possible administrative states.";
      reference "RFC 4268: EntityAdminState";
    }

    typedef oper-state {
      type enumeration {
        enum unknown {
          value 1;
          description
            "The resource is unable to report operational state.";
        }
        enum disabled {
          value 2;
          description
            "The resource is totally inoperable.";
        }
        enum enabled {
          value 3;
          description
            "The resource is partially or fully operable.";
        }
        enum testing {
          value 4;
          description
            "The resource is currently being tested and cannot
```

```
            therefore report whether it is operational or not.";
        }
      }
      description
        "Represents the possible values of operational states.";
      reference "RFC 4268: EntityOperState";
    }

    typedef usage-state {
      type enumeration {
        enum unknown {
          value 1;
          description
            "The resource is unable to report usage state.";
        }
        enum idle {
          value 2;
          description
            "The resource is servicing no users.";
        }
        enum active {
          value 3;
          description
            "The resource is currently in use and it has sufficient
             spare capacity to provide for additional users.";
        }
        enum busy {
          value 4;
          description
            "The resource is currently in use, but it currently has
             no spare capacity to provide for additional users.";
        }
      }
      description
        "Represents the possible values of usage states.";
      reference "RFC 4268, EntityUsageState";
    }

    typedef alarm-status {
      type bits {
        bit unknown {
          position 0;
          description
            "The resource is unable to report alarm state.";
        }
        bit under-repair {
          position 1;
          description
```

```
            "The resource is currently being repaired, which, depending
             on the implementation, may make the other values in this
             bit string not meaningful.";
        }
        bit critical {
          position 2;
          description
            "One or more critical alarms are active against the
             resource.";
        }
        bit major {
          position 3;
          description
            "One or more major alarms are active against the
             resource.";
        }
        bit minor {
          position 4;
          description
            "One or more minor alarms are active against the
             resource.";
        }
        bit warning {
          position 5;
          description
            "One or more warning alarms are active against the
             resource. This alarm status is not defined in X.733.";
        }
        bit indeterminate {
          position 6;
          description
            "One or more alarms of whose perceived severity cannot be
             determined are active against this resource.
             This alarm status is not defined in X.733.";
        }
      }
      description
        "Represents the possible values of alarm status.
         An Alarm [RFC3877] is a persistent indication of an error or
         warning condition.

         When no bits of this attribute are set, then no active
         alarms are known against this component and it is not under
         repair.";
      reference "RFC 4268: EntityAlarmStatus";
    }

    typedef standby-status {
```

```
      type enumeration {
        enum unknown {
          value 1;
          description
            "The resource is unable to report standby state.";
        }
        enum hot-standby {
          value 2;
          description
            "The resource is not providing service, but it will be
             immediately able to take over the role of the resource
             to be backed up, without the need for initialization
             activity, and will contain the same information as the
             resource to be backed up.";
        }
        enum cold-standby {
          value 3;
          description
            "The resource is to back up another resource, but will not
             be immediately able to take over the role of a resource
             to be backed up, and will require some initialization
             activity.";
        }
        enum providing-service {
          value 4;
          description
            "The resource is providing service.";
        }
      }
      description
        "Represents the possible values of standby status.";
      reference "RFC 4268: EntityStandbyStatus";
    }

    typedef sensor-data-type {
      type enumeration {
        enum other {
          value 1;
          description
            "A measure other than those listed below.";
        }
        enum unknown {
          value 2;
          description
            "An unknown measurement, or arbitrary, relative numbers";
        }
        enum volts-AC {
          value 3;
```

```
          description
            "A measure of electric potential (alternating current).";
        }
        enum volts-DC {
          value 4;
          description
            "A measure of electric potential (direct current).";
        }
        enum amperes {
          value 5;
          description
            "A measure of electric current.";
        }
        enum watts {
          value 6;
          description
            "A measure of power.";
        }
        enum hertz {
          value 7;
          description
            "A measure of frequency.";
        }
        enum celsius {
          value 8;
          description
            "A measure of temperature.";
        }
        enum percent-RH {
          value 9;
          description
            "A measure of percent relative humidity.";
        }
        enum rpm {
          value 10;
          description
            "A measure of shaft revolutions per minute.";
        }
        enum cmm {
          value 11;
          description
            "A measure of cubic meters per minute (airflow).";
        }
        enum truth-value {
          value 12;
          description
            "Value is one of 1 (true) or 2 (false)";
        }
```

```
      }
      description
        "An node using this data type represents the Entity Sensor
         measurement data type associated with a physical sensor
         value. The actual data units are determined by examining an
         node of this type together with the associated
         sensor-data-scale node.

         An node of this type SHOULD be defined together with nodes
         of type sensor-data-scale and
         sensor-precision. These three types are used to
         identify the semantics of an node of type
         sensor-value.";
      reference "RFC 3433: EntitySensorDataType";
    }

    typedef sensor-data-scale {
      type enumeration {
        enum yocto {
          value 1;
          description
            "Data scaling factor of 10^-24.";
        }
        enum zepto {
          value 2;
          description
            "Data scaling factor of 10^-21.";
        }
        enum atto {
          value 3;
          description
            "Data scaling factor of 10^-18.";
        }
        enum femto {
          value 4;
          description
            "Data scaling factor of 10^-15.";
        }
        enum pico {
          value 5;
          description
            "Data scaling factor of 10^-12.";
        }
        enum nano {
          value 6;
          description
            "Data scaling factor of 10^-9.";
        }
```

```
          enum micro {
            value 7;
            description
              "Data scaling factor of 10^-6.";
          }
          enum milli {
            value 8;
            description
              "Data scaling factor of 10^-3.";
          }
          enum units {
            value 9;
            description
              "Data scaling factor of 10^0.";
          }
          enum kilo {
            value 10;
            description
              "Data scaling factor of 10^3.";
          }
          enum mega {
            value 11;
            description
              "Data scaling factor of 10^6.";
          }
          enum giga {
            value 12;
            description
              "Data scaling factor of 10^9.";
          }
          enum tera {
            value 13;
            description
              "Data scaling factor of 10^12.";
          }
          enum exa {
            value 14;
            description
              "Data scaling factor of 10^15.";
          }
          enum peta {
            value 15;
            description
              "Data scaling factor of 10^18.";
          }
          enum zetta {
            value 16;
            description
```

```
            "Data scaling factor of 10^21.";
        }
        enum yotta {
          value 17;
          description
            "Data scaling factor of 10^24.";
        }
      }
      description
        "An node using this data type represents a data scaling
         factor, represented with an International System of Units (SI)
         prefix.  The actual data units are determined by examining an
         node of this type together with the associated
         sensor-data-type.

         An node of this type SHOULD be defined together with nodes
         of type sensor-data-type and sensor-precision.
         Together, associated nodes of these three types are used to
         identify the semantics of an node of type
         sensor-value.";
      reference "RFC 3433: EntitySensorDataScale";
    }

    typedef sensor-precision {
      type int32 {
        range "-8 .. 9";
      }
      description
        "An node using this data type represents a sensor
         precision range.

         An node of this type SHOULD be defined together with nodes
         of type sensor-data-type and sensor-data-scale.
         Together, associated nodes of these three types are used to
         identify the semantics of an node of type
         sensor-value.

         If an node of this type contains a value in the range 1 to 9,
         it represents the number of decimal places in the fractional
         part of an associated sensor-value fixed- point number.

         If an node of this type contains a value in the range -8 to
         -1, it represents the number of accurate digits in the
         associated sensor-value fixed-point number.

         The value zero indicates the associated sensor-value
         node is not a fixed-point number.
```

```
          Server implementors must choose a value for the associated
          sensor-precision node so that the precision and
          accuracy of the associated sensor-value node is
          correctly indicated.

          For example, a component representing a temperature
          sensor that can measure 0 degrees to 100 degrees C in 0.1
          degree increments, +/- 0.05 degrees, would have an
          sensor-precision value of '1', an
          sensor-data-scale value of 'units', and an
          sensor-value ranging from '0' to '1000'.  The
          sensor-value would be interpreted as
          'degrees C * 10'.";
      reference "RFC 3433: EntitySensorPrecision";
    }

    typedef sensor-value {
      type int32 {
        range "-1000000000 .. 1000000000";
      }
      description
       "An node using this data type represents an Entity Sensor
        value.

        An node of this type SHOULD be defined together with nodes
        of type sensor-data-type, sensor-data-scale, and
        sensor-precision.  Together, associated nodes of those
        three types are used to identify the semantics of an node of
        this data type.

        The semantics of an node using this data type are determined
        by the value of the associated sensor-data-type node.

        If the associated sensor-data-type node is equal to
        'voltsAC', 'voltsDC', 'amperes', 'watts', 'hertz',
        'celsius', or 'cmm', then an node of this type MUST
        contain a fixed point number ranging from -999,999,999 to
        +999,999,999.  The value -1000000000 indicates an underflow
        error. The value +1000000000 indicates an overflow error.  The
        sensor-precision indicates how many fractional digits
        are represented in the associated sensor-value node.

        If the associated sensor-data-type node is equal to
        'percentRH', then an node of this type MUST contain a
        number ranging from 0 to 100.

        If the associated sensor-data-type node is equal to
        'rpm', then an node of this type MUST contain a number
```

```
        ranging from -999,999,999 to +999,999,999.

        If the associated sensor-data-type node is equal to
        'truthvalue', then an node of this type MUST contain
        either the value 1 (true) or the value 2 (false)'.

        If the associated sensor-data-type node is equal to
        'other' or unknown', then an node of this type MUST
        contain a number ranging from -1000000000 to 1000000000.";
      reference "RFC 3433: EntitySensorValue";
    }

    typedef sensor-status {
      type enumeration {
        enum ok {
          value 1;
          description
            "Indicates that the server can obtain the sensor value.";
        }
        enum unavailable {
          value 2;
          description
            "Indicates that the server presently cannot obtain the
             sensor value.";
        }
        enum nonoperational {
          value 3;
          description
            "Indicates that the server believes the sensor is broken.
             The sensor could have a hard failure (disconnected wire),
             or a soft failure such as out-of-range, jittery, or wildly
             fluctuating readings.";
        }
      }
      description
        "An node using this data type represents the operational
         status of a physical sensor.";
      reference "RFC 3433: EntitySensorStatus";
    }

    /*
     * Operational state data nodes
     */

    container hardware-state {
      config false;
      description
        "Data nodes for the operational state of components.";
```

```
      leaf last-change {
        type yang:date-and-time;
        description
          "The time the '/hardware-state/component' list
           changed.";
      }

      list component {
        key name;
        description
          "List of components.";
        reference "RFC 6933: entPhysicalEntry";

        leaf name {
          type string;
          description
            "Administrative name assigned to this component.
             No restrictions apply. Not required to be the same as
             entPhysicalName.";
        }

        leaf class {
          type identityref {
            base ianaent:entity-physical-class;
          }
          mandatory true;
          description
            "An indication of the general hardware type
             of the component.";
          reference "RFC 6933: entPhysicalClass";
        }

        leaf physical-index {
          if-feature entity-mib;
          type int32 {
            range "1..2147483647";
          }
          description
            "The entPhysicalIndex for the entPhysicalEntry represented
             by this list entry.";
          reference "RFC 6933: entPhysicalIndex";
        }

        leaf description {
          type string;
          description
            "A textual description of component.  This node
             should contain a string that identifies the manufacturer's
```

```
            name for the component and should be set to a
            distinct value for each version or model of the
            component.";
          reference "RFC 6933: entPhysicalDescr";
        }

        leaf-list contained-in {
          type leafref {
            path "../../component/name";
          }
          description
            "The name of the component that 'contains'
             this component.";
          reference "RFC 6933: entPhysicalContainedIn";
        }

        leaf-list contains-child {
          type leafref {
            path "../../component/name";
          }
          description
            "The name of the contained component.";
          reference "RFC 6933: entPhysicalChildIndex";
        }

        leaf parent-rel-pos {
          type int32 {
            range "0 .. 2147483647";
          }
          description
            "An indication of the relative position of this child
             component among all its sibling components.  Sibling
             components are defined as components that share the
             same instance values of each of the contained-in
             and class elements.";
          reference "RFC 6933: entPhysicalParentRelPos";
        }

        leaf hardware-rev {
          type string;
          description
            "The vendor-specific hardware revision string for the
             component.  The preferred value is the hardware
             revision identifier actually printed on the component
             itself (if present).";
          reference "RFC 6933: entPhysicalHardwareRev";
        }
```

```
        leaf firmware-rev {
          type string;
          description
            "The vendor-specific firmware revision string for the
             component.";
          reference "RFC 6933: entPhysicalFirmwareRev";
        }

        leaf software-rev {
          type string;
          description
            "The vendor-specific software revision string for the
             component.";
          reference "RFC 6933: entPhysicalSoftwareRev";
        }

        leaf serial-num {
          type string;
          description
            "The vendor-specific serial number string for the
             component.  The preferred value is the serial number
             string actually printed on the component itself (if
             present).

             If a serial number has been configured for this component
             in /hardware/component/serial-num, this node contains
             the configured value.";
          reference "RFC 6933: entPhysicalSerialNum";
        }

        leaf mfg-name {
          type string;
          description
            "The name of the manufacturer of this physical component.
             The preferred value is the manufacturer name string
             actually printed on the component itself (if present).

             Note that comparisons between instances of the model-name,
             firmware-rev, software-rev, and the serial-num nodes are
             only meaningful amongst component with the same
             value of mfg-name.

             If the manufacturer name string associated with the
             physical component is unknown to the server, then this
             node will contain a zero-length string.";
          reference "RFC 6933: entPhysicalMfgName";
        }
```

```
        leaf model-name {
          type string;
          description
            "The vendor-specific model name identifier string
             associated with this physical component.  The preferred
             value is the customer-visible part number, which may be
             printed on the component itself.

             If the model name string associated with the physical
             component is unknown to the server, then this node will
             contain a zero-length string.";
          reference "RFC 6933: entPhysicalModelName";
        }

        leaf alias {
          type string {
            length "0 .. 32";
          }
          description
            "An 'alias' name for the component, as specified by
             a network manager, and provides a non-volatile 'handle'
             for the component.

             If an alias has been configured for this component in
             /hardware/component/alias, this node contains the
             configured value.  If no such alias has been configured,
             the server may set the value of this node to a locally
             unique value.";
          reference "RFC 6933: entPhysicalAlias";
        }

        leaf asset-id {
          type string {
            length "0 .. 32";
          }
          description
            "This node is a user-assigned asset tracking identifier
             (as specified by a network manager) for the component
             and provides non-volatile storage of this information.

             If an asset tracking identifier has been configured for
             this component in /hardware/component/addet-id, this
             node contains the configured value.";
          reference "RFC 6933: entPhysicalAssetID";
        }

        leaf is-fru {
          type boolean;
```

```
      description
        "This node indicates whether or not this component
         is considered a 'field replaceable unit' by the vendor.
         If this node contains the value 'true', then this
         component identifies a field replaceable unit.  For all
         components that are permanently contained within a field
         replaceable unit, the value 'false' should be returned
         for this node.";
      reference "RFC 6933: entPhysicalIsFRU";
    }

    leaf mfg-date {
      type yang:date-and-time;
      description
        "The date of manufacturing of the managed component.";
      reference "RFC 6933: entPhysicalMfgDate";
    }

    leaf-list uri {
      type inet:uri;
      description
        "This node contains identification information about the
         component.

         If uris have been configured for this component in
         /hardware/component/uri, this node contains the
         configured values.";
      reference "RFC 6933: entPhysicalUris";
    }

    leaf uuid {
      type yang:uuid;
      description
        "A Universally Unique Identifier of the component.";
      reference "RFC 6933: entPhysicalUUID";
    }

    container state {
      if-feature entity-state;
      description
        "State-related nodes";
      reference "RFC 4268: Entity State MIB";

      leaf state-last-changed {
        type yang:date-and-time;
        description
          "The date and time when the value of any of the
           admin-state, oper-state, usage-state, alarm-status, or
```

```
            standby-status changed for this component.

            If there has been no change since the last
            re-initialization of the local system, this node
            contains the date and time of local system
            initialization.  If there has been no change since the
            component was added to the local system, this node
            contains the date and time of the insertion.";
          reference "RFC 4268: entStateLastChanged";
        }

        leaf admin-state {
          type admin-state;
          description
            "The administrative state for this component.

            This node refers to an entities administrative
            permission to service both other entities within its
            containment hierarchy as well other users of its
            services defined by means outside the scope of this
            module.

            Some components exhibit only a subset of the
            remaining administrative state values.  Some entities
            cannot be locked, and hence this node exhibits only the
            'unlocked' state.  Other entities cannot be shutdown
            gracefully, and hence this node does not exhibit the
            'shutting-down' state.";
          reference "RFC 4268: entStateAdmin";
        }

        leaf oper-state {
          type oper-state;
          description
            "The operational state for this component.

            Note that this node does not follow the administrative
            state.  An administrative state of down does not predict
            an operational state of disabled.

            Note that some implementations may not be able to
            accurately report oper-state while the admin-state node
            has a value other than 'unlocked'.  In these cases, this
            node MUST have a value of 'unknown'.";
          reference "RFC 4268: entStateOper";
        }

        leaf usage-state {
```

```
            type usage-state;
            description
              "The usage state for this component.

               This node refers to a component's ability to service
               more components in a containment hierarchy.

               Some entities will exhibit only a subset of the usage
               state values.  Entities that are unable to ever service
               any entities within a containment hierarchy will always
               have a usage state of 'busy'.  Some entities will only
               ever be able to support one component within its
               containment hierarchy and will therefore only exhibit
               values of 'idle' and 'busy'.";
            reference "RFC 4268, entStateUsage";
          }

          leaf alarm-status {
            type alarm-status;
            description
              "The alarm status for this component.  It does not
               include the alarms raised on child components within its
               containment hierarchy.";
            reference "RFC 4268: entStateAlarm";
          }

          leaf standby-status {
            type standby-status;
            description
              "The standby status for this component.

               Some entities will exhibit only a subset of the
               remaining standby state values.  If this component
               cannot operate in a standby role, the value of this
               node will always be 'providing-service'.";
            reference "RFC 4268: entStateStandby";
          }
        }

        container sensor-data {
          when 'derived-from-or-self(../class,
                                     "ianaent:sensor")' {
            description
              "Sensor data nodes present for any component of type
               'sensor'";
          }
          if-feature entity-sensor;
          description
```

```
            "Sensor-related nodes.";
          reference "RFC 3433: Entity Sensor MIB";

          leaf data-type {
            type sensor-data-type;
            description
              "The type of data units associated with the
               sensor value";
            reference "RFC 3433: entPhySensorType";
          }

          leaf data-scale {
            type sensor-data-scale;
            description
              "The (power of 10) scaling factor associated
               with the sensor value";
            reference "RFC 3433: entPhySensorScale";
          }

          leaf precision {
            type sensor-precision;
            description
              "The number of decimal places of precision
               associated with the sensor value";
            reference "RFC 3433: entPhySensorPrecision";
          }

          leaf value {
            type sensor-value;
            description
              "The most recent measurement obtained by the server
               for this sensor.";
            reference "RFC 3433: entPhySensorValue";
          }

          leaf oper-status {
            type sensor-status;
            description
              "The operational status of the sensor.";
            reference "RFC 3433: entPhySensorOperStatus";
          }

          leaf sensor-units-display {
            type string;
            description
              "A textual description of the data units that should be
               used in the display of the sensor value.";
            reference "RFC 3433: entPhySensorUnitsDisplay";
```

```
          }

          leaf value-timestamp {
            type yang:date-and-time;
            description
              "The time the status and/or value of this sensor was
               last obtained by the server.";
            reference "RFC 3433: entPhySensorValueTimeStamp";
          }

          leaf value-update-rate {
            type uint32;
            units "milliseconds";
            description
              "An indication of the frequency that the server updates
               the associated 'value' node, representing in
               milliseconds.  The value zero indicates:

                - the sensor value is updated on demand (e.g.,
                  when polled by the server for a get-request),
                - the sensor value is updated when the sensor
                  value changes (event-driven),
                - the server does not know the update rate.";
            reference "RFC 3433: entPhySensorValueUpdateRate";
          }
        }
      }
    }

    /*
     * Configuration data nodes
     */

    container hardware {
      if-feature hardware-config;
      description
        "Configuration parameters for components.";

      list component {
        key name;
        description
          "List of configuration data for components.";

        leaf name {
          type string;
          description
            "Administrative name assigned to this component.
             No restrictions apply.";
```

```
        }

        leaf serial-num {
          type string;
          description
            "The vendor-specific serial number string for the
             component.  The preferred value is the serial number
             string actually printed on the component itself (if
             present).

             This node is indented to be used for components
             for which the server cannot determine the serial number.";
          reference "RFC 6933: entPhysicalSerialNum";
        }

        leaf alias {
          type string {
            length "0 .. 32";
          }
          description
            "This node is an 'alias' name for the component, as
             specified by a network manager, and provides a non-
             volatile 'handle' for the component.";
          reference "RFC 6933: entPhysicalAlias";
        }

        leaf asset-id {
          type string {
            length "0 .. 32";
          }
          description
            "This node is a user-assigned asset tracking identifier
             (as specified by a network manager) for the component";
          reference "RFC 6933: entPhysicalAssetID";
        }

        leaf-list uri {
          type inet:uri;
          description
            "This node contains identification information about the
             component.";
          reference "RFC 6933: entPhysicalUris";
        }

        leaf admin-state {
          if-feature entity-state;
          type admin-state;
          description
```

```
            "The administrative state for this component.

             This node refers to a component's administrative
             permission to service both other entities within
             its containment hierarchy as well other users of
             its services defined by means outside the scope
             of this module.

             Some components exhibit only a subset of the
             remaining administrative state values.  Some entities
             cannot be locked, and hence this node exhibits only
             the 'unlocked' state.  Other entities cannot be shutdown
             gracefully, and hence this node does not exhibit the
             'shutting-down' state.";
          reference "RFC 4268, entStateAdmin";
        }
      }
    }

    /*
     * Notifications
     */

    notification harwdare-state-change {
      description
        "A hardware-state-change notification is generated when the
         value of /hardware-state/last-change changes.";
      reference "RFC 6933, entConfigChange";
    }

    notification hardware-state-oper-enabled {
      if-feature entity-state;
      description
        "A hardware-state-oper-enabled notification signifies that
         a component has transitioned into the 'enabled' state.";

      leaf name {
        type leafref {
          path "/hardware-state/component/name";
        }
        description
          "The name of the component that has transitioned into the
           'enabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/hardware-state/component/state/admin-state";
        }
```

```
        description
          "The administrative state for the component.";
      }
      leaf alarm-status {
        type leafref {
          path "/hardware-state/component/state/alarm-status";
        }
        description
          "The alarm status for the component.";
      }
      reference "RFC 4268, entStateOperEnabled";
    }

    notification hardware-state-oper-disabled {
      if-feature entity-state;
      description
        "A hardware-state-oper-disabled notification signifies that
         a component has transitioned into the 'disabled' state.";

      leaf name {
        type leafref {
          path "/hardware-state/component/name";
        }
        description
          "The name of the component that has transitioned into the
           'disabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/hardware-state/component/state/admin-state";
        }
        description
          "The administrative state for the component.";
      }
      leaf alarm-status {
        type leafref {
          path "/hardware-state/component/state/alarm-status";
        }
        description
          "The alarm status for the component.";
      }
      reference "RFC 4268, entStateOperDisabled";
    }

  }

  <CODE ENDS>
```

```
   <CODE BEGINS> file "iana-entity@2016-10-25.yang"

   module iana-entity {
     namespace "urn:ietf:params:xml:ns:yang:iana-entity";
     prefix ianaent;

     organization "IANA";
     contact
       "        Internet Assigned Numbers Authority

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

        Tel:    +1 310 823 9358
        <mailto:iana@iana.org>";


     description
       "IANA defined identities for physical class.";
     reference
       "https://www.iana.org/assignments/ianaentity-mib/ianaentity-mib";

     // 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 2016-10-25 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for hardware Management";
     }

     /*
      * Identities
      */

     identity entity-physical-class {
       description
         "This identity is the base for all physical entity class
          identifiers.";
     }


     identity unknown {
       base ianaent:entity-physical-class;
```

```
      description
        "This identity is applicable if the physical entity class
         is unknown to the server.";
    }

    identity chassis {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is an overall container for networking equipment.  Any class
         of physical entity, except a stack, may be contained within a
         chassis; a chassis may only be contained within a stack.";
    }

    identity backplane {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of device for aggregating and forwarding
         networking traffic, such as a shared backplane in a modular
         ethernet switch.  Note that an implementation may model a
         backplane as a single physical entity, which is actually
         implemented as multiple discrete physical components (within a
         chassis or stack).";
    }

    identity container {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is capable of containing one or more removable physical
         entities, possibly of different types.  For example, each
         (empty or full) slot in a chassis will be modeled as a
         container.  Note that all removable physical entities should
         be modeled within a container entity, such as field-
         replaceable modules, fans, or power supplies.  Note that all
         known containers should be modeled by the agent, including
         empty containers.";
    }

    identity power-supply {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is a power-supplying component.";
    }

    identity fan {
```

```
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is a fan or other heat-reduction component.";
    }

    identity sensor {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of sensor, such as a temperature sensor within a
         router chassis.";
    }

    identity module {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of self-contained sub-system.  If a 'module'
         entity is removable, then it should be modeled within a
         container entity; otherwise, it should be modeled directly
         within another physical entity (e.g., a chassis or another
         module).";
    }

    identity port {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of networking port, capable of receiving and/or
         transmitting networking traffic.";
    }

    identity stack {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of super-container (possibly virtual) intended to
         group together multiple chassis entities.  A stack may be
         realized by a 'virtual' cable, a real interconnect cable
         attached to multiple chassis, or multiple interconnect cables.
         A stack should not be modeled within any other physical
         entities, but a stack may be contained within another stack.
         Only chassis entities should be contained within a stack.";
    }

    identity cpu {
      base ianaent:entity-physical-class;
```

```
      description
        "This identity is applicable if the physical entity class
         is some sort of central processing unit.";
    }

    identity energy-object {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of energy object, i.e., a piece of equipment that
         is part of or attached to a communications network that is
         monitored, controlled, or aids in the management of another
         device for Energy Management.";
    }

    identity battery {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of battery.";
    }

    identity storage-drive {
      base ianaent:entity-physical-class;
      description
        "This identity is applicable if the physical entity class
         is some sort of entity with data storage capability as main
         functionality, e.g., disk drive (HDD), solid state device
         (SSD), hybrid (SSHD), object storage (OSD) or other.";
    }
  }


  <CODE ENDS>
```

8. 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-hardware

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-hardware
  namespace:    urn:ietf:params:xml:ns:yang:ietf-hardware
  prefix:       hw
  reference:    RFC XXXX
```

9. Security Considerations

TBD

10. Acknowledgements

TBD

11. Normative References

[I-D.ietf-netmod-rfc6020bis] Bjorklund, M., "The YANG 1.1 Data Modeling Language", draft-ietf-netmod-rfc6020bis-14 (work in progress), June 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3433] Bierman, A., Romascanu, D., and K. Norseth, "Entity Sensor Management Information Base", RFC 3433, DOI 10.17487/RFC3433, December 2002, <http://www.rfc-editor.org/info/rfc3433>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC4268] Chisholm, S. and D. Perkins, "Entity State MIB", RFC 4268, DOI 10.17487/RFC4268, November 2005, <http://www.rfc-editor.org/info/rfc4268>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <http://www.rfc-editor.org/info/rfc6933>.

Appendix A. Open Issues

- Should the model support pre-configuration of hardware components? The current model supports pre-configuration of components provided the operator knows the name of the component to be installed. A more useful model would use the parent component, class, and parent-rel-pos as identification. If the system detects a component and there is configuration available for the parent component, class, and parent-rel-pos then the system would instatiate the component with the provided name, and optionally additional parameters.

- Is there a need for a standard action 'reset' that can be used to reset components?

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Dan Romascanu

Email: dromasca@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 26, 2018

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
J. Dong
Huawei Technologies
D. Romascanu

January 22, 2018

A YANG Data Model for Hardware Management
draft-ietf-netmod-entity-08

Abstract

This document defines a YANG data model for the management of hardware on a single server.

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 July 26, 2018.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC7950] data model for the management of hardware on a single server.

The data model includes configuration and system state (status information and counters for the collection of statistics).

The data model in this document is designed to be compliant with the Network Management Datastore Architecture (NMDA) [I-D.ietf-netmod-revised-datastores]. For implementations that do not yet support NMDA, a temporary module with system state data only is defined in Appendix A.

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [I-D.ietf-netmod-revised-datastores] and are not redefined here:

- client
- server
- configuration
- system state
- operational state
- intended configuration

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Objectives

This section describes some of the design objectives for the hardware model.

- There are many common properties used to identify hardware components, which need to be supported in the hardware data model.
- There are many important information and states about the components, which needs to be collected from the devices which support the hardware data model.
- The hardware data model should be suitable for new implementations to use as is.
- The hardware data model defined in this document can be implemented on a system that also implements ENTITY-MIB, thus the mapping between the hardware data model and ENTITY-MIB should be clear.
- The data model should support pre-provisioning of hardware components.

3. Hardware Data Model

This document defines the YANG module "ietf-hardware", which has the following structure:

```
   module: ietf-hardware
       +--rw hardware
          +--ro last-change?   yang:date-and-time
          +--rw component* [name]
             +--rw name              string
             +--rw class             identityref
             +--ro physical-index?   int32 {entity-mib}?
             +--ro description?      string
             +--rw parent?           -> ../../component/name
             +--rw parent-rel-pos?   int32
             +--ro contains-child*   -> ../../component/name
             +--ro hardware-rev?     string
             +--ro firmware-rev?     string
             +--ro software-rev?     string
             +--ro serial-num?       string
             +--ro mfg-name?         string
             +--ro model-name?       string
             +--rw alias?            string
             +--rw asset-id?         string
             +--ro is-fru?           boolean
             +--ro mfg-date?         yang:date-and-time
             +--rw uri*              inet:uri
             +--ro uuid?             yang:uuid
             +--rw state {hardware-state}?
             |  +--ro state-last-changed?   yang:date-and-time
             |  +--rw admin-state?          admin-state
             |  +--ro oper-state?           oper-state
             |  +--ro usage-state?          usage-state
             |  +--ro alarm-state?          alarm-state
             |  +--ro standby-state?        standby-state
             +--ro sensor-data {hardware-sensor}?
                +--ro value?               sensor-value
                +--ro value-type?          sensor-value-type
                +--ro value-scale?         sensor-value-scale
                +--ro value-precision?     sensor-value-precision
                +--ro oper-status?         sensor-status
                +--ro units-display?       string
                +--ro value-timestamp?     yang:date-and-time
                +--ro value-update-rate?   uint32

     notifications:
       +---n hardware-state-change
       +---n hardware-state-oper-enabled {hardware-state}?
       |  +--ro name?          -> /hardware/component/name
       |  +--ro admin-state?   -> /hardware/component/state/admin-state
       |  +--ro alarm-state?   -> /hardware/component/state/alarm-state
       +---n hardware-state-oper-disabled {hardware-state}?
          +--ro name?          -> /hardware/component/name
```

```
          +--ro admin-state?   -> /hardware/component/state/admin-state
          +--ro alarm-state?   -> /hardware/component/state/alarm-state
```

3.1. The Components Lists

The data model for hardware presented in this document uses a flat list of components. Each component in the list is identified by its name. Furthermore, each component has a mandatory "class" leaf.

The "iana-hardware" module defines YANG identities for the hardware types in the IANA-maintained "IANA-ENTITY-MIB" registry.

The "class" leaf is a YANG identity that describes the type of the hardware. Vendors are encouraged to either directly use one of the common IANA-defined identities, or derive a more specific identity from one of them.

4. Relationship to ENTITY-MIB

If the device implements the ENTITY-MIB [RFC6933], each entry in the "/hardware/component" list in the operational state is mapped to one EntPhysicalEntry. Objects that are writable in the MIB are mapped to "config true" nodes in the "/hardware/component" list, except "entPhysicalSerialNum" which is writable in the MIB, but "config false" in the YANG module.

The "physical-index" leaf MUST contain the value of the corresponding entPhysicalEntry’s entPhysicalIndex.

The "class" leaf is mapped to both entPhysicalClass and entPhysicalVendorType. If the value of the "class" leaf is an identity that is either derived from or is one of the identities in the "iana-hardware" module, then entPhysicalClass contains the corresponding IANAPhysicalClass enumeration value. Otherwise, entPhysicalClass contains the IANAPhysicalClass value "other(1)". Vendors are encouraged to define an identity (derived from an identity in "iana-hardware" if possible) for each enterprise-specific registration identifier used for entPhysicalVendorType, and use that identity for the "class" leaf.

The following tables list the YANG data nodes with corresponding objects in the ENTITY-MIB.

YANG data node in /hardware/component	ENTITY-MIB object
name	entPhysicalName
class	entPhysicalClass
	entPhysicalVendorType
physical-index	entPhysicalIndex
description	entPhysicalDescr
parent	entPhysicalContainedIn
parent-rel-pos	entPhysicalParentRelPos
contains-child	entPhysicalChildIndex
hardware-rev	entPhysicalHardwareRev
firmware-rev	entPhysicalFirmwareRev
software-rev	entPhysicalSoftwareRev
serial-num	entPhysicalSerialNum
mfg-name	entPhysicalMfgName
model-name	entPhysicalModelName
alias	entPhysicalAlias
asset-id	entPhysicalAssetID
is-fru	entPhysicalIsFRU
mfg-date	entPhysicalMfgDate
uri	entPhysicalUris
uuid	entPhysicalUUID

YANG Data Nodes and Related ENTITY-MIB Objects

5. Relationship to ENTITY-SENSOR-MIB

If the device implements the ENTITY-SENSOR-MIB [RFC3433], each entry in the "/hardware/component" list where the container "sensor-data" exists is mapped to one EntPhySensorEntry.

YANG data node in /hardware/component/sensor-data	ENTITY-SENSOR-MIB object
value	entPhySensorValue
value-type	entPhySensorType
value-scale	entPhySensorScale
value-precision	entPhySensorPrecision
oper-status	entPhySensorOperStatus
units-display	entPhySensorUnitsDisplay
value-timestamp	entPhySensorValueTimeStamp
value-update-rate	entPhySensorValueUpdateRate

YANG Data Nodes and Related ENTITY-SENSOR-MIB Objects

6. Relationship to ENTITY-STATE-MIB

If the device implements the ENTITY-STATE-MIB [RFC4268], each entry in the "/hardware/component" list where the container "state" exists is mapped to one EntStateEntry.

YANG data node in /hardware/component/state	ENTITY-STATE-MIB object
state-last-changed	entStateLastChanged
admin-state	entStateAdmin
oper-state	entStateOper
usage-state	entStateUsage
alarm-state	entStateAlarm
standby-state	entStateStandby

YANG Data Nodes and Related ENTITY-SENSOR-MIB Objects

7. Hardware YANG Module

<CODE BEGINS> file "ietf-hardware@2018-01-15.yang"

```
module ietf-hardware {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-hardware";
  prefix hw;

  import ietf-inet-types {
    prefix inet;
  }
```

```
  import ietf-yang-types {
    prefix yang;
  }
  import iana-hardware {
    prefix ianahw;
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Editor:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>

     Editor:   Jie Dong
               <mailto:jie.dong@huawei.com>

     Editor:   Dan Romascanu
               <mailto:dromasca@gmail.com>";

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

  description
    "This module contains a collection of YANG definitions for
     managing hardware.

     This data model is designed for the Network Management Datastore
     Architecture defined in RFC YYYY.

     Copyright (c) 2018 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.: update the date below with the date of RFC publication
     // and remove this note.
     revision 2018-01-15 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for Hardware Management";
     }

     /*
      * Features
      */

     feature entity-mib {
       description
         "This feature indicates that the device implements
          the ENTITY-MIB.";
       reference "RFC 6933: Entity MIB (Version 4)";
     }

     feature hardware-state {
       description
         "Indicates the ENTITY-STATE-MIB objects are supported";
       reference "RFC 4268: Entity State MIB";
     }

     feature hardware-sensor {
       description
         "Indicates the ENTITY-SENSOR-MIB objects are supported";
       reference "RFC 3433: Entity Sensor MIB";
     }

     /*
      * Typedefs
      */

     typedef admin-state {
       type enumeration {
         enum unknown {
           value 1;
           description
             "The resource is unable to report administrative state.";
         }
         enum locked {
           value 2;
           description
             "The resource is administratively prohibited from use.";
         }
```

```
        enum shutting-down {
          value 3;
          description
            "The resource usage is administratively limited to current
             instances of use.";
        }
        enum unlocked {
          value 4;
          description
            "The resource is not administratively prohibited from
             use.";
        }
      }
      description
        "Represents the various possible administrative states.";
      reference "RFC 4268: EntityAdminState";
    }

    typedef oper-state {
      type enumeration {
        enum unknown {
          value 1;
          description
            "The resource is unable to report its operational state.";
        }
        enum disabled {
          value 2;
          description
            "The resource is totally inoperable.";
        }
        enum enabled {
          value 3;
          description
            "The resource is partially or fully operable.";
        }
        enum testing {
          value 4;
          description
            "The resource is currently being tested and cannot
             therefore report whether it is operational or not.";
        }
      }
      description
        "Represents the possible values of operational states.";
      reference "RFC 4268: EntityOperState";
    }

    typedef usage-state {
```

```
        type enumeration {
          enum unknown {
            value 1;
            description
              "The resource is unable to report usage state.";
          }
          enum idle {
            value 2;
            description
              "The resource is servicing no users.";
          }
          enum active {
            value 3;
            description
              "The resource is currently in use and it has sufficient
               spare capacity to provide for additional users.";
          }
          enum busy {
            value 4;
            description
              "The resource is currently in use, but it currently has no
               spare capacity to provide for additional users.";
          }
        }
        description
          "Represents the possible values of usage states.";
        reference "RFC 4268, EntityUsageState";
      }

      typedef alarm-state {
        type bits {
          bit unknown {
            position 0;
            description
              "The resource is unable to report alarm state.";
          }
          bit under-repair {
            position 1;
            description
              "The resource is currently being repaired, which, depending
               on the implementation, may make the other values in this
               bit string not meaningful.";
          }
          bit critical {
            position 2;
            description
              "One or more critical alarms are active against the
               resource.";
```

```
        }
        bit major {
          position 3;
          description
            "One or more major alarms are active against the
             resource.";
        }
        bit minor {
          position 4;
          description
            "One or more minor alarms are active against the
             resource.";
        }
        bit warning {
          position 5;
          description
            "One or more warning alarms are active against the
             resource.";
        }
        bit indeterminate {
          position 6;
          description
            "One or more alarms of whose perceived severity cannot be
             determined are active against this resource.";
        }
      }
      description
        "Represents the possible values of alarm states.  An alarm is a
         persistent indication of an error or warning condition.

         When no bits of this attribute are set, then no active alarms
         are known against this component and it is not under repair.";
      reference "RFC 4268: EntityAlarmStatus";
    }

    typedef standby-state {
      type enumeration {
        enum unknown {
          value 1;
          description
            "The resource is unable to report standby state.";
        }
        enum hot-standby {
          value 2;
          description
            "The resource is not providing service, but it will be
             immediately able to take over the role of the resource to
             be backed up, without the need for initialization
```

```
             activity, and will contain the same information as the
             resource to be backed up.";
         }
         enum cold-standby {
           value 3;
           description
             "The resource is to back up another resource, but will not
              be immediately able to take over the role of a resource to
              be backed up, and will require some initialization
              activity.";
         }
         enum providing-service {
           value 4;
           description
             "The resource is providing service.";
         }
       }
       description
         "Represents the possible values of standby states.";
       reference "RFC 4268: EntityStandbyStatus";
     }

     typedef sensor-value-type {
       type enumeration {
         enum other {
           value 1;
           description
             "A measure other than those listed below.";
         }
         enum unknown {
           value 2;
           description
             "An unknown measurement, or arbitrary, relative numbers";
         }
         enum volts-AC {
           value 3;
           description
             "A measure of electric potential (alternating current).";
         }
         enum volts-DC {
           value 4;
           description
             "A measure of electric potential (direct current).";
         }
         enum amperes {
           value 5;
           description
             "A measure of electric current.";
```

```
        }
        enum watts {
          value 6;
          description
            "A measure of power.";
        }
        enum hertz {
          value 7;
          description
            "A measure of frequency.";
        }
        enum celsius {
          value 8;
          description
            "A measure of temperature.";
        }
        enum percent-RH {
          value 9;
          description
            "A measure of percent relative humidity.";
        }
        enum rpm {
          value 10;
          description
            "A measure of shaft revolutions per minute.";
        }
        enum cmm {
          value 11;
          description
            "A measure of cubic meters per minute (airflow).";
        }
        enum truth-value {
          value 12;
          description
            "Value is one of 1 (true) or 2 (false)";
        }
      }
      description
        "A node using this data type represents the sensor measurement
         data type associated with a physical sensor value. The actual
         data units are determined by examining a node of this type
         together with the associated sensor-value-scale node.

         A node of this type SHOULD be defined together with nodes of
         type sensor-value-scale and sensor-value-precision. These
         three types are used to identify the semantics of a node of
         type sensor-value.";
      reference "RFC 3433: EntitySensorDataType";
```

```
    }

    typedef sensor-value-scale {
      type enumeration {
        enum yocto {
          value 1;
          description
            "Data scaling factor of 10^-24.";
        }
        enum zepto {
          value 2;
          description
            "Data scaling factor of 10^-21.";
        }
        enum atto {
          value 3;
          description
            "Data scaling factor of 10^-18.";
        }
        enum femto {
          value 4;
          description
            "Data scaling factor of 10^-15.";
        }
        enum pico {
          value 5;
          description
            "Data scaling factor of 10^-12.";
        }
        enum nano {
          value 6;
          description
            "Data scaling factor of 10^-9.";
        }
        enum micro {
          value 7;
          description
            "Data scaling factor of 10^-6.";
        }
        enum milli {
          value 8;
          description
            "Data scaling factor of 10^-3.";
        }
        enum units {
          value 9;
          description
            "Data scaling factor of 10^0.";
```

```
        }
        enum kilo {
          value 10;
          description
            "Data scaling factor of 10^3.";
        }
        enum mega {
          value 11;
          description
            "Data scaling factor of 10^6.";
        }
        enum giga {
          value 12;
          description
            "Data scaling factor of 10^9.";
        }
        enum tera {
          value 13;
          description
            "Data scaling factor of 10^12.";
        }
        enum peta {
          value 14;
          description
            "Data scaling factor of 10^15.";
        }
        enum exa {
          value 15;
          description
            "Data scaling factor of 10^18.";
        }
        enum zetta {
          value 16;
          description
            "Data scaling factor of 10^21.";
        }
        enum yotta {
          value 17;
          description
            "Data scaling factor of 10^24.";
        }
      }
      description
        "A node using this data type represents a data scaling factor,
         represented with an International System of Units (SI) prefix.
         The actual data units are determined by examining a node of
         this type together with the associated sensor-value-type.
```

```
          A node of this type SHOULD be defined together with nodes of
          type sensor-value-type and sensor-value-precision.  Together,
          associated nodes of these three types are used to identify the
          semantics of a node of type sensor-value.";
      reference "RFC 3433: EntitySensorDataScale";
    }

    typedef sensor-value-precision {
      type int8 {
        range "-8 .. 9";
      }
      description
        "A node using this data type represents a sensor value
         precision range.

         A node of this type SHOULD be defined together with nodes of
         type sensor-value-type and sensor-value-scale.  Together,
         associated nodes of these three types are used to identify the
         semantics of a node of type sensor-value.

         If a node of this type contains a value in the range 1 to 9,
         it represents the number of decimal places in the fractional
         part of an associated sensor-value fixed- point number.

         If a node of this type contains a value in the range -8 to -1,
         it represents the number of accurate digits in the associated
         sensor-value fixed-point number.

         The value zero indicates the associated sensor-value node is
         not a fixed-point number.

         Server implementers must choose a value for the associated
         sensor-value-precision node so that the precision and accuracy
         of the associated sensor-value node is correctly indicated.

         For example, a component representing a temperature sensor
         that can measure 0 degrees to 100 degrees C in 0.1 degree
         increments, +/- 0.05 degrees, would have a
         sensor-value-precision value of '1', a sensor-value-scale
         value of 'units', and a sensor-value ranging from '0' to
         '1000'.  The sensor-value would be interpreted as
         'degrees C * 10'.";
      reference "RFC 3433: EntitySensorPrecision";
    }

    typedef sensor-value {
      type int32 {
        range "-1000000000 .. 1000000000";
```

```
      }
      description
       "A node using this data type represents a sensor value.

        A node of this type SHOULD be defined together with nodes of
        type sensor-value-type, sensor-value-scale, and
        sensor-value-precision.  Together, associated nodes of those
        three types are used to identify the semantics of a node of
        this data type.

        The semantics of a node using this data type are determined by
        the value of the associated sensor-value-type node.

        If the associated sensor-value-type node is equal to 'voltsAC',
        'voltsDC', 'amperes', 'watts', 'hertz', 'celsius', or 'cmm',
        then a node of this type MUST contain a fixed point number
        ranging from -999,999,999 to +999,999,999.  The value
        -1000000000 indicates an underflow error. The value +1000000000
        indicates an overflow error.  The sensor-value-precision
        indicates how many fractional digits are represented in the
        associated sensor-value node.

        If the associated sensor-value-type node is equal to
        'percentRH', then a node of this type MUST contain a number
        ranging from 0 to 100.

        If the associated sensor-value-type node is equal to 'rpm',
        then a node of this type MUST contain a number ranging from
        -999,999,999 to +999,999,999.

        If the associated sensor-value-type node is equal to
        'truth-value', then a node of this type MUST contain either the
        value 1 (true) or the value 2 (false)'.

        If the associated sensor-value-type node is equal to 'other' or
        unknown', then a node of this type MUST contain a number
        ranging from -1000000000 to 1000000000.";
      reference "RFC 3433: EntitySensorValue";
    }

    typedef sensor-status {
      type enumeration {
        enum ok {
          value 1;
          description
            "Indicates that the server can obtain the sensor value.";
        }
        enum unavailable {
```

```
          value 2;
          description
            "Indicates that the server presently cannot obtain the
             sensor value.";
        }
        enum nonoperational {
          value 3;
          description
            "Indicates that the server believes the sensor is broken.
             The sensor could have a hard failure (disconnected wire),
             or a soft failure such as out-of-range, jittery, or wildly
             fluctuating readings.";
        }
      }
      description
        "A node using this data type represents the operational status
         of a physical sensor.";
      reference "RFC 3433: EntitySensorStatus";
    }

    /*
     * Data nodes
     */

    container hardware {
      description
        "Data nodes representing components.

         If the server supports configuration of hardware components,
         then this data model is instantiated in the configuration
         datastores supported by the server.  The leaf-list 'datastore'
         for the module 'ietf-hardware' in the YANG library provides
         this information.";

      leaf last-change {
        type yang:date-and-time;
        config false;
        description
          "The time the '/hardware/component' list changed in the
           operational state.";
      }

      list component {
        key name;
        description
          "List of components.

           When the server detects a new hardware component, it
```

```
            initializes a list entry in the operational state.

            If the server does not support configuration of hardware
            components, list entries in the operational state are
            initialized with values for all nodes as detected by the
            implementation.

            Otherwise, the following procedure is followed:

              1. If there is an entry in the /hardware/component list in
                 the intended configuration with values for the nodes
                 'class', 'parent', 'parent-rel-pos' that are equal to
                 the detected values, then the list entry in operational
                 state is initialized with the configured values,
                 including the 'name'.

              2. Otherwise (i.e., there is no matching configuration
                 entry), the list entry in the operational state is
                 initialized with values for all nodes as detected by
                 the implementation.

            If the /hardware/component list in the intended
            configuration is modified, then the system MUST behave as if
            it re-initializes itself, and follow the procedure in (1).";
          reference "RFC 6933: entPhysicalEntry";

          leaf name {
            type string;
            description
              "The name assigned to this component.

               This name is not required to be the same as
               entPhysicalName.";
          }

          leaf class {
            type identityref {
              base ianahw:hardware-class;
            }
            mandatory true;
            description
              "An indication of the general hardware type of the
               component.";
            reference "RFC 6933: entPhysicalClass";
          }

          leaf physical-index {
            if-feature entity-mib;
```

```
          type int32 {
            range "1..2147483647";
          }
          config false;
          description
            "The entPhysicalIndex for the entPhysicalEntry represented
             by this list entry.";
          reference "RFC 6933: entPhysicalIndex";
        }

        leaf description {
          type string;
          config false;
          description
            "A textual description of component.  This node should
             contain a string that identifies the manufacturer's name
             for the component and should be set to a distinct value
             for each version or model of the component.";
          reference "RFC 6933: entPhysicalDescr";
        }

        leaf parent {
          type leafref {
            path "../../component/name";
            require-instance false;
          }
          description
            "The name of the component that physically contains this
             component.

             If this leaf is not instantiated, it indicates that this
             component is not contained in any other component.

             In the event that a physical component is contained by
             more than one physical component (e.g., double-wide
             modules), this node contains the name of one of these
             components.  An implementation MUST use the same name
             every time this node is instantiated.";
          reference "RFC 6933: entPhysicalContainedIn";
        }

        leaf parent-rel-pos {
          type int32 {
            range "0 .. 2147483647";
          }
          description
            "An indication of the relative position of this child
             component among all its sibling components.  Sibling
```

```
            components are defined as components that:

              o Share the same value of the 'parent' node; and

              o Share a common base identity for the 'class' node.

            Note that the last rule gives implementations flexibility
            in how components are numbered.  For example, some
            implementations might have a single number series for all
            components derived from 'ianahw:port', while some others
            might have different number series for different
            components with identities derived from 'ianahw:port' (for
            example, one for RJ45 and one for SFP).";

          reference "RFC 6933: entPhysicalParentRelPos";
        }

        leaf-list contains-child {
          type leafref {
            path "../../component/name";
          }
          config false;
          description
            "The name of the contained component.";
          reference "RFC 6933: entPhysicalChildIndex";
        }

        leaf hardware-rev {
          type string;
          config false;
          description
            "The vendor-specific hardware revision string for the
             component.  The preferred value is the hardware revision
             identifier actually printed on the component itself (if
             present).";
          reference "RFC 6933: entPhysicalHardwareRev";
        }

        leaf firmware-rev {
          type string;
          config false;
          description
            "The vendor-specific firmware revision string for the
             component.";
          reference "RFC 6933: entPhysicalFirmwareRev";
        }

        leaf software-rev {
```

```
          type string;
          config false;
          description
            "The vendor-specific software revision string for the
             component.";
          reference "RFC 6933: entPhysicalSoftwareRev";
        }

        leaf serial-num {
          type string;
          config false;
          description
            "The vendor-specific serial number string for the
             component.  The preferred value is the serial number
             string actually printed on the component itself (if
             present).";
          reference "RFC 6933: entPhysicalSerialNum";
        }

        leaf mfg-name {
          type string;
          config false;
          description
            "The name of the manufacturer of this physical component.
             The preferred value is the manufacturer name string
             actually printed on the component itself (if present).

             Note that comparisons between instances of the model-name,
             firmware-rev, software-rev, and the serial-num nodes are
             only meaningful amongst component with the same value of
             mfg-name.

             If the manufacturer name string associated with the
             physical component is unknown to the server, then this
             node is not instantiated.";
          reference "RFC 6933: entPhysicalMfgName";
        }

        leaf model-name {
          type string;
          config false;
          description
            "The vendor-specific model name identifier string
             associated with this physical component.  The preferred
             value is the customer-visible part number, which may be
             printed on the component itself.

             If the model name string associated with the physical
```

```
            component is unknown to the server, then this node is not
            instantiated.";
          reference "RFC 6933: entPhysicalModelName";
        }

        leaf alias {
          type string;
          description
            "An 'alias' name for the component, as specified by a
             network manager, and provides a non-volatile 'handle' for
             the component.

             If no configured value exists, the server MAY set the
             value of this node to a locally unique value in the
             operational state.

             A server implementation MAY map this leaf to the
             entPhysicalAlias MIB object.  Such an implementation needs
             to use some mechanism to handle the differences in size
             and characters allowed between this leaf and
             entPhysicalAlias.  The definition of such a mechanism is
             outside the scope of this document.";
          reference "RFC 6933: entPhysicalAlias";
        }

        leaf asset-id {
          type string;
          description
            "This node is a user-assigned asset tracking identifier for
             the component.

             A server implementation MAY map this leaf to the
             entPhysicalAssetID MIB object.  Such an implementation
             needs to use some mechanism to handle the differences in
             size and characters allowed between this leaf and
             entPhysicalAssetID.  The definition of such a mechanism is
             outside the scope of this document.";
          reference "RFC 6933: entPhysicalAssetID";
        }

        leaf is-fru {
          type boolean;
          config false;
          description
            "This node indicates whether or not this component is
             considered a 'field replaceable unit' by the vendor.  If
             this node contains the value 'true', then this component
             identifies a field replaceable unit.  For all components
```

```
           that are permanently contained within a field replaceable
           unit, the value 'false' should be returned for this
           node.";
        reference "RFC 6933: entPhysicalIsFRU";
      }

      leaf mfg-date {
        type yang:date-and-time;
        config false;
        description
          "The date of manufacturing of the managed component.";
        reference "RFC 6933: entPhysicalMfgDate";
      }

      leaf-list uri {
        type inet:uri;
        description
          "This node contains identification information about the
           component.";
        reference "RFC 6933: entPhysicalUris";
      }

      leaf uuid {
        type yang:uuid;
        config false;
        description
          "A Universally Unique Identifier of the component.";
        reference "RFC 6933: entPhysicalUUID";
      }

      container state {
        if-feature hardware-state;
        description
          "State-related nodes";
        reference "RFC 4268: Entity State MIB";

        leaf state-last-changed {
          type yang:date-and-time;
          config false;
          description
            "The date and time when the value of any of the
             admin-state, oper-state, usage-state, alarm-state, or
             standby-state changed for this component.

             If there has been no change since the last
             re-initialization of the local system, this node
             contains the date and time of local system
             initialization.  If there has been no change since the
```

```
            component was added to the local system, this node
            contains the date and time of the insertion.";
          reference "RFC 4268: entStateLastChanged";
        }

        leaf admin-state {
          type admin-state;
          description
            "The administrative state for this component.

            This node refers to a component's administrative
            permission to service both other components within its
            containment hierarchy as well other users of its
            services defined by means outside the scope of this
            module.

            Some components exhibit only a subset of the remaining
            administrative state values.  Some components cannot be
            locked, and hence this node exhibits only the 'unlocked'
            state.  Other components cannot be shutdown gracefully,
            and hence this node does not exhibit the 'shutting-down'
            state.";
          reference "RFC 4268: entStateAdmin";
        }

        leaf oper-state {
          type oper-state;
          config false;
          description
            "The operational state for this component.

            Note that this node does not follow the administrative
            state.  An administrative state of down does not predict
            an operational state of disabled.

            Note that some implementations may not be able to
            accurately report oper-state while the admin-state node
            has a value other than 'unlocked'.  In these cases, this
            node MUST have a value of 'unknown'.";
          reference "RFC 4268: entStateOper";
        }

        leaf usage-state {
          type usage-state;
          config false;
          description
            "The usage state for this component.
```

```
            This node refers to a component's ability to service
            more components in a containment hierarchy.

            Some components will exhibit only a subset of the usage
            state values.  Components that are unable to ever
            service any components within a containment hierarchy
            will always have a usage state of 'busy'.  Some
            components will only ever be able to support one
            component within its containment hierarchy and will
            therefore only exhibit values of 'idle' and 'busy'.";
          reference "RFC 4268, entStateUsage";
        }

        leaf alarm-state {
          type alarm-state;
          config false;
          description
            "The alarm state for this component.  It does not
             include the alarms raised on child components within its
             containment hierarchy.";
          reference "RFC 4268: entStateAlarm";
        }

        leaf standby-state {
          type standby-state;
          config false;
          description
            "The standby state for this component.

             Some components will exhibit only a subset of the
             remaining standby state values.  If this component
             cannot operate in a standby role, the value of this node
             will always be 'providing-service'.";
          reference "RFC 4268: entStateStandby";
        }
      }

      container sensor-data {
        when 'derived-from-or-self(../class,
                                   "ianahw:sensor")' {
          description
            "Sensor data nodes present for any component of type
             'sensor'";
        }
        if-feature hardware-sensor;
        config false;

        description
```

```
            "Sensor-related nodes.";
          reference "RFC 3433: Entity Sensor MIB";

          leaf value {
            type sensor-value;
            description
              "The most recent measurement obtained by the server
               for this sensor.

               A client that periodically fetches this node should also
               fetch the nodes 'value-type', 'value-scale', and
               'value-precision', since they may change when the value
               is changed.";
            reference "RFC 3433: entPhySensorValue";
          }

          leaf value-type {
            type sensor-value-type;
            description
              "The type of data units associated with the
               sensor value";
            reference "RFC 3433: entPhySensorType";
          }

          leaf value-scale {
            type sensor-value-scale;
            description
              "The (power of 10) scaling factor associated
               with the sensor value";
            reference "RFC 3433: entPhySensorScale";
          }

          leaf value-precision {
            type sensor-value-precision;
            description
              "The number of decimal places of precision
               associated with the sensor value";
            reference "RFC 3433: entPhySensorPrecision";
          }

          leaf oper-status {
            type sensor-status;
            description
              "The operational status of the sensor.";
            reference "RFC 3433: entPhySensorOperStatus";
          }

          leaf units-display {
```

```
            type string;
            description
              "A textual description of the data units that should be
               used in the display of the sensor value.";
            reference "RFC 3433: entPhySensorUnitsDisplay";
          }

          leaf value-timestamp {
            type yang:date-and-time;
            description
              "The time the status and/or value of this sensor was last
               obtained by the server.";
            reference "RFC 3433: entPhySensorValueTimeStamp";
          }

          leaf value-update-rate {
            type uint32;
            units "milliseconds";
            description
              "An indication of the frequency that the server updates
               the associated 'value' node, representing in
               milliseconds.  The value zero indicates:

                - the sensor value is updated on demand (e.g.,
                  when polled by the server for a get-request),
                - the sensor value is updated when the sensor
                  value changes (event-driven),
                - the server does not know the update rate.";
            reference "RFC 3433: entPhySensorValueUpdateRate";
          }
        }
      }
    }

    /*
     * Notifications
     */

    notification hardware-state-change {
      description
        "A hardware-state-change notification is generated when the
         value of /hardware/last-change changes in the operational
         state.";
      reference "RFC 6933, entConfigChange";
    }

    notification hardware-state-oper-enabled {
      if-feature hardware-state;
```

```
      description
        "A hardware-state-oper-enabled notification signifies that a
         component has transitioned into the 'enabled' state.";

      leaf name {
        type leafref {
          path "/hardware/component/name";
        }
        description
          "The name of the component that has transitioned into the
           'enabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/hardware/component/state/admin-state";
        }
        description
          "The administrative state for the component.";
      }
      leaf alarm-state {
        type leafref {
          path "/hardware/component/state/alarm-state";
        }
        description
          "The alarm state for the component.";
      }
      reference "RFC 4268, entStateOperEnabled";
    }

    notification hardware-state-oper-disabled {
      if-feature hardware-state;
      description
        "A hardware-state-oper-disabled notification signifies that a
         component has transitioned into the 'disabled' state.";

      leaf name {
        type leafref {
          path "/hardware/component/name";
        }
        description
          "The name of the component that has transitioned into the
           'disabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/hardware/component/state/admin-state";
        }
        description
```

```
            "The administrative state for the component.";
        }
        leaf alarm-state {
          type leafref {
            path "/hardware/component/state/alarm-state";
          }
          description
            "The alarm state for the component.";
        }
        reference "RFC 4268, entStateOperDisabled";
      }

    }
```

<CODE ENDS>

<CODE BEGINS> file "iana-hardware@2018-01-15.yang"

```
    module iana-hardware {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:iana-hardware";
      prefix ianahw;

      organization "IANA";
      contact
        "        Internet Assigned Numbers Authority

         Postal: ICANN
                 12025 Waterfront Drive, Suite 300
                 Los Angeles, CA  90094-2536
                 United States

         Tel:    +1 310 301 5800
         E-Mail: iana@iana.org>";
      // RFC Ed.: replace XXXX with actual RFC number and remove this
      // note.
      description
        "IANA defined identities for hardware class.

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

         Requests for new values should be made to IANA via
         email (iana@iana.org).

         Copyright (c) 2018 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).

        The initial version of this YANG module is part of RFC XXXX;
        see the RFC itself for full legal notices.";
    reference
      // RFC Ed.: replace PPPP with actual path and remove this note.
      "https://www.iana.org/assignments/PPPP";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    revision 2018-01-15 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for Hardware Management";
    }

    /*
     * Identities
     */

    identity hardware-class {
      description
        "This identity is the base for all hardware class
         identifiers.";
    }

    identity unknown {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is unknown
         to the server.";
    }

    identity chassis {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is an
         overall container for networking equipment.  Any class of
         physical component, except a stack, may be contained within a
         chassis; a chassis may only be contained within a stack.";
    }
```

```
identity backplane {
  base ianahw:hardware-class;
  description
    "This identity is applicable if the hardware class is some sort
     of device for aggregating and forwarding networking traffic,
     such as a shared backplane in a modular ethernet switch.  Note
     that an implementation may model a backplane as a single
     physical component, which is actually implemented as multiple
     discrete physical components (within a chassis or stack).";
}

identity container {
  base ianahw:hardware-class;
  description
    "This identity is applicable if the hardware class is capable
     of containing one or more removable physical entities,
     possibly of different types.  For example, each (empty or
     full) slot in a chassis will be modeled as a container.  Note
     that all removable physical components should be modeled
     within a container component, such as field-replaceable
     modules, fans, or power supplies.  Note that all known
     containers should be modeled by the agent, including empty
     containers.";
}

identity power-supply {
  base ianahw:hardware-class;
  description
    "This identity is applicable if the hardware class is a
     power-supplying component.";
}

identity fan {
  base ianahw:hardware-class;
  description
    "This identity is applicable if the hardware class is a fan or
     other heat-reduction component.";
}

identity sensor {
  base ianahw:hardware-class;
  description
    "This identity is applicable if the hardware class is some sort
     of sensor, such as a temperature sensor within a router
     chassis.";
}

identity module {
```

```
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is some sort
         of self-contained sub-system.  If a module component is
         removable, then it should be modeled within a container
         component; otherwise, it should be modeled directly within
         another physical component (e.g., a chassis or another
         module).";
    }

    identity port {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is some sort
         of networking port, capable of receiving and/or transmitting
         networking traffic.";
    }

    identity stack {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is some sort
         of super-container (possibly virtual) intended to group
         together multiple chassis entities.  A stack may be realized
         by a virtual cable, a real interconnect cable attached to
         multiple chassis, or multiple interconnect cables.  A stack
         should not be modeled within any other physical components,
         but a stack may be contained within another stack.  Only
         chassis components should be contained within a stack.";
    }

    identity cpu {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is some sort
         of central processing unit.";
    }

    identity energy-object {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is some sort
         of energy object, i.e., a piece of equipment that is part of
         or attached to a communications network that is monitored,
         controlled, or aids in the management of another device for
         Energy Management.";
    }
```

```
    identity battery {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is some sort
         of battery.";
    }

    identity storage-drive {
      base ianahw:hardware-class;
      description
        "This identity is applicable if the hardware class is some sort
         of component with data storage capability as main
         functionality, e.g., disk drive (HDD), solid state device
         (SSD), hybrid (SSHD), object storage (OSD) or other.";
    }
  }
```

<CODE ENDS>

8. IANA Considerations

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

The "iana-hardware" YANG module is intended to reflect the "IANA-ENTITY-MIB" MIB module so that if a new enumeration is added to the "IANAPhysicalClass" TEXTUAL-CONVENTION, the same class is added as an identity derived from "ianahw:hardware-class".

When the "iana-hardware" YANG module is updated, a new "revision" statement must be added in front of the existing revision statements.

8.1. URI Registrations

This document registers three 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-hardware
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-hardware
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-hardware-state
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

8.2. YANG Module Registrations

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

```
name:         iana-hardware
namespace:    urn:ietf:params:xml:ns:yang:iana-hardware
prefix:       ianahw
reference:    RFC XXXX

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

name:         ietf-hardware-state
namespace:    urn:ietf:params:xml:ns:yang:ietf-hardware-state
prefix:       hw-state
reference:    RFC XXXX
```

9. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in the YANG module "ietf-hardware" that 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:

/hardware/component/admin-state: Setting this node to 'locked' or 'shutting-down' can cause disruption of services ranging from those running on a port to those on an entire device, depending on the type of component.

Some of the readable data nodes in these YANG modules 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:

/hardware/component: The leafs in this list expose information about the physical components in a device, which may be used to identify the vendor, model, version, and specific device-identification information of each system component.

/hardware/component/sensor-data/value: This node may expose the values of particular physical sensors in a device.

/hardware/component/state: Access to this node allows one to figure out what the active and standby resources in a device are.

10. Acknowledgments

The authors wish to thank the following individuals, who all provided helpful comments on various draft versions of this document: Bart Bogaert, Timothy Carey, William Lupton, Juergen Schoenwaelder.

11. References

11.1. Normative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3433] Bierman, A., Romascanu, D., and K. Norseth, "Entity Sensor Management Information Base", RFC 3433, DOI 10.17487/RFC3433, December 2002, <https://www.rfc-editor.org/info/rfc3433>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4268] Chisholm, S. and D. Perkins, "Entity State MIB", RFC 4268, DOI 10.17487/RFC4268, November 2005, <https://www.rfc-editor.org/info/rfc4268>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC6933] Bierman, A., Romascanu, D., Quittek, J., and M. Chandramouli, "Entity MIB (Version 4)", RFC 6933, DOI 10.17487/RFC6933, May 2013, <https://www.rfc-editor.org/info/rfc6933>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams] Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-04 (work in progress), December 2017.

Appendix A. Hardware State Data Model

This non-normative appendix contains a data model designed as a temporary solution for implementations that do not yet support the Network Management Datastore Architecture (NMDA) defined in [I-D.ietf-netmod-revised-datastores]. It has the following structure:

```
module: ietf-hardware-state
    x--ro hardware
       x--ro last-change?   yang:date-and-time
       x--ro component* [name]
          x--ro name              string
          x--ro class             identityref
          x--ro physical-index?   int32 {entity-mib}?
          x--ro description?      string
          x--ro parent?           -> ../../component/name
          x--ro parent-rel-pos?   int32
          x--ro contains-child*   -> ../../component/name
          x--ro hardware-rev?     string
          x--ro firmware-rev?     string
          x--ro software-rev?     string
          x--ro serial-num?       string
          x--ro mfg-name?         string
          x--ro model-name?       string
          x--ro alias?            string
          x--ro asset-id?         string
          x--ro is-fru?           boolean
          x--ro mfg-date?         yang:date-and-time
```

```
            x--ro uri*              inet:uri
            x--ro uuid?             yang:uuid
            x--ro state {hardware-state}?
            |  x--ro state-last-changed?   yang:date-and-time
            |  x--ro admin-state?          hw:admin-state
            |  x--ro oper-state?           hw:oper-state
            |  x--ro usage-state?          hw:usage-state
            |  x--ro alarm-state?          hw:alarm-state
            |  x--ro standby-state?        hw:standby-state
            x--ro sensor-data {hardware-sensor}?
               x--ro value?               hw:sensor-value
               x--ro value-type?          hw:sensor-value-type
               x--ro value-scale?         hw:sensor-value-scale
               x--ro value-precision?     hw:sensor-value-precision
               x--ro oper-status?         hw:sensor-status
               x--ro units-display?       string
               x--ro value-timestamp?     yang:date-and-time
               x--ro value-update-rate?   uint32

     notifications:
       x---n hardware-state-change
       x---n hardware-state-oper-enabled {hardware-state}?
       |  x--ro name?          -> /hardware/component/name
       |  x--ro admin-state?   -> /hardware/component/state/admin-state
       |  x--ro alarm-state?   -> /hardware/component/state/alarm-state
       x---n hardware-state-oper-disabled {hardware-state}?
          x--ro name?          -> /hardware/component/name
          x--ro admin-state?   -> /hardware/component/state/admin-state
          x--ro alarm-state?   -> /hardware/component/state/alarm-state
```

A.1. Hardware State YANG Module

```
   <CODE BEGINS> file "ietf-hardware-state@2017-12-18.yang"

   module ietf-hardware-state {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-hardware-state";
     prefix hw-state;

     import ietf-inet-types {
       prefix inet;
     }
     import ietf-yang-types {
       prefix yang;
     }
     import iana-hardware {
       prefix ianahw;
     }
```

```
  import ietf-hardware {
    prefix hw;
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

     Editor:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>

     Editor:   Jie Dong
               <mailto:jie.dong@huawei.com>

     Editor:   Dan Romascanu
               <mailto:dromasca@gmail.com>";

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

  description
    "This module contains a collection of YANG definitions for
     monitoring hardware.

     This data model is designed as a temporary solution for
     implementations that do not yet support the Network Management
     Datastore Architecture (NMDA) defined in RFC YYYY.  Such an
     implementation cannot implement the module 'ietf-hardware'
     properly, since without NMDA support, it is not possible to
     distinguish between instances of nodes in the running
     configuration and operational state.

     The data model in this module is the same as the data model in
     'ietf-hardware', except all nodes are marked as 'config false'.

     If a server that implements this module but doesn't support NMDA
     also supports configuration of hardware components, it SHOULD
     also implement the module 'ietf-hardware' in the configuration
     datastores.  The corresponding state data is found in the
     '/hw-state:hardware' subtree.

     Copyright (c) 2017 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.: update the date below with the date of RFC publication
    // and remove this note.
    revision 2017-12-18 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for Hardware Management";
    }

    /*
     * Features
     */

    feature entity-mib {
      status deprecated;
      description
        "This feature indicates that the device implements
         the ENTITY-MIB.";
      reference "RFC 6933: Entity MIB (Version 4)";
    }

    feature hardware-state {
      status deprecated;
      description
        "Indicates the ENTITY-STATE-MIB objects are supported";
      reference "RFC 4268: Entity State MIB";
    }

    feature hardware-sensor {
      status deprecated;
      description
        "Indicates the ENTITY-SENSOR-MIB objects are supported";
      reference "RFC 3433: Entity Sensor MIB";
    }

    /*
```

```
      * Data nodes
      */

     container hardware {
       config false;
       status deprecated;
       description
         "Data nodes representing components.";

       leaf last-change {
         type yang:date-and-time;
         status deprecated;
         description
           "The time the '/hardware/component' list changed in the
            operational state.";
       }

       list component {
         key name;
         status deprecated;
         description
           "List of components.

            When the server detects a new hardware component, it
            initializes a list entry in the operational state.

            If the server does not support configuration of hardware
            components, list entries in the operational state are
            initialized with values for all nodes as detected by the
            implementation.

            Otherwise, the following procedure is followed:

              1. If there is an entry in the /hardware/component list in
                 the intended configuration with values for the nodes
                 'class', 'parent', 'parent-rel-pos' that are equal to
                 the detected values, then:

              1a. If the configured entry has a value for 'mfg-name'
                  that is equal to the detected value, or if the
                  'mfg-name' value cannot be detected, then the list
                  entry in the operational state is initialized with the
                  configured values for all configured nodes, including
                  the 'name'.

                  Otherwise, the list entry in the operational state is
                  initialized with values for all nodes as detected by
                  the implementation.  The implementation may raise an
```

```
                alarm that informs about the 'mfg-name' mismatch
                condition.  How this is done is outside the scope of
                this document.

             1b. Otherwise (i.e., there is no matching configuration
                 entry), the list entry in the operational state is
                 initialized with values for all nodes as detected by
                 the implementation.

           If the /hardware/component list in the intended
           configuration is modified, then the system MUST behave as if
           it re-initializes itself, and follow the procedure in (1).";
        reference "RFC 6933: entPhysicalEntry";

        leaf name {
          type string;
          status deprecated;
          description
            "The name assigned to this component.

             This name is not required to be the same as
             entPhysicalName.";
        }

        leaf class {
          type identityref {
            base ianahw:hardware-class;
          }
          mandatory true;
          status deprecated;
          description
            "An indication of the general hardware type of the
             component.";
          reference "RFC 6933: entPhysicalClass";
        }

        leaf physical-index {
          if-feature entity-mib;
          type int32 {
            range "1..2147483647";
          }
          status deprecated;
          description
            "The entPhysicalIndex for the entPhysicalEntry represented
             by this list entry.";
          reference "RFC 6933: entPhysicalIndex";
        }
```

```
        leaf description {
          type string;
          status deprecated;
          description
            "A textual description of component.  This node should
             contain a string that identifies the manufacturer's name
             for the component and should be set to a distinct value
             for each version or model of the component.";
          reference "RFC 6933: entPhysicalDescr";
        }

        leaf parent {
          type leafref {
            path "../../component/name";
            require-instance false;
          }
          status deprecated;
          description
            "The name of the component that physically contains this
             component.

             If this leaf is not instantiated, it indicates that this
             component is not contained in any other component.

             In the event that a physical component is contained by
             more than one physical component (e.g., double-wide
             modules), this node contains the name of one of these
             components.  An implementation MUST use the same name
             every time this node is instantiated.";
          reference "RFC 6933: entPhysicalContainedIn";
        }

        leaf parent-rel-pos {
          type int32 {
            range "0 .. 2147483647";
          }
          status deprecated;
          description
            "An indication of the relative position of this child
             component among all its sibling components.  Sibling
             components are defined as components that:

               o Share the same value of the 'parent' node; and

               o Share a common base identity for the 'class' node.

             Note that the last rule gives implementations flexibility
             in how components are numbered.  For example, some
```

```
            implementations might have a single number series for all
            components derived from 'ianahw:port', while some others
            might have different number series for different
            components with identities derived from 'ianahw:port' (for
            example, one for RJ45 and one for SFP).";

          reference "RFC 6933: entPhysicalParentRelPos";
        }

        leaf-list contains-child {
          type leafref {
            path "../../component/name";
          }
          status deprecated;
          description
            "The name of the contained component.";
          reference "RFC 6933: entPhysicalChildIndex";
        }

        leaf hardware-rev {
          type string;
          status deprecated;
          description
            "The vendor-specific hardware revision string for the
             component.  The preferred value is the hardware revision
             identifier actually printed on the component itself (if
             present).";
          reference "RFC 6933: entPhysicalHardwareRev";
        }

        leaf firmware-rev {
          type string;
          status deprecated;
          description
            "The vendor-specific firmware revision string for the
             component.";
          reference "RFC 6933: entPhysicalFirmwareRev";
        }

        leaf software-rev {
          type string;
          status deprecated;
          description
            "The vendor-specific software revision string for the
             component.";
          reference "RFC 6933: entPhysicalSoftwareRev";
        }
```

```
        leaf serial-num {
          type string;
          status deprecated;
          description
            "The vendor-specific serial number string for the
             component.  The preferred value is the serial number
             string actually printed on the component itself (if
             present).";
          reference "RFC 6933: entPhysicalSerialNum";
        }

        leaf mfg-name {
          type string;
          status deprecated;
          description
            "The name of the manufacturer of this physical component.
             The preferred value is the manufacturer name string
             actually printed on the component itself (if present).

             Note that comparisons between instances of the model-name,
             firmware-rev, software-rev, and the serial-num nodes are
             only meaningful amongst component with the same value of
             mfg-name.

             If the manufacturer name string associated with the
             physical component is unknown to the server, then this
             node is not instantiated.";
          reference "RFC 6933: entPhysicalMfgName";
        }

        leaf model-name {
          type string;
          status deprecated;
          description
            "The vendor-specific model name identifier string
             associated with this physical component.  The preferred
             value is the customer-visible part number, which may be
             printed on the component itself.

             If the model name string associated with the physical
             component is unknown to the server, then this node is not
             instantiated.";
          reference "RFC 6933: entPhysicalModelName";
        }

        leaf alias {
          type string;
          status deprecated;
```

```
        description
          "An 'alias' name for the component, as specified by a
           network manager, and provides a non-volatile 'handle' for
           the component.

           If no configured value exists, the server MAY set the
           value of this node to a locally unique value in the
           operational state.

           A server implementation MAY map this leaf to the
           entPhysicalAlias MIB object.  Such an implementation needs
           to use some mechanism to handle the differences in size
           and characters allowed between this leaf and
           entPhysicalAlias.  The definition of such a mechanism is
           outside the scope of this document.";
        reference "RFC 6933: entPhysicalAlias";
      }

      leaf asset-id {
        type string;
        status deprecated;
        description
          "This node is a user-assigned asset tracking identifier for
           the component.

           A server implementation MAY map this leaf to the
           entPhysicalAssetID MIB object.  Such an implementation
           needs to use some mechanism to handle the differences in
           size and characters allowed between this leaf and
           entPhysicalAssetID.  The definition of such a mechanism is
           outside the scope of this document.";
        reference "RFC 6933: entPhysicalAssetID";
      }

      leaf is-fru {
        type boolean;
        status deprecated;
        description
          "This node indicates whether or not this component is
           considered a 'field replaceable unit' by the vendor.  If
           this node contains the value 'true', then this component
           identifies a field replaceable unit.  For all components
           that are permanently contained within a field replaceable
           unit, the value 'false' should be returned for this
           node.";
        reference "RFC 6933: entPhysicalIsFRU";
      }
```

```
        leaf mfg-date {
          type yang:date-and-time;
          status deprecated;
          description
            "The date of manufacturing of the managed component.";
          reference "RFC 6933: entPhysicalMfgDate";
        }

        leaf-list uri {
          type inet:uri;
          status deprecated;
          description
            "This node contains identification information about the
             component.";
          reference "RFC 6933: entPhysicalUris";
        }

        leaf uuid {
          type yang:uuid;
          status deprecated;
          description
            "A Universally Unique Identifier of the component.";
          reference "RFC 6933: entPhysicalUUID";
        }

        container state {
          if-feature hardware-state;
          status deprecated;
          description
            "State-related nodes";
          reference "RFC 4268: Entity State MIB";

          leaf state-last-changed {
            type yang:date-and-time;
            status deprecated;
            description
              "The date and time when the value of any of the
               admin-state, oper-state, usage-state, alarm-state, or
               standby-state changed for this component.

               If there has been no change since the last
               re-initialization of the local system, this node
               contains the date and time of local system
               initialization.  If there has been no change since the
               component was added to the local system, this node
               contains the date and time of the insertion.";
            reference "RFC 4268: entStateLastChanged";
          }
```

```
          leaf admin-state {
            type hw:admin-state;
            status deprecated;
            description
              "The administrative state for this component.

               This node refers to a component's administrative
               permission to service both other components within its
               containment hierarchy as well other users of its
               services defined by means outside the scope of this
               module.

               Some components exhibit only a subset of the remaining
               administrative state values.  Some components cannot be
               locked, and hence this node exhibits only the 'unlocked'
               state.  Other components cannot be shutdown gracefully,
               and hence this node does not exhibit the 'shutting-down'
               state.";
            reference "RFC 4268: entStateAdmin";
          }

          leaf oper-state {
            type hw:oper-state;
            status deprecated;
            description
              "The operational state for this component.

               Note that this node does not follow the administrative
               state.  An administrative state of down does not predict
               an operational state of disabled.

               Note that some implementations may not be able to
               accurately report oper-state while the admin-state node
               has a value other than 'unlocked'.  In these cases, this
               node MUST have a value of 'unknown'.";
            reference "RFC 4268: entStateOper";
          }

          leaf usage-state {
            type hw:usage-state;
            status deprecated;
            description
              "The usage state for this component.

               This node refers to a component's ability to service
               more components in a containment hierarchy.

               Some components will exhibit only a subset of the usage
```

```
            state values.  Components that are unable to ever
            service any components within a containment hierarchy
            will always have a usage state of 'busy'.  Some
            components will only ever be able to support one
            component within its containment hierarchy and will
            therefore only exhibit values of 'idle' and 'busy'.";
          reference "RFC 4268, entStateUsage";
        }

        leaf alarm-state {
          type hw:alarm-state;
          status deprecated;
          description
            "The alarm state for this component.  It does not
             include the alarms raised on child components within its
             containment hierarchy.";
          reference "RFC 4268: entStateAlarm";
        }

        leaf standby-state {
          type hw:standby-state;
          status deprecated;
          description
            "The standby state for this component.

             Some components will exhibit only a subset of the
             remaining standby state values.  If this component
             cannot operate in a standby role, the value of this node
             will always be 'providing-service'.";
          reference "RFC 4268: entStateStandby";
        }
      }

      container sensor-data {
        when 'derived-from-or-self(../class,
                                   "ianahw:sensor")' {
          description
            "Sensor data nodes present for any component of type
             'sensor'";
        }
        if-feature hardware-sensor;
        status deprecated;

        description
          "Sensor-related nodes.";
        reference "RFC 3433: Entity Sensor MIB";

        leaf value {
```

```
        type hw:sensor-value;
        status deprecated;
        description
          "The most recent measurement obtained by the server
           for this sensor.

           A client that periodically fetches this node should also
           fetch the nodes 'value-type', 'value-scale', and
           'value-precision', since they may change when the value
           is changed.";
        reference "RFC 3433: entPhySensorValue";
      }

      leaf value-type {
        type hw:sensor-value-type;
        status deprecated;
        description
          "The type of data units associated with the
           sensor value";
        reference "RFC 3433: entPhySensorType";
      }

      leaf value-scale {
        type hw:sensor-value-scale;
        status deprecated;
        description
          "The (power of 10) scaling factor associated
           with the sensor value";
        reference "RFC 3433: entPhySensorScale";
      }

      leaf value-precision {
        type hw:sensor-value-precision;
        status deprecated;
        description
          "The number of decimal places of precision
           associated with the sensor value";
        reference "RFC 3433: entPhySensorPrecision";
      }

      leaf oper-status {
        type hw:sensor-status;
        status deprecated;
        description
          "The operational status of the sensor.";
        reference "RFC 3433: entPhySensorOperStatus";
      }
```

```
          leaf units-display {
            type string;
            status deprecated;
            description
              "A textual description of the data units that should be
               used in the display of the sensor value.";
            reference "RFC 3433: entPhySensorUnitsDisplay";
          }

          leaf value-timestamp {
            type yang:date-and-time;
            status deprecated;
            description
              "The time the status and/or value of this sensor was last
               obtained by the server.";
            reference "RFC 3433: entPhySensorValueTimeStamp";
          }

          leaf value-update-rate {
            type uint32;
            units "milliseconds";
            status deprecated;
            description
              "An indication of the frequency that the server updates
               the associated 'value' node, representing in
               milliseconds.  The value zero indicates:

                - the sensor value is updated on demand (e.g.,
                  when polled by the server for a get-request),
                - the sensor value is updated when the sensor
                  value changes (event-driven),
                - the server does not know the update rate.";
            reference "RFC 3433: entPhySensorValueUpdateRate";
          }
        }
      }
    }

    /*
     * Notifications
     */

    notification hardware-state-change {
      status deprecated;
      description
        "A hardware-state-change notification is generated when the
         value of /hardware/last-change changes in the operational
         state.";
```

```
      reference "RFC 6933, entConfigChange";
    }

    notification hardware-state-oper-enabled {
      if-feature hardware-state;
      status deprecated;
      description
        "A hardware-state-oper-enabled notification signifies that a
         component has transitioned into the 'enabled' state.";

      leaf name {
        type leafref {
          path "/hardware/component/name";
        }
        status deprecated;
        description
          "The name of the component that has transitioned into the
           'enabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/hardware/component/state/admin-state";
        }
        status deprecated;
        description
          "The administrative state for the component.";
      }
      leaf alarm-state {
        type leafref {
          path "/hardware/component/state/alarm-state";
        }
        status deprecated;
        description
          "The alarm state for the component.";
      }
      reference "RFC 4268, entStateOperEnabled";
    }

    notification hardware-state-oper-disabled {
      if-feature hardware-state;
      status deprecated;
      description
        "A hardware-state-oper-disabled notification signifies that a
         component has transitioned into the 'disabled' state.";

      leaf name {
        type leafref {
          path "/hardware/component/name";
```

```
        }
        status deprecated;
        description
          "The name of the component that has transitioned into the
           'disabled' state.";
      }
      leaf admin-state {
        type leafref {
          path "/hardware/component/state/admin-state";
        }
        status deprecated;
        description
          "The administrative state for the component.";
      }
      leaf alarm-state {
        type leafref {
          path "/hardware/component/state/alarm-state";
        }
        status deprecated;
        description
          "The alarm state for the component.";
      }
      reference "RFC 4268, entStateOperDisabled";
    }

  }

  <CODE ENDS>
```

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Jie Dong
Huawei Technologies

Email: jie.dong@huawei.com

Dan Romascanu

Email: dromasca@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2017

M. Bjorklund
Tail-f Systems
L. Lhotka
CZ.NIC

October 31, 2016

YANG Schema Mount
draft-ietf-netmod-schema-mount-03

Abstract

This document defines a mechanism to combine YANG modules into the schema defined in other 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 May 4, 2017.

Table of Contents

1. Introduction

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. 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:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration data (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Background

YANG has two mechanisms for extending a data model with additional nodes; "uses" and "augment". The "uses" statement explicitly incorporates the contents of a "grouping" defined in some other module. The "augment" statement explicitly adds contents to a target node defined in some other module. In both these cases, the source and/or target model explicitly defines the relationship between the models.

In some cases these mechanisms are not sufficient. For example, suppose we have a model like ietf-interfaces [RFC7223] that is defined to be implemented in a device. Now suppose we want to model a device that supports multiple logical devices [I-D.rtgyangdt-rtgwg-device-model], where each such logical device has its own instantiation of ietf-interfaces (and other models), but at the same time, we'd like to be able to manage all these logical devices from the main device. We would like something like this:

```
  +--rw interfaces
  | +--rw interface* [name]
  |    ...
  +--rw logical-device* [name]
     +--rw name             string
     |   ...
     +--rw interfaces
       +--rw interface* [name]
          ...
```

With the "uses" approach, ietf-interfaces would have to define a grouping with all its nodes, and the new model for logical devices would have to use this grouping. This is a not a scalable solution, since every time there is a new model defined, we would have to update our model for logical devices to use a grouping from the new model. Another problem is that this approach cannot handle vendor-specific modules.

With the "augment" approach, ietf-interfaces would have to augment the logical-device list with all its nodes, and at the same time define all its nodes on the top-level. This approach is also not scalable, since there may be other models to which we would like to add the interface list.

3. Schema Mount

The schema mount mechanism defined in this document takes a different approach to the extensibility problem described in the previous section. It decouples the definition of the relation between the source and target models from the definitions of the models themselves.

This is accomplished with a YANG extension statement that is used to specify a mount point in a data model. The purpose of a mount point is to define a place in the node hierarchy where other YANG data models may be attached, without any special notation in the other YANG data models. Only "anydata" nodes can be used as mount points.

For each mount point supported by a server, the server populates an operational state node hierarchy with information about which models it has mounted. This node hierarchy can be read by a client in order to learn what is implemented on a server.

Schema mount applies to the data model, and specifically does not assume anything about how the mounted data is implemented. It may be implemented using the same instrumentation as the rest of the system, or it may be implemented by querying some other system. Future specifications may define mechanisms to control or monitor the implementation of specific mount points.

This document allows mounting of complete data models only. Other specifications may extend this model by defining additional mechanisms, for example mounting of sub-hierarchies of a module.

3.1. Augment and Validation in Mounted Data

All paths (in leafrefs, instance-identifiers, XPath expressions, and target nodes of augments) in the data models mounted at a mount point are interpreted with the mount point as the root node, and the mounted data nodes as its children. This means that data within a mounted subtree can never refer to data outside of this subtree.

3.2. Top-level RPCs

If any mounted data model defines RPCs, these RPCs can be invoked by clients by treating them as actions defined where the mount point is specified. An example of this is given in Appendix C.1.

3.3. Top-level Notifications

If the server emits a notification defined at the top-level in any mounted data model, it is treated as if the notification was attached to the data node where the mount point is specified.

4. Data Model

This document defines the YANG 1.1 module [RFC7950] "ietf-yang-schema-mount", which has the following structure:

```
   module: ietf-yang-schema-mount
       +--ro schema-mounts
          +--ro namespace* [prefix]
          |  +--ro prefix    yang:yang-identifier
          |  +--ro ns-uri?   inet:uri
          +--ro mount-point* [module name]
          |  +--ro module        yang:yang-identifier
          |  +--ro name          yang:yang-identifier
          |  +--ro (subschema-ref)?
          |     +--:(inline)
          |     |  +--ro inline?       empty
          |     +--:(use-schema)
          |        +--ro use-schema* [name]
          |           +--ro name    -> /schema-mounts/schema/name
          |           +--ro when?   yang:xpath1.0
          +--ro schema* [name]
             +--ro name           string
             +--ro module* [name revision]
             |  +--ro name                yang:yang-identifier
             |  +--ro revision            union
             |  +--ro schema?             inet:uri
             |  +--ro namespace           inet:uri
             |  +--ro feature*            yang:yang-identifier
             |  +--ro deviation* [name revision]
             |  |  +--ro name        yang:yang-identifier
             |  |  +--ro revision    union
             |  +--ro conformance-type    enumeration
             |  +--ro submodule* [name revision]
             |     +--ro name        yang:yang-identifier
             |     +--ro revision    union
             |     +--ro schema?     inet:uri
             +--ro mount-point* [module name]
                +--ro module        yang:yang-identifier
                +--ro name          yang:yang-identifier
                +--ro (subschema-ref)?
                   +--:(inline)
                   |  +--ro inline?       empty
                   +--:(use-schema)
                      +--ro use-schema* [name]
                         +--ro name    -> /schema-mounts/schema/name
                         +--ro when?   yang:xpath1.0
```

5. Schema Mount YANG Module

This module references [RFC6991] and [RFC7895].

<CODE BEGINS> file "ietf-yang-schema-mount@2016-04-05.yang"

```
  module ietf-yang-schema-mount {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount";
    prefix yangmnt;

    import ietf-inet-types {
      prefix inet;
      reference
        "RFC 6991: Common YANG Data Types";
    }

    import ietf-yang-types {
      prefix yang;
      reference
        "RFC 6991: Common YANG Data Types";
    }

    import ietf-yang-library {
      prefix yanglib;
      reference
        "RFC 7895: YANG Module Library";
    }

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

    contact
      "WG Web:   <https://tools.ietf.org/wg/netmod/>
       WG List:  <mailto:netmod@ietf.org>

       WG Chair:  Lou Berger
                  <mailto:lberger@labn.net>

       WG Chair:  Kent Watsen
                  <mailto:kwatsen@juniper.net>

       Editor:    Martin Bjorklund
                  <mailto:mbj@tail-f.com>

       Editor:    Ladislav Lhotka
                  <mailto:lhotka@nic.cz>";

    description
      "This module defines a YANG extension statement that can be used
       to incorporate data models defined in other YANG modules in a
       module. It also defines operational state data that specify the
       overall structure of the data model.
```

```
      Copyright (c) 2016 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
      (https://trustee.ietf.org/license-info).

      The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
      NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and
      'OPTIONAL' in the module text are to be interpreted as described
      in RFC 2119 (https://tools.ietf.org/html/rfc2119).

      This version of this YANG module is part of RFC XXXX
      (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
      full legal notices.";

  revision 2016-10-26 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: YANG Schema Mount";
  }

  /*
   * Extensions
   */

  extension mount-point {
    argument name;
    description
      "The argument 'name' is a yang-identifier. The name of the
       mount point MUST be unique within the module where it is
       defined.

       The 'mount-point' statement can only be present as a
       substatement of 'anydata'.

       If a mount point is defined in a grouping, its name is bound
       to the module where the grouping is used. Note that this
       implies that such a grouping can be used at most once in a
       module.

       A mount point defines a place in the node hierarchy where
       other data models may be attached. A server that implements a
       module with a mount point, populates the
```

```
         /schema-mounts/mount-point list with detailed information on
         which data models are mounted at each mount point.";
    }

    /*
     * Groupings
     */

    grouping mount-point-list {
      description
        "This grouping is used inside the 'schema-mounts' container and
         inside the 'schema' list.";
      list mount-point {
        key "module name";
        description
          "Each entry of this list specifies a subschema for a
           particular mount point.

           Each mount point MUST be defined using the 'mount-point'
           extension in one of the modules listed in the corresponding
           YANG library instance with conformance type 'implement'. The
           corresponding YANG library instance is:

           - standard YANG library state data as defined in RFC 7895, if
             the 'mount-point' list is a child of 'schema-mounts',

           - the contents of the sibling 'yanglib:modules-state'
             container, if the 'mount-point' list is a child of
             'schema'.";
        leaf module {
          type yang:yang-identifier;
          description
            "Name of a module containing the mount point.";
        }
        leaf name {
          type yang:yang-identifier;
          description
            "Name of the mount point defined using the 'mount-point'
             extension.";
        }
        choice subschema-ref {
          description
            "Alternative way for specifying the subschema.";
          leaf inline {
            type empty;
            description
              "This leaf indicates that the server has mounted
               'ietf-yang-library' and 'ietf-schema-mount' at the mount
```

```
            point, and their instantiation (i.e., state data
            containers 'yanglib:modules-state' and 'schema-mounts')
            provides the information about the mounted schema.";
        }
        list use-schema {
          key "name";
          description
            "Each entry of this list contains a reference to a
             subschema defined in the /schema-mounts/schema list. The
             entry can be made conditional by specifying an XPath
             expression in the 'when' leaf.";
          leaf name {
            type leafref {
              path "/schema-mounts/schema/name";
            }
            description
              "Name of the referenced schema.";
          }
          leaf when {
            type yang:xpath1.0;
            description
              "This leaf contains an XPath expression. If it is
               present, then the current entry applies if and only if
               the expression evaluates to true.

               The XPath expression is evaluated once for each
               instance of the anydata node containing the mount
               point for which the 'when' leaf is defined.

               The XPath expression is evaluated using the rules
               specified in sec. 6.4 of RFC 7950, with these
               modifications:

               - The context node is the anydata instance containing
                 the corresponding 'mount-point' statement.

               - The accessible tree contains only data belonging to
                 the parent schema, i.e., all instances of anydata
                 nodes containing the mount points are considered
                 empty.

               - The set of namespace declarations is the set of all
                 prefix/namespace pairs defined in the
                 /schema-mounts/namespace list. Names without a
                 namespace prefix belong to the same namespace as the
                 context node.";
          }
        }
```

```
        }
      }
    }

    /*
     * State data nodes
     */

    container schema-mounts {
      config "false";
      description
        "Contains information about the structure of the overall data
         model implemented in the server.";
      list namespace {
        key "prefix";
        description
          "This list provides a mapping of namespace prefixes that are
           used in XPath expressions of 'when' leafs to the
           corresponding namespace URI references.";
        leaf prefix {
          type yang:yang-identifier;
          description
            "Namespace prefix.";
        }
        leaf ns-uri {
          type inet:uri;
          description
            "Namespace URI reference.";
        }
      }
      uses mount-point-list;
      list schema {
        key "name";
        description
          "Each entry specifies a schema that can be mounted at a mount
           point. The schema information consists of two parts:

           - an instance of YANG library that defines YANG modules used
             in the schema,

           - mount-point list with content identical to the top-level
             mount-point list (this makes the schema structure
             recursive).";
        leaf name {
          type string;
          description
            "Arbitrary name of the entry.";
        }
```

```
        uses yanglib:module-list;
        uses mount-point-list;
      }
    }
  }
```

<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-yang-schema-mount

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-yang-schema-mount
namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount
prefix:       yangmnt
reference:    RFC XXXX
```

7. Security Considerations

TBD

8. Contributors

The idea of having some way to combine schemas from different YANG modules into one has been proposed independently by several groups of people: Alexander Clemm, Jan Medved, and Eric Voit ([I-D.clemm-netmod-mount]); Ladislav Lhotka ([I-D.lhotka-netmod-ysdl]); and Lou Berger and Christian Hopps.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

9.2. Informative References

[I-D.clemm-netmod-mount] Clemm, A., Medved, J., and E. Voit, "Mounting YANG-Defined Information from Remote Datastores", draft-clemm-netmod-mount-05 (work in progress), September 2016.

[I-D.lhotka-netmod-ysdl] Lhotka, L., "YANG Schema Dispatching Language", draft-lhotka-netmod-ysdl-00 (work in progress), November 2015.

[I-D.rtgyangdt-rtgwg-device-model] Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Organizational Models", draft-rtgyangdt-rtgwg-device-model-05 (work in progress), August 2016.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

Appendix A. Example: Logical Devices

Logical devices within a device typically use the same set of data models in each instance. This can be modelled with a mount point:

```
module example-logical-devices {
  yang-version 1.1;
  namespace "urn:example:logical-devices";
  prefix exld;

  import ietf-yang-schema-mount {
    prefix yangmnt;
  }

  container logical-devices {
    list logical-device {
      key name;
      leaf name {
        type string;
      }

      anydata root {
        yangmnt:mount-point logical-device;
      }
    }
  }
}
```

A server with two logical devices that both implement "ietf-interfaces" [RFC7223], "ietf-ip" [RFC7277], and "ietf-system" [RFC7317] YANG modules might populate the "schema-mounts" container with:

```
   <schema-mounts
       xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount">
     <mount-point>
       <module>example-logical-devices</module>
       <name>logical-device</name>
       <use-schema>
         <name>logical-device</name>
       </use-schema>
     </mount-point>
     <schema>
       <name>logical-device</name>
       <module>
         <name>ietf-interface</name>
         <revision>2014-05-08</revision>
         <namespace>
           urn:ietf:params:xml:ns:yang:ietf-interfaces
         </namespace>
         <conformance-type>implement</conformance-type>
       </module>
       <module>
         <name>ietf-ip</name>
         <revision>2014-06-16</revision>
         <namespace>
           urn:ietf:params:xml:ns:yang:ietf-ip
         </namespace>
         <conformance-type>implement</conformance-type>
       </module>
       <module>
         <name>ietf-system</name>
         <revision>2014-08-06</revision>
         <namespace>
           urn:ietf:params:xml:ns:yang:ietf-system
         </namespace>
         <conformance-type>implement</conformance-type>
       </module>
       <module>
         <name>ietf-yang-types</name>
         <revision>2013-07-15</revision>
         <namespace>
           urn:ietf:params:xml:ns:yang:ietf-yang-types
         </namespace>
         <conformance-type>import</conformance-type>
       </module>
     </schema>
   </schema-mounts>
```

and the "logical-devices" container might have:

```
<logical-devices xmlns="urn:example:logical-devices">
  <logical-device>
    <name>vrtrA</name>
    <root>
      <interfaces
          xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>eth0</name>
            <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
              <enabled>true</enabled>
              ...
            </ipv6>
          ...
        </interface>
      </interfaces>
      <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
        ...
      </system>
    </root>
  </logical-device>
  <logical-device>
    <name>vrtrB</name>
    <root>
      <interfaces
          xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>eth0</name>
            <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
              <enabled>true</enabled>
              ...
            </ipv6>
          ...
        </interface>
      </interfaces>
      <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
        ...
      </system>
    </root>
  </logical-device>
</logical-devices>
```

Appendix B. Example: Network Manager with Fixed Device Models

This example shows how a Network Manager application can use schema mount to define a data model for a network consisting of devices whose data models are known a priori and fixed.

Assume for simplicity that only two device types are used (switch and router), and they are identified by identities defined in the module "example-device-types":

```
module example-device-types {
  namespace "http://example.org/device-types";
  prefix edt;
  identity device-type;
  identity switch-device {
    base device-type;
  }
  identity router-device {
    base device-type;
  }
}
```

Schema mount is used to mount the device data models conditionally, depending on the "type" leaf that is a sibling of the mount point. This approach is similar to "ietf-interfaces" [RFC7223] where the same effect is achieved via conditional augments.

The top-level module may look as follows:

```
module example-network-manager-fixed {
  yang-version 1.1;
  namespace "urn:example:network-manager-fixed";
  prefix exf;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-schema-mount {
    prefix yangmnt;
  }
  import example-device-types {
    prefix edt;
  }

  container managed-devices {
    description
      "The managed devices and device communication settings.";

    list device {
      key name;
      leaf name {
        type string;
      }
      leaf type {
```

```
          type identityref {
            base edt:device-type;
          }
        }
        container transport {
          choice protocol {
            mandatory true;
            container netconf {
              leaf address {
                type inet:ip-address;
                mandatory true;
              }
              container authentication {
                // ...
              }
            }
            container restconf {
              leaf address {
                type inet:ip-address;
                mandatory true;
              }
              // ...
            }
          }
        }
        anydata root {
          yangmnt:mount-point managed-device;
        }
      }
    }
  }
```

The "schema-mounts" container may have the following data:

```
<data-model
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount">
  <namespace>
    <prefix>edt</prefix>
    <ns-uri>http://example.org/device-types</ns-uri>
  </namespace>
  <mount-point>
    <module>example-network-manager</module>
    <name>managed-device</name>
    <use-schema>
      <name>switch</name>
      <when>derived-from-or-self(../type, 'edt:switch-device')</when>
    </use-schema>
    <use-schema>
      <name>router</name>
      <when>derived-from-or-self(../type, 'edt:router-device')</when>
    </use-schema>
  </mount-point>
  <schema>
    <name>switch</name>
    <module>
      ...
    </module>
    ...
  </schema>
  <schema>
    <name>router</name>
    <module>
      ...
    </module>
    ...
  </schema>
</data-model>
```

The "devices" list may contain any number of instances of either type.

Appendix C. Example: Network Manager with Arbitrary Device Models

This example shows how a Network Manager application can use schema mount to define a data model for a network consisting of devices whose data models are not known in advance -- each device is expected to provide its data model dynamically.

Schema mount is used to mount the data models that each device supports, and these data models can be discovered by inspecting state data under the corresponding mount point. Every such device must

therefore implement "ietf-yang-library" and optionally "ietf-schema-mount".

```
   module example-network-manager-arbitrary {
     yang-version 1.1;
     namespace "urn:example:network-manager-arbitrary";
     prefix exa;

     import ietf-inet-types {
       prefix inet;
     }
     import ietf-yang-schema-mount {
       prefix yangmnt;
     }

     container managed-devices {
       description
         "The managed devices and device communication settings.";

       list device {
         key name;
         leaf name {
           type string;
         }
         container transport {
           choice protocol {
             mandatory true;
             container netconf {
               leaf address {
                 type inet:ip-address;
                 mandatory true;
               }
               container authentication {
                 // ...
               }
             }
             container restconf {
               leaf address {
                 type inet:ip-address;
                 mandatory true;
               }
               // ...
             }
           }
         }
         anydata root {
           yangmnt:mount-point managed-device;
         }
       }
     }
   }
```

The "schema-mounts" container may have the following data:

```
<data-model
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount">
  <mount-point>
    <module>example-network-manager</module>
    <name>managed-device</name>
    <inline/>
  </mount-point>
</data-model>
```

The "devices" container might have:

```
<devices xmlns="urn:example:network-manager">
  <device>
    <name>rtrA</name>
    <transport>
      <netconf>
        <address>2001:db8::2</address>
        <authentication>
          ...
        </authentication>
        ...
      </netconf>
    </transport>
    <root>
      <modules-state
          xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
        <module>
          <name>ietf-system</name>
          ...
        </module>
      </modules-state>
      <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
        ...
      </system>
    </root>
  </device>
  <device>
    <name>rtrB</name>
    <transport>
      <restconf>
        <address>2001:db8::3</address>
        <authentication>
          ...
        </authentication>
        ...
      </restconf>
```

```
      </transport>
      <root>
        <modules-state
            xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
          <module>
            <name>ietf-interfaces</name>
            ...
          </module>
        </modules-state>
        <interfaces
            xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
          ...
        </interfaces>
      </root>
    </device>
  </devices>
```

C.1. Invoking an RPC

A client that wants to invoke the "restart" operation [RFC7317] on the managed device "rtrA" over NETCONF [RFC6241] can send:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <managed-devices xmlns="urn:example:network-manager">
      <device>
        <name>rtrA</name>
        <root>
          <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
            <restart/>
          </system>
        </root>
      </device>
    </managed-devices>
  </action>
</rpc>
```

Appendix D. Open Issues

- Is the 'mount-point' extension really needed? Now that mount points can only appear under anydata nodes, there seems to be little need to otherwise restrict mount point locations. In the 'mount-point' list, schema node identifiers (as in 'augment' statements) can be used instead of the (module, name) pair for identifying mount points. As a useful side effect, a grouping containing mount points could be used any number of times in the same module. OTOH, by using this extension, the intention of the

data modeller is clear, and it provides a formal machine readable instruction about where mounts are allowed to occur.

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Ladislav Lhotka
CZ.NIC

Email: mbj@lhotka@nic.cz

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2019

M. Bjorklund
Tail-f Systems
L. Lhotka
CZ.NIC

October 17, 2018

YANG Schema Mount
draft-ietf-netmod-schema-mount-12

Abstract

This document defines a mechanism to add the schema trees defined by a set of YANG modules onto a mount point defined in the schema tree in some 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 April 20, 2019.

Copyright Notice

Table of Contents

1. Introduction

Modularity and extensibility were among the leading design principles of the YANG data modeling language. As a result, the same YANG module can be combined with various sets of other modules and thus form a data model that is tailored to meet the requirements of a specific use case. Server implementors are only required to specify all YANG modules comprising the data model (together with their revisions and other optional choices) in the YANG library data ([RFC7895], [I-D.ietf-netconf-rfc7895bis] and Section 5.6.4 of [RFC7950]) implemented by the server. Such YANG modules appear in the data model "side by side", i.e., top-level data nodes of each module - if there are any - are also top-level nodes of the overall data model.

YANG has two mechanisms for contributing a schema hierarchy defined elsewhere to the contents of an internal node of the schema tree; these mechanisms are realized through the following YANG statements:

- The "uses" statement explicitly incorporates the contents of a grouping defined in the same or another module. See Section 4.2.6 of [RFC7950] for more details.

- The "augment" statement explicitly adds contents to a target node defined in the same or another module. See Section 4.2.8 of [RFC7950] for more details.

With both mechanisms, the YANG module with the "uses" or "augment" statement explicitly defines the exact location in the schema tree where the new nodes are placed.

In some cases these mechanisms are not sufficient; it is sometimes necessary that an existing module (or a set of modules) is added to the data model starting at locations other than the root. For example, YANG modules such as "ietf-interfaces" [RFC8343] are defined so as to be used in a data model of a physical device. Now suppose we want to model a device that supports multiple logical devices [I-D.ietf-rtgwg-lne-model], each of which has its own instantiation of "ietf-interfaces", and possibly other modules, but, at the same time, we want to be able to manage all these logical devices from the master device. Hence, we would like to have a schema tree like this:

```
  +--rw interfaces
  |  +--rw interface* [name]
  |     ...
  +--rw logical-network-element* [name]
     +--rw name
     |   ...
     +--rw interfaces
       +--rw interface* [name]
          ...
```

With the "uses" approach, the complete schema tree of "ietf-interfaces" would have to be wrapped in a grouping, and then this grouping would have to be used at the top level (for the master device) and then also in the "logical-network-element" list (for the logical devices). This approach has several disadvantages:

- It is not scalable because every time there is a new YANG module that needs to be added to the logical device model, we have to update the model for logical devices with another "uses" statement pulling in contents of the new module.

- Absolute references to nodes defined inside a grouping may break if the grouping is used in different locations.

- Nodes defined inside a grouping belong to the namespace of the module where it is used, which makes references to such nodes from other modules difficult or even impossible.

- It would be difficult for vendors to add proprietary modules when the "uses" statements are defined in a standard module.

With the "augment" approach, "ietf-interfaces" would have to augment the "logical-network-element" list with all its nodes, and at the same time define all its nodes at the top level. The same hierarchy of nodes would thus have to be defined twice, which is clearly not scalable either.

This document introduces a new mechanism, denoted as schema mount, that allows for mounting one data model consisting of any number of YANG modules at a specified location of another (parent) schema. Unlike the "uses" and "augment" approaches discussed above, the mounted modules needn't be specially prepared for mounting and, consequently, existing modules such as "ietf-interfaces" can be mounted without any modifications.

The basic idea of schema mount is to label a data node in the parent schema as the mount point, and then define a complete data model to be attached to the mount point so that the labeled data node effectively becomes the root node of the mounted data model.

In principle, the mounted schema can be specified at three different phases of the data model life cycle:

1. Design-time: the mounted schema is defined along with the mount point in the parent YANG module. In this case, the mounted schema has to be the same for every implementation of the parent module.

2. Implementation-time: the mounted schema is defined by a server implementor and is as stable as the YANG library information of the server.

3. Run-time: the mounted schema is defined by instance data that is part of the mounted data model. If there are multiple instances of the same mount point (e.g., in multiple entries of a list), the mounted data model may be different for each instance.

The schema mount mechanism defined in this document provides support only for the latter two cases. Design-time mounts are outside the

scope of this document, and could be possibly dealt with in a future revision of the YANG data modeling language.

Schema mount applies to the data model, and specifically does not assume anything about the source of instance data for the mounted schemas. It may be implemented using the same instrumentation as the rest of the system, or it may be implemented by querying some other system. Future specifications may define mechanisms to control or monitor the implementation of specific mount points.

How and when specific mount points are instantiated by the server is out of scope for this document. Such mechanisms may be defined in future specifications.

This document allows mounting of complete data models only. Other specifications may extend this model by defining additional mechanisms such as mounting sub-hierarchies of a module.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].

2. Terminology and Notation

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC7950] and are not redefined here:

- action
- container
- data node
- list
- RPC operation
- schema node
- schema tree

The following terms are defined in [RFC8342] and are not redefined here:

- client
- notification
- operational state
- server

The following term is defined in [RFC8343] and is not redefined here:

- system-controlled interface

The following term is defined in [I-D.ietf-netconf-rfc7895bis] is not redefined here:

- YANG library content identifier

The following additional terms are used within this document:

- mount point: A container or a list node whose definition contains the "mount-point" extension statement. The argument of the "mount-point" statement defines a label for the mount point.
- schema: A collection of schema trees with a common root.
- top-level schema: A schema rooted at the root node.
- mounted schema: A schema rooted at a mount point.
- parent schema (of a mounted schema): A schema containing the mount point.
- schema mount: The mechanism to combine data models defined in this document.

2.1. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340]

2.2. Namespace Prefixes

In this document, names of data nodes, YANG extensions, actions and other data model objects are often used 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
yangmnt	ietf-yang-schema-mount	Section 9
inet	ietf-inet-types	[RFC6991]
yang	ietf-yang-types	[RFC6991]
yanglib	ietf-yang-library	[RFC7895], [I-D.ietf-netconf-rfc7895bis]

Table 1: Namespace Prefixes

3. Schema Mount

The schema mount mechanism defined in this document provides a new extensibility mechanism for use with YANG 1.1. In contrast to the existing mechanisms described in Section 1, schema mount defines the relationship between the source and target YANG modules outside these modules. The procedure consists of two separate steps that are described in the following subsections.

3.1. Mount Point Definition

A "container" or "list" node becomes a mount point if the "mount-point" extension (defined in the "ietf-yang-schema-mount" module) is used in its definition. This extension can appear only as a substatement of "container" and "list" statements.

The argument of the "mount-point" extension is a YANG identifier that defines a label for the mount point. A module MAY contain multiple "mount-point" statements having the same argument.

It is therefore up to the designer of the parent schema to decide about the placement of mount points. A mount point can also be made conditional by placing "if-feature" and/or "when" as substatements of the "container" or "list" statement that represents the mount point.

The "mount-point" statement MUST NOT be used in a YANG version 1 module [RFC6020]. The reason for this is that otherwise it is not possible to invoke mounted RPC operations, and receive mounted notifications. See Section 5 for details. Note, however, that modules written in any YANG version, including version 1, can be mounted under a mount point.

Note that the "mount-point" statement does not define a new data node.

3.2. Data Model

This document defines the YANG 1.1 module [RFC7950] "ietf-yang-schema-mount", which has the following structure:

```
module: ietf-yang-schema-mount
  +--ro schema-mounts
     +--ro namespace* [prefix]
     |  +--ro prefix    yang:yang-identifier
     |  +--ro uri?      inet:uri
     +--ro mount-point* [module label]
        +--ro module                yang:yang-identifier
        +--ro label                 yang:yang-identifier
        +--ro config?               boolean
        +--ro (schema-ref)
           +--:(inline)
           |  +--ro inline!
           +--:(shared-schema)
              +--ro shared-schema!
                 +--ro parent-reference*   yang:xpath1.0
```

3.3. Specification of the Mounted Schema

Mounted schemas for all mount points in the parent schema are determined from state data in the "/schema-mounts" container.

Generally, the modules that are mounted under a mount point have no relation to the modules in the parent schema; specifically, if a module is mounted it may or may not be present in the parent schema and, if present, its data will generally have no relationship to the data of the parent. Exceptions are possible and such needs to be defined in the model defining the exception. For example, [I-D.ietf-rtgwg-lne-model] defines a mechanism to bind interfaces to mounted logical network elements.

The "/schema-mounts" container has the "mount-point" list as one of its children. Every entry of this list refers through its key to a mount point and specifies the mounted schema.

If a mount point is defined in the parent schema but does not have an entry in the "mount-point" list, then the mounted schema is void, i.e., instances of that mount point MUST NOT contain any data except those that are defined in the parent schema.

If multiple mount points with the same name are defined in the same module - either directly or because the mount point is defined in a grouping and the grouping is used multiple times - then the

corresponding "mount-point" entry applies equally to all such mount points.

The "config" property of mounted schema nodes is overridden and all nodes in the mounted schema are read-only ("config false") if at least one of the following conditions is satisfied for a mount point:

- the mount point is itself defined as "config false"
- the "config" leaf in the corresponding entry of the "mount-point" list is set to "false".

An entry of the "mount-point" list can specify the mounted schema in two different ways, "inline" or "shared-schema".

The mounted schema is determined at run time: every instance of the mount point that exists in the operational state MUST contain a copy of YANG library data that defines the mounted schema exactly as for a top-level schema. A client is expected to retrieve this data from the instance tree. In the "inline" case, instances of the same mount point MAY use different mounted schemas, whereas in the "shared-schema" case, all instances MUST use the same mounted schema. This means that in the "shared-schema" case, all instances of the same mount point MUST have the same YANG library content identifier. In the "inline" case, if two instances have the same YANG library content identifier it is not guaranteed that the YANG library contents are equal for these instances.

Examples of "inline" and "shared-schema" can be found in Appendix A.2 and Appendix A.3, respectively.

3.4. Multiple Levels of Schema Mount

YANG modules in a mounted schema MAY again contain mount points under which other schemas can be mounted. Consequently, it is possible to construct data models with an arbitrary number of mounted schemas. A schema for a mount point contained in a mounted module can be specified by implementing "ietf-yang-library" and "ietf-yang-schema-mount" modules in the mounted schema, and specifying the schemas exactly as it is done in the top-level schema.

4. Referring to Data Nodes in the Parent Schema

A fundamental design principle of schema mount is that the mounted schema works exactly as a top-level schema, i.e., it is confined to the "mount jail". This means that all paths in the mounted schema (in leafrefs, instance-identifiers, XPath [XPATH] expressions, and target nodes of augments) are interpreted with the mount point as the

root node. YANG modules of the mounted schema as well as corresponding instance data thus cannot refer to schema nodes or instance data outside the mount jail.

However, this restriction is sometimes too severe. A typical example is network instances (NI) [I-D.ietf-rtgwg-ni-model], where each NI has its own routing engine but the list of interfaces is global and shared by all NIs. If we want to model this organization with the NI schema mounted using schema mount, the overall schema tree would look schematically as follows:

```
  +--rw interfaces
  |  +--rw interface* [name]
  |     ...
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name
        +--rw root
           +--rw routing
              ...
```

Here, the "root" node is the mount point for the NI schema. Routing configuration inside an NI often needs to refer to interfaces (at least those that are assigned to the NI), which is impossible unless such a reference can point to a node in the parent schema (interface name).

Therefore, schema mount also allows for such references. For every mount point in the "shared-schema" case, it is possible to specify a leaf-list named "parent-reference" that contains zero or more XPath 1.0 expressions. Each expression is evaluated with the node in the parent data tree where the mount point is defined as the context node. The result of this evaluation MUST be a nodeset (see the description of the "parent-reference" node for a complete definition of the evaluation context). For the purposes of evaluating XPath expressions within the mounted data tree, the union of all such nodesets is added to the accessible data tree.

It is worth emphasizing that the nodes specified in "parent-reference" leaf-list are available in the mounted schema only for XPath evaluations. In particular, they cannot be accessed there via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040].

5. RPC operations and Notifications

If a mounted YANG module defines an RPC operation, clients can invoke this operation as if it were defined as an action for the corresponding mount point, see Section 7.15 of [RFC7950]. An example of this is given in Appendix A.4.

Similarly, if the server emits a notification defined at the top level of any mounted module, it MUST be represented as if the notification was connected to the mount point, see Section 7.16 of [RFC7950].

Note, inline actions and notifications will not work when they are contained within a list node without a "key" statement (see section 7.15 and 7.16 of [RFC7950]). Therefore, to be useful, mount points that contain modules with RPCs, actions, and notifications SHOULD NOT have any ancestor node that is a list node without a "key" statement. This requirement applies to the definition of modules using the "mount-point" extension statement.

6. Network Management Datastore Architecture (NMDA) Considerations

The schema mount solution presented in this document is designed to work both with servers that implement the NMDA [RFC8342], and old servers that don’t implement the NMDA.

Note to RFC Editor: please update the date YYYY-MM-DD below with the revision of the ietf-yang-library in the published version of draft-ietf-netconf-rfc7895bis, and remove this note.

Specifically, a server that doesn’t support the NMDA, MAY implement revision 2016-06-21 of "ietf-yang-library" [RFC7895] under a mount point. A server that supports the NMDA, MUST implement at least revision YYYY-MM-DD of "ietf-yang-library" [I-D.ietf-netconf-rfc7895bis] under the mount points.

7. Interaction with the Network Configuration Access Control Model (NACM)

If NACM [RFC8341] is implemented on a server, it is used to control access to nodes defined by the mounted schema in the same way as for nodes defined by the top-level schema.

For example, suppose the module "ietf-interfaces" is mounted in the "root" container in the "logical-network-element" list defined in [I-D.ietf-rtgwg-lne-model]. Then the following NACM path can be used to control access to the "interfaces" container (where the character

'\' is used where a line break has been inserted for formatting reasons):

```
  <path xmlns:lne=
          "urn:ietf:params:xml:ns:yang:ietf-logical-network-element"
        xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    /lne:logical-network-elements\
      /lne:logical-network-element/lne:root/if:interfaces
  </path>
```

8. Implementation Notes

Network management of devices that use a data model with schema mount can be implemented in different ways. However, the following implementations options are envisioned as typical:

- shared management: instance data of both parent and mounted schemas are accessible within the same management session.

- split management: one (master) management session has access to instance data of both parent and mounted schemas but, in addition, an extra session exists for every instance of the mount point, having access only to the mounted data tree.

9. Schema Mount YANG Module

This module references [RFC6991].

<CODE BEGINS> file "ietf-yang-schema-mount@2018-10-16"

```
module ietf-yang-schema-mount {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount";
  prefix yangmnt;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
```

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

    contact
      "WG Web:   <https://tools.ietf.org/wg/netmod/>
       WG List:  <mailto:netmod@ietf.org>

       Editor:   Martin Bjorklund
                 <mailto:mbj@tail-f.com>

       Editor:   Ladislav Lhotka
                 <mailto:lhotka@nic.cz>";

    // RFC Ed.: replace XXXX with actual RFC number and
    // remove this note.
    description
      "This module defines a YANG extension statement that can be used
       to incorporate data models defined in other YANG modules in a
       module. It also defines operational state data that specify the
       overall structure of the data model.

       Copyright (c) 2018 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
       (https://trustee.ietf.org/license-info).

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
       NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
       'MAY', and 'OPTIONAL' in the module text are to be interpreted
       as described in BCP 14 [RFC 2119] [RFC8174] when, and only when,
       they appear in all capitals, as shown here.

       This version of this YANG module is part of RFC XXXX
       (https://tools.ietf.org/html/rfcXXXX); see the RFC itself for
       full legal notices.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    revision 2018-10-16 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: YANG Schema Mount";
    }
```

```
     /*
      * Extensions
      */

     extension mount-point {
       argument label;
       description
         "The argument 'label' is a YANG identifier, i.e., it is of the
          type 'yang:yang-identifier'.

          The 'mount-point' statement MUST NOT be used in a YANG
          version 1 module, neither explicitly nor via a 'uses'
          statement.

          The 'mount-point' statement MAY be present as a substatement
          of 'container' and 'list', and MUST NOT be present elsewhere.
          There MUST NOT be more than one 'mount-point' statement in a
          given 'container' or 'list' statement.

          If a mount point is defined within a grouping, its label is
          bound to the module where the grouping is used.

          A mount point defines a place in the node hierarchy where
          other data models may be attached. A server that implements a
          module with a mount point populates the
          /schema-mounts/mount-point list with detailed information on
          which data models are mounted at each mount point.

          Note that the 'mount-point' statement does not define a new
          data node.";
     }

     /*
      * State data nodes
      */

     container schema-mounts {
       config false;
       description
         "Contains information about the structure of the overall
          mounted data model implemented in the server.";
       list namespace {
         key "prefix";
         description
           "This list provides a mapping of namespace prefixes that are
            used in XPath expressions of 'parent-reference' leafs to the
            corresponding namespace URI references.";
         leaf prefix {
```

```
          type yang:yang-identifier;
          description
            "Namespace prefix.";
        }
        leaf uri {
          type inet:uri;
          description
            "Namespace URI reference.";
        }
      }
      list mount-point {
        key "module label";
        description
          "Each entry of this list specifies a schema for a particular
           mount point.

           Each mount point MUST be defined using the 'mount-point'
           extension in one of the modules listed in the server's
           YANG library instance with conformance type 'implement'.";
        leaf module {
          type yang:yang-identifier;
          description
            "Name of a module containing the mount point.";
        }
        leaf label {
          type yang:yang-identifier;
          description
            "Label of the mount point defined using the 'mount-point'
             extension.";
        }
        leaf config {
          type boolean;
          default "true";
          description
            "If this leaf is set to 'false', then all data nodes in the
             mounted schema are read-only (config false), regardless of
             their 'config' property.";
        }
        choice schema-ref {
          mandatory true;
          description
            "Alternatives for specifying the schema.";
          container inline {
            presence
              "A complete self-contained schema is mounted at the
               mount point.";
            description
              "This node indicates that the server has mounted at least
```

```
            the module 'ietf-yang-library' at the mount point, and
            its instantiation provides the information about the
            mounted schema.

            Different instances of the mount point may have
            different schemas mounted.";
        }
        container shared-schema {
          presence
            "The mounted schema together with the 'parent-reference'
             make up the schema for this mount point.";
          description
            "This node indicates that the server has mounted at least
             the module 'ietf-yang-library' at the mount point, and
             its instantiation provides the information about the
             mounted schema.  When XPath expressions in the mounted
             schema are evaluated, the 'parent-reference' leaf-list
             is taken into account.

             Different instances of the mount point MUST have the
             same schema mounted.";
          leaf-list parent-reference {
            type yang:xpath1.0;
            description
              "Entries of this leaf-list are XPath 1.0 expressions
               that are evaluated in the following context:

                - The context node is the node in the parent data tree
                  where the mount-point is defined.

                - The accessible tree is the parent data tree
                  *without* any nodes defined in modules that are
                  mounted inside the parent schema.

                - The context position and context size are both equal
                  to 1.

                - The set of variable bindings is empty.

                - The function library is the core function library
                  defined in [XPath] and the functions defined in
                  Section 10 of [RFC7950].

                - The set of namespace declarations is defined by the
                  'namespace' list under 'schema-mounts'.

               Each XPath expression MUST evaluate to a nodeset
               (possibly empty). For the purposes of evaluating XPath
```

```
                    expressions whose context nodes are defined in the
                    mounted schema, the union of all these nodesets
                    together with ancestor nodes are added to the
                    accessible data tree.

                    Note that in the case 'ietf-yang-schema-mount' is
                    itself mounted, a 'parent-reference' in the mounted
                    module may refer to nodes that were brought into the
                    accessible tree through a 'parent-reference' in the
                    parent schema.";
              }
            }
          }
        }
      }
    }
```

<CODE ENDS>

10. 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-yang-schema-mount

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-yang-schema-mount
  namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount
  prefix:       yangmnt
  reference:    RFC XXXX
```

11. Security Considerations

YANG module "ietf-yang-schema-mount" specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The network configuration access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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:

- /schema-mounts: The schema defined by this state data provides detailed information about a server implementation may help an attacker identify the server capabilities and server implementations with known bugs. Server vulnerabilities may be specific to particular modules included in the schema, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the schema information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

It is important to take the security considerations for all nodes in the mounted schemas into account, and control access to these nodes by using the mechanism described in Section 7.

Care must be taken when the "parent-reference" XPath expressions are constructed, since the result of the evaluation of these expressions is added to the accessible tree for any XPath expression found in the mounted schema.

12. Contributors

The idea of having some way to combine schemas from different YANG modules into one has been proposed independently by several groups of people: Alexander Clemm, Jan Medved, and Eric Voit ([I-D.clemm-netmod-mount]); and Lou Berger and Christian Hopps:

- Lou Berger, LabN Consulting, L.L.C., <lberger@labn.net>
- Alexander Clemm, Huawei, <alexander.clemm@huawei.com>
- Christian Hopps, Deutsche Telekom, <chopps@chopps.org>
- Jan Medved, Cisco, <jmedved@cisco.com>
- Eric Voit, Cisco, <evoit@cisco.com>

13. References

13.1. Normative References

[I-D.ietf-netconf-rfc7895bis]
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-06 (work in progress), April 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[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>.

13.2. Informative References

[I-D.clemm-netmod-mount] Clemm, A., Voit, E., and J. Medved, "Mounting YANG-Defined Information from Remote Datastores", draft-clemm-netmod-mount-06 (work in progress), March 2017.

[I-D.ietf-isis-yang-isis-cfg] Litkowski, S., Yeung, D., Lindem, A., Zhang, Z., and L. Lhotka, "YANG Data Model for IS-IS protocol", draft-ietf-isis-yang-isis-cfg-24 (work in progress), August 2018.

[I-D.ietf-rtgwg-device-model] Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

[I-D.ietf-rtgwg-lne-model] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", draft-ietf-rtgwg-lne-model-10 (work in progress), March 2018.

[I-D.ietf-rtgwg-ni-model] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Network Instances", draft-ietf-rtgwg-ni-model-12 (work in progress), March 2018.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

Appendix A. Example: Device Model with LNEs and NIs

This non-normative example demonstrates an implementation of the device model as specified in Section 2 of [I-D.ietf-rtgwg-device-model], using both logical network elements (LNE) and network instances (NI).

In these examples, the character '\' is used where a line break has been inserted for formatting reasons.

A.1. Physical Device

The data model for the physical device may be described by this YANG library content, assuming the server supports the NMDA:

```
{
   "ietf-yang-library:yang-library": {
     "content-id": "14e2ab5dc325f6d86f743e8d3ade233f1a61a899",
     "module-set": [
       {
         "name": "physical-device-modules",
         "module": [
           {
             "name": "ietf-datastores",
             "revision": "2018-02-14",
             "namespace":
               "urn:ietf:params:xml:ns:yang:ietf-datastores"
           },
           {
             "name": "iana-if-type",
             "revision": "2015-06-12",
             "namespace": "urn:ietf:params:xml:ns:yang:iana-if-type"
           },
           {
             "name": "ietf-interfaces",
             "revision": "2018-02-20",
```

```
                "feature": ["arbitrary-names", "pre-provisioning" ],
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-interfaces"
              },
              {
                "name": "ietf-ip",
                "revision": "2018-02-22",
                "namespace": "urn:ietf:params:xml:ns:yang:ietf-ip"
              },
              {
                "name": "ietf-logical-network-element",
                "revision": "2016-10-21",
                "feature": [ "bind-lne-name" ],
                "namespace":
                  "urn:ietf:params:xml:ns:yang:\
                  ietf-logical-network-element"
              },
              {
                "name": "ietf-yang-library",
                "revision": "2018-02-21",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-library"
              },
              {
                "name": "ietf-yang-schema-mount",
                "revision": "2018-03-20",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount"
              }
            ],
            "import-only-module": [
              {
                "name": "ietf-inet-types",
                "revision": "2013-07-15",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-inet-types"
              },
              {
                "name": "ietf-yang-types",
                "revision": "2013-07-15",
                "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-types"
              }
            ]
          }
        ],
        "schema": [
          {
```

```
            "name": "physical-device-schema",
            "module-set": [ "physical-device-modules" ]
          }
        ],
        "datastore": [
          {
            "name": "ietf-datastores:running",
            "schema": "physical-device-schema"
          },
          {
            "name": "ietf-datastores:operational",
            "schema": "physical-device-schema"
          }
        ]
      }
    }
```

A.2. Logical Network Elements

Each LNE can have a specific data model that is determined at run time, so it is appropriate to mount it using the "inline" method, hence the following "schema-mounts" data:

```
{
  "ietf-yang-schema-mount:schema-mounts": {
    "mount-point": [
      {
        "module": "ietf-logical-network-element",
        "label": "root",
        "inline": {}
      }
    ]
  }
}
```

An administrator of the host device has to configure an entry for each LNE instance, for example,

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "enabled": true,
        "ietf-logical-network-element:bind-lne-name": "eth0"
      }
    ]
  },
  "ietf-logical-network-element:logical-network-elements": {
    "logical-network-element": [
      {
        "name": "lne-1",
        "managed": true,
        "description": "LNE with NIs",
        "root": {
          ...
        }
      }
      ...
    ]
  }
}
```

and then also place necessary state data as the contents of the "root" instance, which should include at least

- YANG library data specifying the LNE's data model, for example, assuming the server does not implement the NMDA:

```
{
  "ietf-yang-library:modules-state": {
    "module-set-id": "9358e11874068c8be06562089e94a89e0a392019",
    "module": [
      {
        "name": "iana-if-type",
        "revision": "2014-05-08",
        "namespace": "urn:ietf:params:xml:ns:yang:iana-if-type",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-inet-types",
        "revision": "2013-07-15",
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types",
        "conformance-type": "import"
      },
```

```
        {
          "name": "ietf-interfaces",
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "feature": [
            "ipv6-privacy-autoconf"
          ],
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-network-instance",
          "revision": "2016-10-27",
          "feature": [
            "bind-network-instance-name"
          ],
          "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-network-instance",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
```

```
      ]
    }
  }
```

o state data for interfaces assigned to the LNE instance (that effectively become system-controlled interfaces for the LNE), for example:

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "iana-if-type:ethernetCsmacd",
        "oper-status": "up",
        "statistics": {
          "discontinuity-time": "2016-12-16T17:11:27+02:00"
        },
        "ietf-ip:ipv6": {
          "address": [
            {
              "ip": "fe80::42a8:f0ff:fea8:24fe",
              "origin": "link-layer",
              "prefix-length": 64
            }
          ]
        }
      }
    ]
  }
}
```

A.3. Network Instances

Assuming that network instances share the same data model, it can be mounted using the "shared-schema" method as follows:

```
{
  "ietf-yang-schema-mount:schema-mounts": {
    "namespace": [
      {
          "prefix": "if",
          "uri": "urn:ietf:params:xml:ns:yang:ietf-interfaces"
      },
      {
          "prefix": "ni",
          "uri": "urn:ietf:params:xml:ns:yang:ietf-network-instance"
      }
    ],
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "root",
          "shared-schema": {
            "parent-reference": [
              "/if:interfaces/if:interface[\
              ni:bind-network-instance-name = current()/../ni:name]"
            ]
          }
      }
    ]
  }
}
```

Note also that the "ietf-interfaces" module appears in the "parent-reference" leaf-list for the mounted NI schema. This means that references to LNE interfaces, such as "outgoing-interface" in static routes, are valid despite the fact that "ietf-interfaces" isn’t part of the NI schema.

A.4. Invoking an RPC Operation

Assume that the mounted NI data model also implements the "ietf-isis" module [I-D.ietf-isis-yang-isis-cfg]. An RPC operation defined in this module, such as "clear-adjacency", can be invoked by a client session of a LNE’s RESTCONF server as an action tied to a the mount point of a particular network instance using a request URI like this (all on one line):

```
POST /restconf/data/ietf-network-instance:network-instances/
    network-instance=rtrA/root/ietf-isis:clear-adjacency HTTP/1.1
```

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

NETMOD WG
Internet-Draft
Intended status: Standards Track
Expires: May 17, 2017

C. Wildes, Ed.
K. Koushik, Ed.
Cisco Systems Inc.

November 13, 2016

A YANG Data Model for Syslog Configuration
draft-ietf-netmod-syslog-model-11

Abstract

This document describes a data model for the configuration of syslog.

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 May 17, 2017.

Table of Contents

1. Introduction

Operating systems, processes and applications generate messages indicating their own status or the occurrence of events. These messages are useful for managing and/or debugging the network and its services. The BSD syslog protocol is a widely adopted protocol that is used for transmission and processing of the messages.

Since each process, application and operating system was written somewhat independently, there is little uniformity to the content of syslog messages. For this reason, no assumption is made upon the formatting or contents of the messages. The protocol is simply designed to transport these event messages. No acknowledgement of the receipt is made.

Essentially, a syslog process receives messages (from the kernel, processes, applications or other syslog processes) and processes those. The processing involves logging to a local file, displaying on console, user terminal, and/or relaying to syslog processes on other machines. The processing is determined by the "facility" that originated the message and the "severity" assigned to the message by the facility.

We are using definitions of syslog protocol from [RFC5424] in this RFC.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

1.2. Terminology

The term "originator" is defined in [RFC5424]: an "originator" generates syslog content to be carried in a message.

The terms "relay" and "collectors" are as defined in [RFC5424].

2. Problem Statement

This document defines a YANG [RFC6020] configuration data model that may be used to configure the syslog feature running on a system. YANG models can be used with network management protocols such as NETCONF [RFC6241] to install, manipulate, and delete the configuration of network devices.

The data model makes use of the YANG "feature" construct which allows implementations to support only those syslog features that lie within their capabilities.

This module can be used to configure the syslog application conceptual layers as implemented on the target system [RFC5424].

3. Design of the Syslog Model

The syslog model was designed by comparing various syslog features implemented by various vendors’ in different implementations.

This draft addresses the common leafs between implementations and creates a common model, which can be augmented with proprietary features, if necessary. The base model is designed to be very simple for maximum flexibility.

Syslog consists of originators, and collectors. The following digram shows syslog messages flowing from an originator, to collectors where suppression filtering can take place.

Many vendors extend the list of facilities available for logging in their implementation. An example is included in Extending Facilities (Appendix A.1).

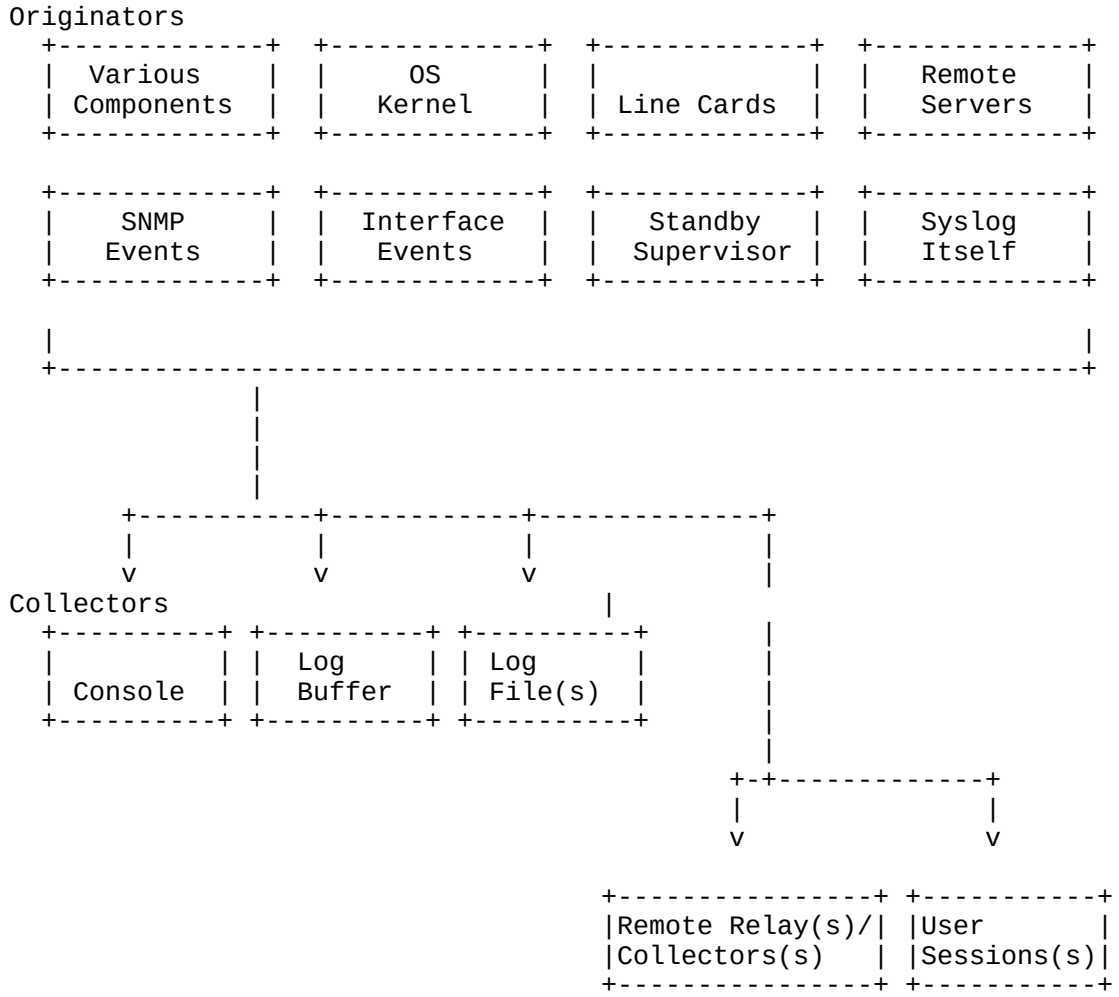


Figure 1. Syslog Processing Flow

The leaves in the base syslog model actions container correspond to each message collector:

console
log buffer
log file(s)
remote relay(s)/collector(s)
user session(s).

Within each action, a selector is used to filter syslog messages. A selector consists of two parts: one or more facility-severity

matches, and if supported via the select-match feature, an optional regular expression pattern match that is performed on the SYSLOG-MSG field.

The facility is one of a specific syslogtypes:syslog-facility, none, or all facilities. None is a special case that can be used to disable an action.

The severity is one of syslogtypes:severity, all severities, or none. None is a special case that can be used to disable a facility. When filtering severity, the default comparison is that all messages of the specified severity and higher are logged. This is shown in the model as ?default equals-or-higher?. This behavior can be altered if the select-sev-compare feature is enabled to specify: ?equals? to specify only this single severity; ?not-equals? to ignore that severity; ?equals-or-higher? to specify all messages of the specified severity and higher.

Optional features are used to specified functionality that is present in specific vendor configurations.

3.1. Syslog Module

A simplified graphical representation of the complete data tree is presented here.

Each node is printed as:

<status> <flags> <name> <opts> <type> <if-features>

<status> is one of:

```
          +  for current
          x  for deprecated
          o  for obsolete
```

<flags> is one of:

```
          rw  for configuration data
          ro  for non-configuration data
          -x  for rpcs
          -n  for notifications
```

<name> is the name of the node

```
          (<name>) means that the node is a choice node
          :(<name>) means that the node is a case node
```

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>.

<opts> is one of:

```
          ?  for an optional leaf or choice
          !  for a presence container
          *  for a leaf-list or list
          [<keys>] for a list's keys
```

<type> is the name of the type for leafs and leaf-lists

If the type is a leafref, the type is printed as "-> TARGET", where TARGET is either the leafref path, with prefixed removed if possible.

<if-features> is the list of features this node depends on, printed within curly brackets and a question mark "{...}?"

```
module: ietf-syslog
   +--rw syslog!
      +--rw actions
         +--rw console!
         |  +--rw selector
         |     +--rw (selector-facility)
         |     |  +--:(facility)
         |     |  |  +--rw no-facilities?   empty
         |     |  +--:(name)
         |     |     +--rw facility-list* [facility]
         |     |        +--rw facility    union
         |     |        +--rw severity    union
         |     |        +--rw compare?    enumeration {select-sev-compare}?
         |     +--rw pattern-match?   string {select-match}?
         +--rw buffer {buffer-action}?
         |  +--rw selector
         |  |  +--rw (selector-facility)
         |  |  |  +--:(facility)
         |  |  |  |  +--rw no-facilities?   empty
         |  |  |  +--:(name)
         |  |  |     +--rw facility-list* [facility]
         |  |  |        +--rw facility    union
         |  |  |        +--rw severity    union
         |  |  |        +--rw compare?    enumeration {select-sev-compare}?
         |  |  +--rw pattern-match?   string {select-match}?
         |  +--rw structured-data?         boolean {structured-data}?
         |  +--rw buffer-limit-bytes?      uint64 {buffer-limit-bytes}?
         |  +--rw buffer-limit-messages?   uint64 {buffer-limit-messages}?
         +--rw file
         |  +--rw log-file* [name]
```

```
|     +--rw name               inet:uri
|     +--rw selector
|     |  +--rw (selector-facility)
|     |  |  +--:(facility)
|     |  |  |  +--rw no-facilities?   empty
|     |  |  +--:(name)
|     |  |     +--rw facility-list* [facility]
|     |  |        +--rw facility    union
|     |  |        +--rw severity    union
|     |  |        +--rw compare?    enumeration {select-sev-compare}?
|     |  +--rw pattern-match?   string {select-match}?
|     +--rw structured-data?   boolean {structured-data}?
|     +--rw file-rotation
|        +--rw number-of-files?   uint32 {file-limit-size}?
|        +--rw max-file-size?     uint64 {file-limit-size}?
|        +--rw rollover?          uint32 {file-limit-duration}?
|        +--rw retention?         uint16 {file-limit-duration}?
+--rw remote
|  +--rw destination* [name]
|     +--rw name                 string
|     +--rw (transport)
|     |  +--:(tcp)
|     |  |  +--rw tcp
|     |  |     +--rw address?   inet:host
|     |  |     +--rw port?      inet:port-number
|     |  +--:(udp)
|     |     +--rw udp
|     |        +--rw address?   inet:host
|     |        +--rw port?      inet:port-number
|     +--rw selector
|     |  +--rw (selector-facility)
|     |  |  +--:(facility)
|     |  |  |  +--rw no-facilities?   empty
|     |  |  +--:(name)
|     |  |     +--rw facility-list* [facility]
|     |  |        +--rw facility    union
|     |  |        +--rw severity    union
|     |  |        +--rw compare?    enumeration {select-sev-compare}?
|     |  +--rw pattern-match?   string {select-match}?
|     +--rw structured-data?     boolean {structured-data}?
|     +--rw facility-override?   identityref
|     +--rw source-interface?    if:interface-ref
|     +--rw signing-options! {signed-messages}?
|        +--rw cert-initial-repeat    uint16
|        +--rw cert-resend-delay      uint16
|        +--rw cert-resend-count      uint16
|        +--rw max-delay              uint16
|        +--rw number-resends         uint16
```

```
         |        +--rw resend-delay           uint16
         |        +--rw resend-count           uint16
         +--rw session
            +--rw all-users!
            |  +--rw selector
            |     +--rw (selector-facility)
            |     |  +--:(facility)
            |     |  |  +--rw no-facilities?   empty
            |     |  +--:(name)
            |     |     +--rw facility-list* [facility]
            |     |        +--rw facility    union
            |     |        +--rw severity    union
            |     |        +--rw compare?    enumeration {select-sev-compare}?
            |     +--rw pattern-match?   string {select-match}?
            +--rw user* [name]
               +--rw name        string
               +--rw selector
                  +--rw (selector-facility)
                  |  +--:(facility)
                  |  |  +--rw no-facilities?   empty
                  |  +--:(name)
                  |     +--rw facility-list* [facility]
                  |        +--rw facility    union
                  |        +--rw severity    union
                  |        +--rw compare?    enumeration {select-sev-compare}?
                  +--rw pattern-match?   string {select-match}?
```

Figure 2. ietf-syslog Module Tree

4. Syslog YANG Modules

4.1. The ietf-syslog-types Module

This module references [RFC5424].

```
<CODE BEGINS> file "ietf-syslog-types.yang"
module ietf-syslog-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-syslog-types";
  prefix syslogtypes;

  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: Lou Berger
               <mailto:lberger@labn.net>
```

```
     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

     Editor:   Kiran Agrahara Sreenivasa
               <mailto:kkoushik@cisco.com>

     Editor:   Clyde Wildes
               <mailto:cwildes@cisco.com>";
  description
    "This module contains a collection of YANG type definitions for
     SYSLOG.

     Copyright (c) 2016 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).

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and
     'OPTIONAL' in the module text are to be interpreted as described
     in RFC 2119 (http://tools.ietf.org/html/rfc2119).

     This version of this YANG module is part of RFC XXXX
     (http://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  reference
    "RFC 5424: The Syslog Protocol";

  revision 2016-11-13 {
    description
      "Initial Revision";
    reference
      "RFC XXXX: SYSLOG YANG Model";
  }

  typedef severity {
    type enumeration {
      enum "emergency" {
        value 0;
        description
          "The severity level 'Emergency' indicating that the system
           is unusable.";
```

```
      }
      enum "alert" {
        value 1;
        description
          "The severity level 'Alert' indicating that an action must be
           taken immediately.";
      }
      enum "critical" {
        value 2;
        description
          "The severity level 'Critical' indicating a critical condition.";
      }
      enum "error" {
        value 3;
        description
          "The severity level 'Error' indicating an error condition.";
      }
      enum "warning" {
        value 4;
        description
          "The severity level 'Warning' indicating a warning condition.";
      }
      enum "notice" {
        value 5;
        description
          "The severity level 'Notice' indicating a normal but significant
           condition.";
      }
      enum "info" {
        value 6;
        description
          "The severity level 'Info' indicating an informational message.";
      }
      enum "debug" {
        value 7;
        description
          "The severity level 'Debug' indicating a debug-level message.";
      }
    }
    description
      "The definitions for Syslog message severity as per RFC 5424.";
  }

  identity syslog-facility {
    description
      "This identity is used as a base for all syslog facilities as
       per RFC 5424.";
  }
```

```
  identity kern {
    base syslog-facility;
    description
      "The facility for kernel messages (0) as defined in RFC 5424.";
  }

  identity user {
    base syslog-facility;
    description
      "The facility for user-level messages (1) as defined in RFC 5424.";
  }

  identity mail {
    base syslog-facility;
    description
      "The facility for the mail system (2) as defined in RFC 5424.";
  }

  identity daemon {
    base syslog-facility;
    description
      "The facility for the system daemons (3) as defined in RFC 5424.";
   }

  identity auth {
    base syslog-facility;
    description
      "The facility for security/authorization messages (4) as defined
       in RFC 5424.";
  }

  identity syslog {
    base syslog-facility;
    description
      "The facility for messages generated internally by syslogd
       facility (5) as defined in RFC 5424.";
  }

  identity lpr {
    base syslog-facility;
    description
      "The facility for the line printer subsystem (6) as defined in
       RFC 5424.";
  }

  identity news {
    base syslog-facility;
    description
```

```
      "The facility for the network news subsystem (7) as defined in
       RFC 5424.";
  }

  identity uucp {
    base syslog-facility;
    description
      "The facility for the UUCP subsystem (8) as defined in RFC 5424.";
  }

  identity cron {
    base syslog-facility;
    description
      "The facility for the clock daemon (9) as defined in RFC 5424.";
  }

  identity authpriv {
    base syslog-facility;
    description
      "The facility for privileged security/authorization messages (10)
       as defined in RFC 5424.";
  }

  identity ftp {
    base syslog-facility;
    description
      "The facility for the FTP daemon (11) as defined in RFC 5424.";
  }

  identity ntp {
    base syslog-facility;
    description
      "The facility for the NTP subsystem (12) as defined in RFC 5424.";
  }

  identity audit {
    base syslog-facility;
    description
      "The facility for log audit messages (13) as defined in RFC 5424.";
  }

  identity console {
    base syslog-facility;
    description
      "The facility for log alert messages (14) as defined in RFC 5424.";
  }

  identity cron2 {
```

```
    base syslog-facility;
    description
      "The facility for the second clock daemon (15) as defined in
       RFC 5424.";
  }

  identity local0 {
    base syslog-facility;
    description
      "The facility for local use 0 messages (16) as defined in
       RFC 5424.";
  }

  identity local1 {
    base syslog-facility;
    description
      "The facility for local use 1 messages (17) as defined in
       RFC 5424.";
  }

  identity local2 {
    base syslog-facility;
    description
      "The facility for local use 2 messages (18) as defined in
       RFC 5424.";
  }

  identity local3 {
    base syslog-facility;
    description
      "The facility for local use 3 messages (19) as defined in
       RFC 5424.";
  }

  identity local4 {
    base syslog-facility;
    description
      "The facility for local use 4 messages (20) as defined in
       RFC 5424.";
  }

  identity local5 {
    base syslog-facility;
    description
      "The facility for local use 5 messages (21) as defined in
       RFC 5424.";
  }
```

```
  identity local6 {
    base syslog-facility;
    description
      "The facility for local use 6 messages (22) as defined in
       RFC 5424.";
  }

  identity local7 {
    base syslog-facility;
    description
      "The facility for local use 7 messages (23) as defined in
       RFC 5424.";
  }
}
<CODE ENDS>
```

Figure 3. ietf-syslog-types Module

4.2. The ietf-syslog Module

This module imports typedefs from [RFC6021] and [RFC7223], and it references [RFC5424], [RFC5425], [RFC5426], [RFC6587], and [RFC5848].

```
<CODE BEGINS> file "ietf-syslog.yang"
module ietf-syslog {
  namespace "urn:ietf:params:xml:ns:yang:ietf-syslog";
  prefix syslog;

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-syslog-types {
    prefix syslogtypes;
  }

  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: Lou Berger
               <mailto:lberger@labn.net>
```

```
     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

     Editor:   Kiran Agrahara Sreenivasa
               <mailto:kkoushik@cisco.com>

     Editor:   Clyde Wildes
               <mailto:cwildes@cisco.com>";
  description
    "This module contains a collection of YANG definitions
     for syslog configuration.

     Copyright (c) 2016 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).

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
     NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and
     'OPTIONAL' in the module text are to be interpreted as described
     in RFC 2119 (http://tools.ietf.org/html/rfc2119).

     This version of this YANG module is part of RFC XXXX
     (http://tools.ietf.org/html/rfcXXXX); see the RFC itself for
     full legal notices.";

  reference
    "RFC 5424: The Syslog Protocol
     RFC 5426: Transmission of Syslog Messages over UDP
     RFC 6587: Transmission of Syslog Messages over TCP
     RFC 5848: Signed Syslog Messages";

  revision 2016-11-13 {
    description
      "Initial Revision";
    reference
      "RFC XXXX: Syslog YANG Model";
  }

  feature buffer-action {
    description
      "This feature indicates that the local memory logging buffer
       action is supported.";
```

```
  }

  feature buffer-limit-bytes {
    description
      "This feature indicates that the local memory logging buffer
       is limited in size using a limit expressed in bytes.";
  }

  feature buffer-limit-messages {
    description
      "This feature indicates that the local memory logging buffer
       is limited in size using a limit expressed in number of log
       messages.";
  }

  feature file-limit-size {
    description
      "This feature indicates that file logging resources
       are managed using size and number limits.";
  }

  feature file-limit-duration {
    description
      "This feature indicates that file logging resources
       are managed using time based limits.";
  }

  feature select-sev-compare {
    description
      "This feature represents the ability to select messages
       using the additional operators equal to, or not equal to
       when comparing the syslog message severity.";
  }

  feature select-match {
    description
      "This feature represents the ability to select messages based
       on a Posix 1003.2 regular expression pattern match.";
  }

  feature structured-data {
    description
      "This feature represents the ability to log messages
       in structured-data format as per RFC 5424.";
  }

  feature signed-messages {
    description
```

```
      "This feature represents the ability to configure signed
       syslog messages according to RFC 5848.";
  }

  grouping log-severity {
    description
      "This grouping defines the severity value that is used to
       select log messages.";
    leaf severity {
      type union {
        type syslogtypes:severity;
        type enumeration {
          enum none {
            value -2;
            description
              "This enum describes the case where no severities
               are selected.";
          }
          enum all {
            value -1;
            description
              "This enum describes the case where all severities
               are selected.";
          }
        }
      }
      mandatory true;
      description
        "This leaf specifies the syslog message severity. When
         severity is specified, the default severity comparison
         is all messages of the specified severity and greater are
         selected. 'all' is a special case which means all severities
         are selected. 'none' is a special case which means that
         no selection should occur or disable this filter.";
    }
    leaf compare {
      when '../severity != "all" and
           ../severity != "none"' {
        description
          "The compare leaf is not applicable for severity 'all' or
           severity 'none'";
      }
      if-feature select-sev-compare;
      type enumeration {
        enum equals-or-higher {
          description
            "This enum specifies all messages of the specified
             severity and higher are logged according to the
```

```
             given log-action";
        }
        enum equals {
          description
            "This enum specifies all messages that are for
             the specified severity are logged according to the
             given log-action";
        }
        enum not-equals {
          description
            "This enum specifies all messages that are not for
             the specified severity are logged according to the
             given log-action";
        }
      }
      default equals-or-higher;
      description
        "This leaf describes the option to specify how the
         severity comparison is performed.";
    }
  }

  grouping selector {
    description
      "This grouping defines a syslog selector which is used to
       select log messages for the log-action (console, file,
       remote, etc). Choose one of the following:
         no-facility
         facility [<facility> <severity>...]";
    container selector {
      description
        "This container describes the log selector parameters
         for syslog.";
      choice selector-facility {
        mandatory true;
        description
          "This choice describes the option to specify no
           facilities, or a specific facility which can be
           all for all facilities.";
        case facility {
          description
            "This case specifies no facilities will match when
             comparing the syslog message facility. This is a
             method that can be used to effectively disable a
             particular log-action (buffer, file, etc).";
          leaf no-facilities {
            type empty;
            description
```

```
            "This leaf specifies that no facilities are selected
             for this log-action.";
          }
        }
        case name {
          description
            "This case specifies one or more specified facilities
             will match when comparing the syslog message facility.";
          list facility-list {
            key facility;
            description
              "This list describes a collection of syslog
               facilities and severities.";
            leaf facility {
              type union {
                type identityref {
                  base syslogtypes:syslog-facility;
                }
                type enumeration {
                  enum all {
                    description
                      "This enum describes the case where all
                       facilities are requested.";
                  }
                }
              }
              description
                "The leaf uniquely identifies a syslog facility.";
            }
            uses log-severity;
          }
        }
      }
      leaf pattern-match {
        if-feature select-match;
        type string;
        description
          "This leaf desribes a Posix 1003.2 regular expression
           string that can be used to select a syslog message for
           logging. The match is performed on the RFC 5424
           SYSLOG-MSG field.";
      }
    }
  }

  grouping structured-data {
    description
      "This grouping defines the syslog structured data option
```

```
       which is used to select the format used to write log
       messages.";
    leaf structured-data {
      if-feature structured-data;
      type boolean;
      default false;
      description
        "This leaf describes how log messages are written.
         If true, messages will be written with one or more
         STRUCTURED-DATA elements as per RFC5424; if false,
         messages will be written with STRUCTURED-DATA =
         NILVALUE.";
    }
  }

  container syslog {
    presence "Enables logging.";
    description
      "This container describes the configuration parameters for
       syslog.";
    container actions {
      description
        "This container describes the log-action parameters
         for syslog.";
      container console {
        presence "Enables logging console configuration";
        description
          "This container describes the configuration parameters for
           console logging.";
        uses selector;
      }
      container buffer {
        if-feature buffer-action;
        description
          "This container describes the configuration parameters for
           local memory buffer logging. The buffer is circular in
           nature, so newer messages overwrite older messages after
           the buffer is filled. The method used to read syslog messages
           from the buffer is supplied by the local implementation.";
        uses selector;
        uses structured-data;
        leaf buffer-limit-bytes {
          if-feature buffer-limit-bytes;
          type uint64;
          units "bytes";
          description
            "This leaf configures the amount of memory (in bytes) that
             will be dedicated to the local memory logging buffer.
```

```
            The default value varies by implementation.";
        }
        leaf buffer-limit-messages {
          if-feature buffer-limit-messages;
          type uint64;
          units "log messages";
          description
            "This leaf configures the number of log messages that
             will be dedicated to the local memory logging buffer.
             The default value varies by implementation.";
        }
      }
      container file {
        description
          "This container describes the configuration parameters for
           file logging. If file-archive limits are not supplied, it
           is assumed that the local implementation defined limits will
           be used.";
        list log-file {
          key "name";
          description
            "This list describes a collection of local logging
             files.";
          leaf name {
            type inet:uri {
              pattern 'file:.*';
            }
            description
              "This leaf specifies the name of the log file which
               MUST use the uri scheme file:.";
          }
          uses selector;
          uses structured-data;
          container file-rotation {
            description
              "This container describes the configuration
               parameters for log file rotation.";
            leaf number-of-files {
              if-feature file-limit-size;
              type uint32;
              description
                "This leaf specifies the maximum number of log
                 files retained. Specify 1 for implementations
                 that only support one log file.";
            }
            leaf max-file-size {
              if-feature file-limit-size;
              type uint64;
```

```
            units "megabytes";
            description
              "This leaf specifies the maximum log file size.";
          }
          leaf rollover {
            if-feature file-limit-duration;
            type uint32;
            units "minutes";
            description
              "This leaf specifies the length of time that log
               events should be written to a specific log file.
               Log events that arrive after the rollover period
               cause the current log file to be closed and a new
               log file to be opened.";
          }
          leaf retention {
            if-feature file-limit-duration;
            type uint16;
            units "hours";
            description
              "This leaf specifies the length of time that
               completed/closed log event files should be stored
               in the file system before they are deleted.";
          }
        }
      }
    }
    container remote {
      description
        "This container describes the configuration parameters for
         forwarding syslog messages to remote relays or collectors.";
      list destination {
        key "name";
        description
          "This list describes a collection of remote logging
           destinations.";
        leaf name {
          type string;
          description
            "An arbitrary name for the endpoint to connect to.";
        }
        choice transport {
          mandatory true;
          description
            "This choice describes the transport option.";
          case tcp {
            container tcp {
              description
```

```
                "This container describes the TCP transport
                 options.";
              reference
                "RFC 6587: Transmission of Syslog Messages over TCP";
              leaf address {
                type inet:host;
                description
                  "The leaf uniquely specifies the address of
                   the remote host. One of the following must
                   be specified: an ipv4 address, an ipv6
                   address, or a host name.";
              }
              leaf port {
                type inet:port-number;
                default 514;
                description
                  "This leaf specifies the port number used to
                   deliver messages to the remote server.";
              }
            }
          }
          case udp {
            container udp {
              description
                "This container describes the UDP transport
                 options.";
              reference
                "RFC 5426: Transmission of Syslog Messages over UDP";
              leaf address {
                type inet:host;
                description
                  "The leaf uniquely specifies the address of
                   the remote host. One of the following must be
                   specified: an ipv4 address, an ipv6 address,
                   or a host name.";
              }
              leaf port {
                type inet:port-number;
                default 514;
                description
                  "This leaf specifies the port number used to
                   deliver messages to the remote server.";
              }
            }
          }
        }
        uses selector;
        uses structured-data;
```

```
          leaf facility-override {
            type identityref {
              base syslogtypes:syslog-facility;
            }
            description
              "If specified, this leaf specifies the facility used
               to override the facility in messages delivered to the
               remote server.";
          }
          leaf source-interface {
            type if:interface-ref;
            description
              "This leaf sets the source interface to be used to send
               message to the remote syslog server. If not set,
               messages sent to a remote syslog server will
               contain the IP address of the interface the syslog
               message uses to exit the network element";
          }
          container signing-options {
            if-feature signed-messages;
            presence
              "If present, syslog-signing options is activated.";
            description
              "This container describes the configuration
               parameters for signed syslog messages as described
               by RFC 5848.";
            reference
              "RFC 5848: Signed Syslog Messages";
            leaf cert-initial-repeat {
              type uint16;
              mandatory true;
              description
              "This leaf specifies the number of times each
               Certificate Block should be sent before the first
               message is sent.";
            }
            leaf cert-resend-delay {
              type uint16;
              units "seconds";
              mandatory true;
              description
                "This leaf specifies the maximum time delay in
                 seconds until resending the Certificate Block.";
            }
            leaf cert-resend-count {
              type uint16;
              mandatory true;
              description
```

```
              "This leaf specifies the maximum number of other
               syslog messages to send until resending the
               Certificate Block.";
            }
            leaf max-delay {
              type uint16;
              units "seconds";
              mandatory true;
              description
                "This leaf specifies when to generate a new
                 Signature Block. If this many seconds have
                 elapsed since the message with the first message
                 number of the Signature Block was sent, a new
                 Signature Block should be generated.";
            }
            leaf number-resends {
              type uint16;
              mandatory true;
              description
                "This leaf specifies the number of times a
                 Signature Block is resent. (It is recommended to
                 select a value of greater than 0 in particular
                 when the UDP transport [RFC5426] is used.).";
            }
            leaf resend-delay {
              type uint16;
              units "seconds";
              mandatory true;
              description
                "This leaf specifies when to send the next
                 Signature Block transmission based on time. If
                 this many seconds have elapsed since the previous
                 sending of this Signature Block, resend it.";
            }
            leaf resend-count {
              type uint16;
              mandatory true;
              description
                "This leaf specifies when to send the next
                 Signature Block transmission based on a count.
                 If this many other syslog messages have been sent
                 since the previous sending of this Signature
                 Block, resend it.";
            }
          }
        }
      }
      container session {
```

```
        description
          "This container describes the configuration parameters for
           user CLI session logging configuration.";
        container all-users {
          presence "Enables logging to all user sessions.";
          description
            "This container describes the configuration
             parameters for all users.";
          uses selector;
        }
        list user {
          key "name";
          description
            "This list describes a collection of user names.";
          leaf name {
            type string;
            description
              "This leaf uniquely describes a user name which
               is the login name of the user whose session
               is to receive log messages.";
          }
          uses selector;
        }
      }
    }
  }
}
<CODE ENDS>
```

Figure 4. ietf-syslog Module

5. Usage Examples

Requirement:
Enable console logging of syslogs of severity critical

Here is the example syslog configuration xml:

```
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <syslog xmlns="urn:ietf:params:xml:ns:yang:ietf-syslog"
          xmlns:syslog="urn:ietf:params:xml:ns:yang:ietf-syslog">
    <actions>
      <console>
        <selector>
          <facility-list>
            <facility>all</facility>
            <severity>critical</severity>
          </facility>
        </selector>
      </console>
    </actions>
  </syslog>
</config>
```

Enable remote logging of syslogs to udp destination 2001:db8:a0b:12f0::1
for facility auth, severity error

```
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <syslog xmlns="urn:ietf:params:xml:ns:yang:ietf-syslog"
          xmlns:syslog="urn:ietf:params:xml:ns:yang:ietf-syslog">
    <actions>
      <remote>
        <destination>
          <name>remote1</name>
          <udp>
            <address>2001:db8:a0b:12f0::1</address>
          </udp>
          <selector>
            <facility-list>
              <facility xmlns:syslogtypes=
                "urn:ietf:params:xml:ns:yang:ietf-syslog-types">
                syslogtypes:auth</facility>
              <severity>error</severity>
            <facility>
          <selector>
        </destination>
      </remote>
    </actions>
  </syslog>
</config>
```

Figure 5. ietf-syslog Examples

6. Acknowledgements

The authors wish to thank the following who commented on this proposal:

Martin Bjorklund
Jim Gibson
Jeffrey Haas
John Heasley
Giles Heron
Lisa Huang
Mahesh Jethanandani
Jeffrey K Lange
Jan Lindblad
Chris Lonvick
Tom Petch
Juergen Schoenwaelder
Phil Shafer
Jason Sterne
Peter Van Horne
Bert Wijnen
Aleksandr Zhdankin

7. IANA Considerations

This document registers two URIs 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-syslog-types

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-syslog-types namespace: urn:ietf:params:xml:ns:yang:ietf-syslog-types

prefix: ietf-syslog-types reference: RFC XXXX

Following the format in RFC 3688, the following registration is requested to be made:

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

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-syslog namespace: urn:ietf:params:xml:ns:yang:ietf-syslog

prefix: ietf-syslog

reference: RFC XXXX

8. 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]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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.

8.1. Resource Constraints

Network administrators must take the time to estimate the appropriate memory limits caused by the configuration of actions/buffer using buffer-limit-bytes and/or buffer-limit-messages where necessary to limit the amount of memory used.

Network administrators must take the time to estimate the appropriate storage capacity caused by the configuration of actions/file using file-archive attributes to limit storage used.

It is the responsibility of the network administrator to ensure that the configured message flow does not overwhelm system resources.

8.2. Inappropriate Configuration

It is the responsibility of the network admisintrator to ensure that the messages are actually going to the intended recipients.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <http://www.rfc-editor.org/info/rfc5424>.

[RFC5426] Okmianski, A., "Transmission of Syslog Messages over UDP", RFC 5426, DOI 10.17487/RFC5426, March 2009, <http://www.rfc-editor.org/info/rfc5426>.

[RFC5848] Kelsey, J., Callas, J., and A. Clemm, "Signed Syslog Messages", RFC 5848, DOI 10.17487/RFC5848, May 2010, <http://www.rfc-editor.org/info/rfc5848>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <http://www.rfc-editor.org/info/rfc6021>.

[RFC6587] Gerhards, R. and C. Lonvick, "Transmission of Syslog Messages over TCP", RFC 6587, DOI 10.17487/RFC6587, April 2012, <http://www.rfc-editor.org/info/rfc6587>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

9.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

Appendix A. Implementor Guidelines

A.1. Extending Facilities

Many vendors extend the list of facilities available for logging in their implementation. Additional facilities may not work with the syslog protocol as defined in [RFC5424] and hence such facilities apply for local syslog-like logging functionality.

The following is an example that shows how additional facilities could be added to the list of available facilities (in this example two facilities are added):

```
   module vendor-syslog-types-example {
     namespace "urn:vendor:params:xml:ns:yang:vendor-syslog-types";
     prefix vendor-syslogtypes;

     import ietf-syslog-types {
       prefix syslogtypes;
     }

     organization "Example, Inc.";
     contact
       "Example, Inc.
        Customer Service

        E-mail: syslog-yang@example.com";

     description
       "This module contains a collection of vendor-specific YANG type
        definitions for SYSLOG.";

     revision 2016-11-13 {
       description
         "Version 1.0";
       reference
         "Vendor SYSLOG Types: SYSLOG YANG Model";
     }

     identity vendor_specific_type_1 {
       base syslogtypes:syslog-facility;
     }

     identity vendor_specific_type_2 {
       base syslogtypes:syslog-facility;
     }
   }
```

Authors' Addresses

Clyde Wildes (editor)
Cisco Systems Inc.
170 West Tasman Drive
San Jose, CA 95134
US

Phone: +1 408 527-2672
Email: cwildes@cisco.com

Kiran Koushik (editor)
Cisco Systems Inc.
12515Research Blvd., Building 4
Austin, TX 78759
US

Phone: +1 512 378-1482
Email: kkoushik@cisco.com

NETMOD WG
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2018

C. Wildes, Ed.
Cisco Systems Inc.
K. Koushik, Ed.
Verizon Wireless
March 15, 2018

A YANG Data Model for Syslog Configuration
draft-ietf-netmod-syslog-model-26

Abstract

This document defines a YANG data model for the configuration of a syslog process. It is intended this model be used by vendors who implement syslog in their systems.

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 September 14, 2018.

Table of Contents

1. Introduction

This document defines a YANG [RFC7950] configuration data model that may be used to configure the syslog feature running on a system. YANG models can be used with network management protocols such as NETCONF [RFC6241] to install, manipulate, and delete the configuration of network devices.

The data model makes use of the YANG "feature" construct which allows implementations to support only those syslog features that lie within their capabilities.

This module can be used to configure the syslog application conceptual layers as implemented on the target system.

Essentially, a syslog process receives messages (from the kernel, processes, applications or other syslog processes) and processes them. The processing may involve logging to a local file, and/or displaying on console, and/or relaying to syslog processes on other machines. The processing is determined by the "facility" that originated the message and the "severity" assigned to the message by the facility.

Such definitions of syslog protocol are defined in [RFC5424], and are used in this RFC.

The YANG model in this document conforms to the Network Management Datastore Architecture defined in [draft-ietf-netmod-revised-datastores].

1.1. Requirements Language

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

The term "originator" is defined in [RFC5424]: an "originator" generates syslog content to be carried in a message.

The term "relay" is defined in [RFC5424]: a "relay" forwards messages, accepting messages from originators or other relays and sending them to collectors or other relays

The term "collectors" is defined in [RFC5424]: a "collector" gathers syslog content for further analysis.

The term "action" refers to the processing that takes place for each syslog message received.

1.3. NDMA Compliance

The YANG model in this document conforms to the Network Management Datastore Architecture defined in I-D.ietf-netmod-revised-datastores [I-D.ietf-netmod-revised-datastores].

1.4. Editorial Note (To be removed by RFC Editor)

This document contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

- "I-D.ietf-netconf-keystore" --> the assigned RFC value for draft-ietf-netconf-keystore
- "I-D.ietf-netconf-tls-client-server" --> the assigned RFC value for draft-ietf-netconf-tls-client-server
- "zzzz" --> the assigned RFC value for this draft
- I-D.ietf-netmod-revised-datastores --> the assigned RFC value for draft-ietf-netmod-revised-datastores

2. Design of the Syslog Model

The syslog model was designed by comparing various syslog features

implemented by various vendors' in different implementations.

This document addresses the common leafs between implementations and creates a common model, which can be augmented with proprietary features, if necessary. This model is designed to be very simple for maximum flexibility.

Some optional features are defined in this document to specify functionality that is present in specific vendor configurations.

Syslog consists of originators and collectors. The following diagram shows syslog messages flowing from originators, to collectors where filtering can take place.

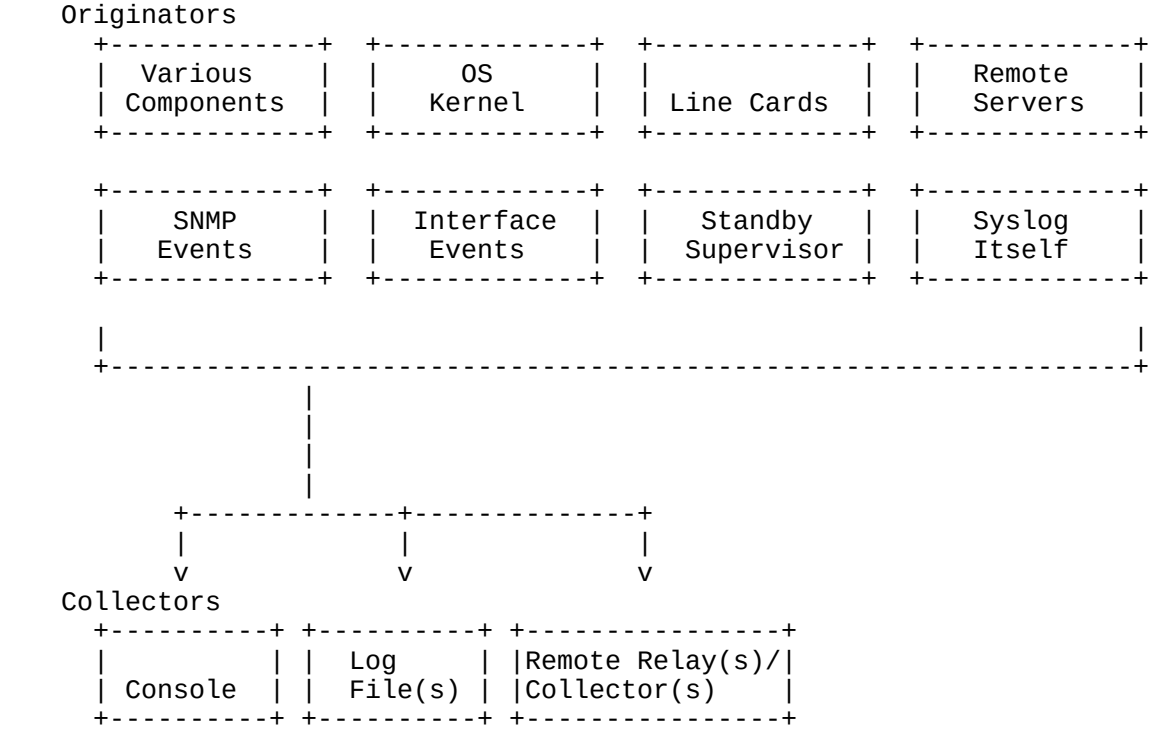


Figure 1. Syslog Processing Flow

Collectors are configured using the leaves in the syslog model "actions" container which correspond to each message collector:

console

log file(s)

remote relay(s)/collector(s)

Within each action, a selector is used to filter syslog messages. A selector consists of a list of one or more filters specified by facility-severity pairs, and, if supported via the select-match feature, an optional regular expression pattern match that is performed on the [RFC5424] field.

A syslog message is processed if:

```
    There is an element of facility-list (F, S) where
        the message facility matches F
        and the message severity matches S
    and/or the message text matches the regex pattern (if it
        is present)
```

The facility is one of a specific syslog-facility, or all facilities.

The severity is one of type syslog-severity, all severities, or none. None is a special case that can be used to disable a filter. When filtering severity, the default comparison is that messages of the specified severity and higher are selected to be logged. This is shown in the model as "default equals-or-higher". This behavior can be altered if the select-adv-compare feature is enabled to specify a compare operation and an action. Compare operations are: "equals" to select messages with this single severity, or "equals-or-higher" to select messages of the specified severity and higher. Actions are used to log the message or block the message from being logged.

Many vendors extend the list of facilities available for logging in their implementation. An example is included in Extending Facilities (Appendix A.1).

2.1. Syslog Module

A simplified graphical representation of the data model is used in this document. Please see [I-D.ietf-netmod-yang-tree-diagrams] for tree diagram notation.

```
   module: ietf-syslog
      +--rw syslog!
         +--rw actions
            +--rw console! {console-action}?
            |  +--rw facility-filter
            |  |  +--rw facility-list* [facility severity]
            |  |     +--rw facility            union
            |  |     +--rw severity            union
            |  |     +--rw advanced-compare {select-adv-compare}?
            |  |        +--rw compare?   enumeration
            |  |        +--rw action?    enumeration
            |  +--rw pattern-match?     string {select-match}?
            +--rw file {file-action}?
            |  +--rw log-file* [name]
            |     +--rw name                inet:uri
            |     +--rw facility-filter
            |     |  +--rw facility-list* [facility severity]
            |     |     +--rw facility            union
            |     |     +--rw severity            union
            |     |     +--rw advanced-compare {select-adv-compare}?
            |     |        +--rw compare?   enumeration
            |     |        +--rw action?    enumeration
            |     +--rw pattern-match?     string {select-match}?
            |     +--rw structured-data?   boolean {structured-data}?
            |     +--rw file-rotation
            |        +--rw number-of-files?   uint32 {file-limit-size}?
            |        +--rw max-file-size?     uint32 {file-limit-size}?
            |        +--rw rollover?          uint32
            |        |       {file-limit-duration}?
            |        +--rw retention?         uint32
            |                 {file-limit-duration}?
            +--rw remote {remote-action}?
               +--rw destination* [name]
                  +--rw name                 string
                  +--rw (transport)
                  |  +--:(udp)
                  |  |  +--rw udp
                  |  |     +--rw address?   inet:host
                  |  |     +--rw port?      inet:port-number
                  |  +--:(tls)
                  |     +--rw tls
                  |        +--rw address?        inet:host
                  |        +--rw port?           inet:port-number
                  |        +--rw client-auth
                  |        |  +--rw (auth-type)?
                  |        |     +--:(certificate)
                  |        |        +--rw certificate?   leafref
                  |        +--rw server-auth
                  |        |  +--rw pinned-ca-certs?       leafref
                  |        |  +--rw pinned-server-certs?   leafref
                  |        +--rw hello-params
                  |                 {tls-client-hello-params-config}?
                  |           +--rw tls-versions
```

```
|           |  +--rw tls-version*   identityref
|           +--rw cipher-suites
|              +--rw cipher-suite*   identityref
+--rw facility-filter
|  +--rw facility-list* [facility severity]
|     +--rw facility            union
|     +--rw severity            union
|     +--rw advanced-compare {select-adv-compare}?
|        +--rw compare?   enumeration
|        +--rw action?    enumeration
+--rw pattern-match?       string {select-match}?
+--rw structured-data?     boolean {structured-data}?
+--rw facility-override?   identityref
+--rw source-interface?    if:interface-ref
|       {remote-source-interface}?
+--rw signing! {signed-messages}?
   +--rw cert-signers
      +--rw cert-signer* [name]
      |  +--rw name              string
      |  +--rw cert
      |  |  +--rw algorithm?
      |  |  |       identityref
      |  |  +--rw private-key?
      |  |  |       union
      |  |  +--rw public-key?
      |  |  |       binary
      |  |  +---x generate-private-key
      |  |  |  +---w input
      |  |  |     +---w algorithm?
      |  |  |             identityref
      |  |  +--rw certificates
      |  |  |  +--rw certificate* [name]
      |  |  |     +--rw name     string
      |  |  |     +--rw value?   binary
      |  |  +---x generate-certificate-signing-request
      |  |     +---w input
      |  |     |  +---w subject       binary
      |  |     |  +---w attributes?   binary
      |  |     +--ro output
      |  |        +--ro certificate-signing-request
      |  |                binary
      |  +--rw hash-algorithm?   enumeration
      +--rw cert-initial-repeat?   uint32
      +--rw cert-resend-delay?     uint32
      +--rw cert-resend-count?     uint32
      +--rw sig-max-delay?         uint32
      +--rw sig-number-resends?    uint32
      +--rw sig-resend-delay?      uint32
      +--rw sig-resend-count?      uint32
```

Figure 2. ietf-syslog Module Tree

3. Syslog YANG Module

3.1. The ietf-syslog Module

This module imports typedefs from [RFC6991], [I-D.ietf-netmod-rfc7223bis], groupings from [I-D.ietf-netconf-keystore], and [I-D.ietf-netconf-tls-client-server], and it references [RFC5424], [RFC5425], [RFC5426], [RFC5848], [RFC8089], [RFC8174], and [Std-1003.1-2008].

```
   <CODE BEGINS> file "ietf-syslog@2018-03-15.yang"
   module ietf-syslog {
     yang-version 1.1;

     namespace "urn:ietf:params:xml:ns:yang:ietf-syslog";
     prefix syslog;

     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     import ietf-interfaces {
       prefix if;
       reference
         "I-D.ietf-netmod-rfc7223bis: A YANG Data Model
          for Interface Management";
     }

     import ietf-tls-client {
       prefix tlsc;
       reference
         "I-D.ietf-netconf-tls-client-server:
          YANG Groupings for TLS Clients and TLS Servers";
     }

     import ietf-keystore {
       prefix ks;
       reference
         "I-D.ietf-netconf-keystore: YANG Data Model for a
          Keystore Mechanism";
     }

     organization
         "IETF NETMOD (Network Modeling) Working Group";

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

        Editor:   Kiran Agrahara Sreenivasa
                  <mailto:kirankoushik.agraharasreenivasa@
                          verizonwireless.com>

        Editor:   Clyde Wildes
                  <mailto:cwildes@cisco.com>";
     description
       "This module contains a collection of YANG definitions
        for syslog configuration.

        Copyright (c) 2018 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).

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
        NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and
        'OPTIONAL' in the module text are to be interpreted as
        described in RFC 2119 (http://tools.ietf.org/html/rfc2119).

        This version of this YANG module is part of RFC zzzz
        (http://tools.ietf.org/html/rfczzzz); see the RFC itself for
        full legal notices.";

    revision 2018-03-15 {
      description
        "Initial Revision";
      reference
        "RFC zzzz: Syslog YANG Model";
    }

    feature console-action {
      description
        "This feature indicates that the local console action is
         supported.";
    }

    feature file-action {
      description
        "This feature indicates that the local file action is
         supported.";
    }

    feature file-limit-size {
      description
        "This feature indicates that file logging resources
         are managed using size and number limits.";
    }

    feature file-limit-duration {
      description
        "This feature indicates that file logging resources
         are managed using time based limits.";
    }

    feature remote-action {
      description
        "This feature indicates that the remote server action is
         supported.";
    }
```

```
     feature remote-source-interface {
       description
         "This feature indicates that source-interface is supported
          supported for the remote-action.";
     }

     feature select-adv-compare {
       description
         "This feature represents the ability to select messages
          using the additional comparison operators when comparing
          the syslog message severity.";
     }

     feature select-match {
       description
         "This feature represents the ability to select messages
          based on a Posix 1003.2 regular expression pattern match.";
     }

     feature structured-data {
       description
         "This feature represents the ability to log messages
          in structured-data format.";
       reference
         "RFC 5424: The Syslog Protocol";
     }

     feature signed-messages {
       description
         "This feature represents the ability to configure signed
          syslog messages.";
       reference
         "RFC 5848: Signed Syslog Messages";
     }

     typedef syslog-severity {
         type enumeration {
         enum "emergency" {
           value 0;
           description
             "The severity level 'Emergency' indicating that the
              system is unusable.";
         }
         enum "alert" {
           value 1;
           description
             "The severity level 'Alert' indicating that an action
              must be taken immediately.";
         }
         enum "critical" {
           value 2;
           description
             "The severity level 'Critical' indicating a critical
              condition.";
```

```
        }
        enum "error" {
          value 3;
          description
            "The severity level 'Error' indicating an error
             condition.";
        }
        enum "warning" {
          value 4;
          description
            "The severity level 'Warning' indicating a warning
             condition.";
        }
        enum "notice" {
          value 5;
          description
            "The severity level 'Notice' indicating a normal but
             significant condition.";
        }
        enum "info" {
          value 6;
          description
            "The severity level 'Info' indicating an informational
             message.";
        }
        enum "debug" {
          value 7;
          description
            "The severity level 'Debug' indicating a debug-level
             message.";
        }
      }
      description
        "The definitions for Syslog message severity.
         Note that a lower value is a higher severity. Comparisons of
         equal-or-higher severity mean equal or lower numeric value";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity syslog-facility {
      description
        "This identity is used as a base for all syslog facilities.";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity kern {
      base syslog-facility;
      description
        "The facility for kernel messages (0).";
      reference
        "RFC 5424: The Syslog Protocol";
    }
```

```
identity user {
  base syslog-facility;
  description
    "The facility for user-level messages (1).";
  reference
    "RFC 5424: The Syslog Protocol";
}

identity mail {
  base syslog-facility;
  description
    "The facility for the mail system (2).";
  reference
    "RFC 5424: The Syslog Protocol";
}

identity daemon {
  base syslog-facility;
  description
    "The facility for the system daemons (3).";
  reference
    "RFC 5424: The Syslog Protocol";
 }

identity auth {
  base syslog-facility;
  description
    "The facility for security/authorization messages (4).";
  reference
    "RFC 5424: The Syslog Protocol";
}

identity syslog {
  base syslog-facility;
  description
    "The facility for messages generated internally by syslogd
     facility (5).";
  reference
    "RFC 5424: The Syslog Protocol";
}

identity lpr {
  base syslog-facility;
  description
    "The facility for the line printer subsystem (6).";
  reference
    "RFC 5424: The Syslog Protocol";
}

identity news {
  base syslog-facility;
  description
    "The facility for the network news subsystem (7).";
```

```
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity uucp {
      base syslog-facility;
      description
        "The facility for the UUCP subsystem (8).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity cron {
      base syslog-facility;
      description
        "The facility for the clock daemon (9).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity authpriv {
      base syslog-facility;
      description
        "The facility for privileged security/authorization messages
         (10).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity ftp {
      base syslog-facility;
      description
        "The facility for the FTP daemon (11).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity ntp {
      base syslog-facility;
      description
        "The facility for the NTP subsystem (12).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity audit {
      base syslog-facility;
      description
        "The facility for log audit messages (13).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity console {
```

```
      base syslog-facility;
      description
        "The facility for log alert messages (14).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity cron2 {
      base syslog-facility;
      description
        "The facility for the second clock daemon (15).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity local0 {
      base syslog-facility;
      description
        "The facility for local use 0 messages (16).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity local1 {
      base syslog-facility;
      description
        "The facility for local use 1 messages (17).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity local2 {
      base syslog-facility;
      description
        "The facility for local use 2 messages (18).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity local3 {
      base syslog-facility;
      description
        "The facility for local use 3 messages (19).";
      reference
        "RFC 5424: The Syslog Protocol";
    }

    identity local4 {
      base syslog-facility;
      description
        "The facility for local use 4 messages (20).";
      reference
        "RFC 5424: The Syslog Protocol";
    }
```

```
identity local5 {
  base syslog-facility;
  description
    "The facility for local use 5 messages (21).";
  reference
    "RFC 5424: The Syslog Protocol";
}

identity local6 {
  base syslog-facility;
  description
    "The facility for local use 6 messages (22).";
  reference
    "RFC 5424: The Syslog Protocol";
}

identity local7 {
  base syslog-facility;
  description
    "The facility for local use 7 messages (23).";
  reference
    "RFC 5424: The Syslog Protocol";
}

grouping severity-filter {
  description
    "This grouping defines the processing used to select
     log messages by comparing syslog message severity using
     the following processing rules:
      - if 'none', do not match.
      - if 'all', match.
      - else compare message severity with the specified severity
        according to the default compare rule (all messages of the
        specified severity and greater match) or if the
        select-adv-compare feature is present, use the
        advance-compare rule.";
  leaf severity {
    type union {
      type syslog-severity;
      type enumeration {
        enum none {
          value 2147483647;
          description
            "This enum describes the case where no severities
             are selected.";
        }
        enum all {
          value -2147483648;
          description
            "This enum describes the case where all severities
             are selected.";
        }
      }
```

```
          }
          mandatory true;
          description
            "This leaf specifies the syslog message severity.";
        }
        container advanced-compare {
          when '../severity != "all" and
               ../severity != "none"' {
            description
              "The advanced compare container is not applicable for
               severity 'all' or severity 'none'";
          }
          if-feature select-adv-compare;
          leaf compare {
            type enumeration {
              enum equals {
                description
                  "This enum specifies that the severity comparison
                   operation will be equals.";
              }
              enum equals-or-higher {
                description
                  "This enum specifies that the severity comparison
                   operation will be equals or higher.";
              }
            }
            default equals-or-higher;
            description
              "The compare can be used to specify the comparison
               operator that should be used to compare the syslog message
               severity with the specified severity.";
          }
          leaf action {
            type enumeration {
              enum log {
                description
                  "This enum specifies that if the compare operation is
                   true the message will be logged.";
               }
              enum block {
                description
                  "This enum specifies that if the compare operation is
                   true the message will not be logged.";
              }
            }
            default log;
            description
              "The action can be used to specify if the message should
               be logged or blocked based on the outcome of the compare
               operation.";
          }
          description
            "This container describes additional severity compare
             operations that can be used in place of the default
```

```
            severity comparison. The compare leaf specifies the type of
            the compare that is done and the action leaf specifies the
            intended result.
            Example: compare->equals and action->block means
            messages that have a severity that are equal to the
            specified severity will not be logged.";
        }
      }

      grouping selector {
        description
          "This grouping defines a syslog selector which is used to
           select log messages for the log-actions (console, file,
           remote, etc.). Choose one or both of the following:
             facility [<facility> <severity>...]
             pattern-match regular-expression-match-string
           If both facility and pattern-match are specified, both must
           match in order for a log message to be selected.";
        container facility-filter {
          description
            "This container describes the syslog filter parameters.";
          list facility-list {
            key "facility severity";
            ordered-by user;
            description
              "This list describes a collection of syslog
               facilities and severities.";
            leaf facility {
              type union {
                type identityref {
                  base syslog-facility;
                }
                type enumeration {
                  enum all {
                    description
                      "This enum describes the case where all
                       facilities are requested.";
                  }
                }
              }
              description
                "The leaf uniquely identifies a syslog facility.";
            }
            uses severity-filter;
          }
        }
        leaf pattern-match {
          if-feature select-match;
          type string;
          description
            "This leaf describes a Posix 1003.2 regular expression
             string that can be used to select a syslog message for
             logging. The match is performed on the SYSLOG-MSG field.";
          reference
```

```
            "RFC 5424: The Syslog Protocol
             Std-1003.1-2008 Regular Expressions";
        }
      }

      grouping structured-data {
        description
          "This grouping defines the syslog structured data option
           which is used to select the format used to write log
           messages.";
        leaf structured-data {
          if-feature structured-data;
          type boolean;
          default false;
          description
            "This leaf describes how log messages are written.
             If true, messages will be written with one or more
             STRUCTURED-DATA elements; if false, messages will be
             written with STRUCTURED-DATA = NILVALUE.";
          reference
            "RFC 5424: The Syslog Protocol";
        }
      }

      container syslog {
        presence "Enables logging.";
        description
          "This container describes the configuration parameters for
           syslog.";
        container actions {
          description
            "This container describes the log-action parameters
             for syslog.";
          container console {
            if-feature console-action;
            presence "Enables logging to the console";
            description
              "This container describes the configuration parameters
               for console logging.";
            uses selector;
          }
          container file {
            if-feature file-action;
            description
              "This container describes the configuration parameters for
               file logging. If file-archive limits are not supplied, it
               is assumed that the local implementation defined limits
               will be used.";
            list log-file {
              key "name";
              description
                "This list describes a collection of local logging
                 files.";
              leaf name {
```

```
            type inet:uri {
              pattern 'file:.*';
            }
            description
              "This leaf specifies the name of the log file which
               MUST use the uri scheme file:.";
            reference
                "RFC 8089: The file URI Scheme";
          }
          uses selector;
          uses structured-data;
          container file-rotation {
            description
              "This container describes the configuration
               parameters for log file rotation.";
            leaf number-of-files {
              if-feature file-limit-size;
              type uint32;
              default 1;
              description
                "This leaf specifies the maximum number of log
                 files retained. Specify 1 for implementations
                 that only support one log file.";
            }
            leaf max-file-size {
              if-feature file-limit-size;
              type uint32;
              units "megabytes";
              description
                "This leaf specifies the maximum log file size.";
            }
            leaf rollover {
              if-feature file-limit-duration;
              type uint32;
              units "minutes";
              description
                "This leaf specifies the length of time that log
                 events should be written to a specific log file.
                 Log events that arrive after the rollover period
                 cause the current log file to be closed and a new
                 log file to be opened.";
            }
            leaf retention {
              if-feature file-limit-duration;
              type uint32;
              units "minutes";
              description
                "This leaf specifies the length of time that
                 completed/closed log event files should be stored
                 in the file system before they are removed.";
            }
          }
        }
      }
```

```
          container remote {
            if-feature remote-action;
            description
              "This container describes the configuration parameters
               for forwarding syslog messages to remote relays or
               collectors.";
            list destination {
              key "name";
              description
                "This list describes a collection of remote logging
                 destinations.";
              leaf name {
                type string;
                description
                  "An arbitrary name for the endpoint to connect to.";
              }
              choice transport {
                mandatory true;
                description
                  "This choice describes the transport option.";
                case udp {
                  container udp {
                    description
                      "This container describes the UDP transport
                       options.";
                    reference
                      "RFC 5426: Transmission of Syslog Messages over
                       UDP";
                    leaf address {
                      type inet:host;
                      description
                        "The leaf uniquely specifies the address of
                         the remote host. One of the following must be
                         specified: an ipv4 address, an ipv6 address,
                         or a host name.";
                    }
                    leaf port {
                      type inet:port-number;
                      default 514;
                      description
                        "This leaf specifies the port number used to
                         deliver messages to the remote server.";
                    }
                  }
                }
                case tls {
                  container tls {
                    description
                      "This container describes the TLS transport
                       options.";
                    reference
                      "RFC 5425: Transport Layer Security (TLS)
                       Transport Mapping for Syslog ";
                    leaf address {
```

```
                    type inet:host;
                    description
                      "The leaf uniquely specifies the address of
                       the remote host. One of the following must be
                       specified: an ipv4 address, an ipv6 address,
                       or a host name.";
                  }
                  leaf port {
                    type inet:port-number;
                    default 6514;
                    description
                      "TCP port 6514 has been allocated as the default
                       port for syslog over TLS.";
                  }
                  uses tlsc:tls-client-grouping;
                }
              }
            }
            uses selector;
            uses structured-data;
            leaf facility-override {
              type identityref {
                base syslog-facility;
              }
              description
                "If specified, this leaf specifies the facility used
                 to override the facility in messages delivered to
                 the remote server.";
            }
            leaf source-interface {
              if-feature remote-source-interface;
              type if:interface-ref;
              description
                "This leaf sets the source interface to be used to
                 send messages to the remote syslog server. If not
                 set, messages can be sent on any interface.";
            }
            container signing {
              if-feature signed-messages;
              presence
                "If present, syslog-signing options is activated.";
              description
                "This container describes the configuration
                 parameters for signed syslog messages.";
              reference
                "RFC 5848: Signed Syslog Messages";
              container cert-signers {
                description
                  "This container describes the signing certificate
                   configuration for Signature Group 0 which covers
                   the case for administrators who want all Signature
                   Blocks to be sent to a single destination.";
                list cert-signer {
                  key "name";
```

```
                description
                  "This list describes a collection of syslog
                   message signers.";
                leaf name {
                  type string;
                  description
                    "This leaf specifies the name of the syslog
                     message signer.";
                }
                container cert {
                  uses ks:private-key-grouping;
                  uses ks:certificate-grouping;
                  description
                   "This is the certificate that is periodically
                    sent to the remote receiver. Selection of the
                    certificate also implicitly selects the private
                    key used to sign the syslog messages.";
                }
                leaf hash-algorithm {
                  type enumeration {
                    enum SHA1 {
                      value 1;
                      description
                        "This enum describes the SHA1 algorithm.";
                    }
                    enum SHA256 {
                      value 2;
                      description
                        "This enum describes the SHA256 algorithm.";
                    }
                  }
                  description
                    "This leaf describes the syslog signer hash
                     algorithm used.";
                }
              }
              leaf cert-initial-repeat {
                type uint32;
                default 3;
                description
                "This leaf specifies the number of times each
                 Certificate Block should be sent before the first
                 message is sent.";
              }
              leaf cert-resend-delay {
                type uint32;
                units "seconds";
                default 3600;
                description
                  "This leaf specifies the maximum time delay in
                   seconds until resending the Certificate Block.";
              }
              leaf cert-resend-count {
                type uint32;
```

```
                    default 0;
                    description
                      "This leaf specifies the maximum number of other
                       syslog messages to send until resending the
                       Certificate Block.";
                  }
                  leaf sig-max-delay {
                    type uint32;
                    units "seconds";
                    default 60;
                    description
                      "This leaf specifies when to generate a new
                       Signature Block. If this many seconds have
                       elapsed since the message with the first message
                       number of the Signature Block was sent, a new
                       Signature Block should be generated.";
                  }
                  leaf sig-number-resends {
                    type uint32;
                    default 0;
                    description
                      "This leaf specifies the number of times a
                       Signature Block is resent. (It is recommended to
                       select a value of greater than 0 in particular
                       when the UDP transport RFC 5426 is used.).";
                  }
                  leaf sig-resend-delay {
                    type uint32;
                    units "seconds";
                    default 5;
                    description
                      "This leaf specifies when to send the next
                       Signature Block transmission based on time. If
                       this many seconds have elapsed since the previous
                       sending of this Signature Block, resend it.";
                  }
                  leaf sig-resend-count {
                    type uint32;
                    default 0;
                    description
                      "This leaf specifies when to send the next
                       Signature Block transmission based on a count.
                       If this many other syslog messages have been
                       sent since the previous sending of this
                       Signature Block, resend it. A value of 0 means
                       that you don't resend based on the number of
                       messages.";
                  }
                }
              }
            }
          }
        }
      }
    }
```

```
   }
   <CODE ENDS>
```

Figure 3. ietf-syslog Module

4. Usage Examples

```
      Requirement:
      Enable console logging of syslogs of severity critical

      <syslog xmlns="urn:ietf:params:xml:ns:yang:ietf-syslog">
        <actions>
          <console>
            <facility-filter>
              <facility-list>
                <facility>all</facility>
                <severity>critical</severity>
              </facility-list>
            </facility-filter>
          </console>
        </actions>
      </syslog>

      Enable remote logging of syslogs to udp destination
      foo.example.com for facility auth, severity error

      <syslog xmlns="urn:ietf:params:xml:ns:yang:ietf-syslog">
        <actions>
          <remote>
            <destination>
              <name>remote1</name>
              <udp>
                <address>foo.example.com</address>
              </udp>
              <facility-filter>
                <facility-list>
                  <facility>auth</facility>
                  <severity>error</severity>
                </facility-list>
              </facility-filter>
            </destination>
          </remote>
        </actions>
      </syslog>
```

Figure 4. ietf-syslog Examples

5. Acknowledgements

The authors wish to thank the following who commented on this proposal:

Andy Bierman, Martin Bjorklund, Alex Campbell, Alex Clemm, Francis Dupont, Jim Gibson, Jeffrey Haas, Bob Harold, John Heasley, Giles Heron, Lisa Huang, Mahesh Jethanandani, Warren Kumari, Jeffrey K Lange, Jan Lindblad, Chris Lonvick, Alexey Melnikov, Kathleen Moriarty, Tom Petch, Adam Roach, Juergen Schoenwaelder, Phil Shafer, Yaron Sheffer, Jason Sterne, Peter Van Horne, Kent Watsen, Bert Wijnen, Dale R Worley, and Aleksandr Zhdankin.

6. IANA Considerations

6.1. The IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-syslog
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

6.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC7895]. Following the format in [RFC7950], the following registration is requested:

```
name:         ietf-syslog
namespace:    urn:ietf:params:xml:ns:yang:ietf-syslog
prefix:       ietf-syslog
reference:    RFC zzzz
```

7. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC6536] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes should be considered sensitive or vulnerable in all network environments. Logging in particular is used to assess the state of systems and can be used to indicate a network compromise. If logging were to be disabled through malicious means, attacks may not be readily detectable. Therefore write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations and on network security.

In addition there are data nodes that require careful analysis and review. These are the subtrees and data nodes and their sensitivity/ vulnerability:

- facility-filter/pattern-match: When writing this node, implementations MUST ensure that the regular expression pattern match is not constructed to cause a regular expression denial of service attack due to a pattern that causes the regular expression implementation to work very slowly (exponentially related to input size).

- remote/destination/signing/cert-signer: When writing this subtree, implementations MUST NOT specify a private key that is used for any other purpose.

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:

- remote/destination/transport: This subtree contains information about other hosts in the network, and the TLS transport certificate properties if TLS is selected as the transport protocol.

- remote/destination/signing: This subtree contains information about the syslog message signing properties including signing certificate information.

There are no RPC operations defined in this YANG module.

8. References

8.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a "Keystore" Mechanism", Internet-Draft draft-ietf-netconf-keystore-04, October 2017.

[I-D.ietf-netconf-tls-client-server]

Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", Internet-Draft draft-ietf-netconf-tls-client-server-05, October 2017.

[I-D.ietf-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", Internet-Draft draft-ietf-netmod-rfc7223bis-03, January 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <http://www.rfc-editor.org/info/rfc5424>.

[RFC5425] Miao, F., Ed., Ma, Y.Ed., and J. Salowey, Ed., "Transport Layer Security (TLS) Transport Mapping for Syslog", RFC 5425, DOI 10.17487/RFC5425, March 2009, <https://www.rfc-editor.org/info/rfc5425>.

[RFC5426] Okmianski, A., "Transmission of Syslog Messages over UDP", RFC 5426, DOI 10.17487/RFC5426, March 2009, <http://www.rfc-editor.org/info/rfc5426>.

[RFC5848] Kelsey, J., Callas, J. and A. Clemm, "Signed Syslog Messages", RFC 5848, DOI 10.17487/RFC5848, May 2010, <http://www.rfc-editor.org/info/rfc5848>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7895] Bierman, A., Bjorklund, M. and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC8089] Kerwin, M., "The "file" URI Scheme", RFC 8089, DOI 10.17487/RFC8089, February 2017, <https://www.rfc-editor.org/info/rfc8089>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <http://www.rfc-editor.org/info/rfc8174>.

[Std-1003.1-2008]

The Open Group, ""Chapter 9: Regular Expressions". The Open Group Base Specifications Issue 6, IEEE Std 1003.1-2008, 2016 Edition.", September 2016, <http://pubs.opengroup.org/onlinepubs/9699919799/>.

8.2. Informative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K. and R. Wilton, "Network Management Datastore Architecture", Internet-Draft draft-ietf-netmod-revised-datastores-10, January 2018.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", Internet-Draft draft-ietf-netmod-yang-tree-diagrams-06, February 2018.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J.Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC8040] Bierman, A., Bjorklund, M. and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

Appendix A. Implementer Guidelines

Appendix A.1. Extending Facilities

Many vendors extend the list of facilities available for logging in their implementation. Additional facilities may not work with the syslog protocol as defined in [RFC5424] and hence such facilities apply for local syslog-like logging functionality.

The following is an example that shows how additional facilities could be added to the list of available facilities (in this example two facilities are added):

```
   module example-vendor-syslog-types {
     namespace "http://example.com/ns/vendor-syslog-types";
     prefix vendor-syslogtypes;

     import ietf-syslog {
       prefix syslogtypes;
     }

     organization "Example, Inc.";
     contact
       "Example, Inc.
        Customer Service

        E-mail: syslog-yang@example.com";

     description
       "This module contains a collection of vendor-specific YANG type
        definitions for SYSLOG.";

     revision 2017-08-11 {
       description
         "Version 1.0";
       reference
         "Vendor SYSLOG Types: SYSLOG YANG Model";
     }

     identity vendor_specific_type_1 {
       base syslogtypes:syslog-facility;
       description
         "Adding vendor specific type 1 to syslog-facility";
     }

     identity vendor_specific_type_2 {
       base syslogtypes:syslog-facility;
       description
         "Adding vendor specific type 2 to syslog-facility";
     }
   }
```

Appendix A.2. Syslog Terminal Output

Terminal output with requirements more complex than the console subtree currently provides, are expected to be supported via vendor extensions rather than handled via the file subtree.

Appendix A.3. Syslog File Naming Convention

The syslog/file/log-file/file-rotation container contains configuration parameters for syslog file rotation. This section describes how these fields might be used by an implementer to name syslog files in a rotation process. This information is offered as an informative guide only.

When an active syslog file with a name specified by log-file/name, reaches log-file/max-file-size and/or syslog events arrive after the period specified by log-file/rollover, the logging system can close the file, can compress it, and can name the archive file <log-file/name>.0.gz. The logging system can then open a new active syslog file <log-file/name>.

When the new syslog file reaches either of the size limits referenced above, <log-file/name>.0.gz can be renamed <log-file/name>.1.gz and the new syslog file can be closed, compressed and renamed <log-file/name>.0.gz. Each time that a new syslog file is closed, each of the prior syslog archive files named <log-file/name>.<n>.gz can be renamed to <log-file/name>.<n + 1>.gz.

Removal of archive log files could occur when either or both:

- log-file/number-of-files specified - the logging system can create up to log-file/number-of-files syslog archive files after which, the contents of the oldest archived file could be overwritten.

- log-file/retention specified - the logging system can remove those syslog archive files whose file expiration time (file creation time plus the specified log-file/retention time) is prior to the current time.

Authors' Addresses

Clyde Wildes, editor
Cisco Systems Inc.
170 West Tasman Drive
San Jose, CA 95134
US

Phone: +1 408 527-2672
Email: cwildes@cisco.com

Kiran Koushik, editor
Verizon Wireless
500 W Dove Rd.
Southlake, TX 76092
US

Phone: +1 512 650-0210
Email: kirankoushik.agraharasreenivasa@verizonwireless.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 14, 2017

A. Lindem, Ed.
Cisco Systems
L. Berger, Ed.
LabN Consulting, L.L.C.
D. Bogdanovic

C. Hopps
Deutsche Telekom

March 13, 2017

Network Device YANG Logical Organization
draft-ietf-rtgwg-device-model-02

Abstract

This document presents an approach for organizing YANG models in a comprehensive logical structure that may be used to configure and operate network devices. The structure is itself represented as an example YANG model, with all of the related component models logically organized in a way that is operationally intuitive, but this model is not expected to be implemented. The identified component modules are expected to be defined and implemented on common network devices.

This document is derived from work submitted to the IETF by members of the informal OpenConfig working group of network operators and is a product of the Routing Area YANG Architecture design team.

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 September 14, 2017.

Table of Contents

1. Introduction

"Operational Structure and Organization of YANG Models" [I-D.openconfig-netmod-model-structure], highlights the value of organizing individual, self-standing YANG [RFC6020] models into a more comprehensive structure. This document builds on that work and presents a derivative logical structure for use in representing the networking infrastructure aspects of physical and virtual devices. [I-D.openconfig-netmod-model-structure] and earlier versions of this document presented a single device-centric model root, this document no longer contains this element. Such an element would have translated to a single device management model that would be the root

of all other models and was judged to be overly restrictive in terms of definition, implementation, and operation.

The document presents a logical network device YANG organizational structure that provides a conceptual framework for the models that may be used to configure and operate network devices. The structure is itself presented as an example YANG module, with all of the related component modules logically organized in a way that is operationally intuitive. This network device model is not expected to be implemented, but rather provide as context for the identified representative component modules with are expected to be defined, and supported on typical network devices.

This document refers to two new modules that are expected to be implemented. These models are defined to support the configuration and operation of network-devices that allow for the partitioning of resources from both, or either, management and networking perspectives. Two forms of resource partitioning are referenced:

The first form provides a logical partitioning of a network device where each partition is separately managed as essentially an independent network element which is 'hosted' by the base network device. These hosted network elements are referred to as logical network elements, or LNEs, and are supported by the logical-network-element module defined in [I-D.ietf-rtgwg-lne-model]. The module is used to identify LNEs and associate resources from the network-device with each LNE. LNEs themselves are represented in YANG as independent network devices; each accessed independently. Optionally, and when supported by the implementation, they may also be accessed from the host system. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

The second form provides support what is commonly referred to as Virtual Routing and Forwarding (VRF) instances as well as Virtual Switch Instances (VSI), see [RFC4026]. In this form of resource partitioning multiple control plane and forwarding/bridging instances are provided by and managed via a single (physical or logical) network device. This form of resource partitioning is referred to as Network Instances and are supported by the network-instance module defined in [I-D.ietf-rtgwg-ni-model]. Configuration and operation of each network-instance is always via the network device and the network-instance module.

This document was motivated by, and derived from, [I-D.openconfig-netmod-model-structure]. The requirements from that document have been combined with the requirements from "Consistent Modeling of Operational State Data in YANG",

[I-D.openconfig-netmod-opstate], into "NETMOD Operational State Requirements", [I-D.ietf-netmod-opstate-reqs]. This document is aimed at the requirement related to a common model-structure, currently Requirement 7, and also aims to provide a modeling base for Operational State representation.

The approach taken in this (and the original) document is to organize the models describing various aspects of network infrastructure, focusing on devices, their subsystems, and relevant protocols operating at the link and network layers. The proposal does not consider a common model for higher level network services. We focus on the set of models that are commonly used by network operators, and suggest a corresponding organization.

A significant portion of the text and model contained in this document was taken from the -00 of [I-D.openconfig-netmod-model-structure].

1.1. Status of Work and Open Issues

This version of the document and structure are a product of the Routing Area YANG Architecture design team and is very much a work in progress rather than a final proposal. This version is a major change from the prior version and this change was enabled by the work on the previously mentioned Schema Mount.

Schema Mount enables a dramatic simplification of the presented device model, particularly for "lower-end" devices which are unlikely to support multiple network instances or logical network elements. Should structural-mount/YSDL not be available, the more explicit tree structure presented in earlier versions of this document will need to be utilized.

The top open issues are:

1. This document will need to match the evolution and standardization of [I-D.openconfig-netmod-opstate] or [I-D.ietf-netmod-opstate-reqs] by the Netmod WG.

2. Interpretation of different policy containers requires clarification.

3. It may make sense to use the identityref structuring with hardware and QoS model.

4. Which document(s) define the base System management, network services, and oam protocols modules is TBD. This includes the

possibility of simply using RFC7317 in place of the presented System management module.

5. The model will be updated once the "opstate" requirements are addressed.

2. Module Overview

In this document, we consider network devices that support protocols and functions defined within the IETF Routing Area, e.g, routers, firewalls and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs) each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

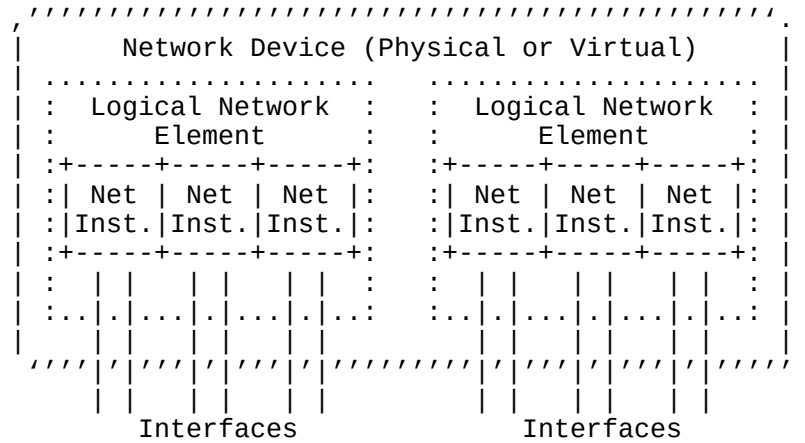


Figure 1: Module Element Relationships

A model for LNEs is described in [I-D.ietf-rtgwg-lne-model] and the model for network instances is covered in [I-D.ietf-rtgwg-ni-model].

The presented notional network device module can itself be thought of as a "meta-model" as it describes the relationships between individual models. We choose to represent it also as a simple YANG module consisting of other models, which are in fact independent top

level individual models. Although it is never expected to be implemented.

The presented modules do not follow the hierarchy of any Particular implementation, and hence is vendor-neutral. Nevertheless, the structure should be familiar to network operators and also readily mapped to vendor implementations.

The overall structure is:

```
    module: example-network-device
       +--rw modules-state             [RFC7895]
       |
       +--rw interfaces                [RFC7223]
       +--rw hardware
       +--rw qos
       |
       +--rw system-management         [RFC7317 or derived]
       +--rw network-services
       +--rw oam-protocols
       |
       +--rw routing                   [I-D.ietf-netmod-routing-cfg]
       +--rw mpls
       +--rw ieee-dot1Q
       |
       +--rw acls                      [I-D.ietf-netmod-acl-model]
       +--rw key-chains                [I-D.ietf-rtgwg-yang-key-chain]
       |
       +--rw logical-network-elements [I-D.ietf-rtgwg-lne-model]
       +--rw network-instances         [I-D.ietf-rtgwg-ni-model]
```

The network device is composed of top level modules that can be used to configure and operate a network device. (This is a significant difference from earlier versions of this document where there was a strict model hierarchy.) Importantly the network device structure is the same for a physical network device or a logical network device, such as those instantiated via the logical-network-element model. Extra spacing is included to denote different types of modules included.

YANG library [RFC7895] is included as it used to identify details of the top level modules supported by the (physical or logical) network device. Th ability to identify supported modules is particularly important for LNEs which may have a set of supported modules which differs from the set supported by the host network device.

The interface management model [RFC7223] is included at the top level. The hardware module is a placeholder for a future device-

specific configuration and operational state data model. For example, a common structure for the hardware model might include chassis, line cards, and ports, but we leave this unspecified. The quality of service (QoS) section is also a placeholder module for device configuration and operational state data which relates to the treatment of traffic across the device. This document references augmentations to the interface module to support LNEs and NIs. Similar elements, although perhaps only for LNEs, may also need to be included as part of the definition of the future hardware and QoS modules.

System management, network services, and oam protocols represent new top level modules that are used to organize data models of similar functions. Additional information on each is provided below.

The routing and MPLS modules provide core support for the configuration and operation of a devices control plane and data plane functions. IEEE dot1Q [IEEE-8021Q] is an example of another module that provides similar functions for VLAN bridging, and other similar modules are also possible. Each of these modules is expected to be LNE and NI unaware, and to be instantiated as needed as part of the LNE and NI configuration and operation supported by the logical-network-element and network-instance modules. (Note that this is a change from [I-D.ietf-netmod-routing-cfg] which is currently defined with VRF/NI semantics.)

The access control list (ACL) and key chain modules are included as examples of other top level modules that may be supported by a network device.

The logical network element and network instance modules enable LNEs and NIs respectively and are defined below.

2.1. Interface Model Components

Interfaces are a crucial part of any network device’s configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [RFC7223].

The logical-network-element and network-instance modules defined in [I-D.ietf-rtgwg-lne-model] and [I-D.ietf-rtgwg-ni-model] augment the existing interface management model in two ways: The first, by the

logical-network-element module, adds an identifier which is used on physical interface types to identify an associated LNE. The second, by the network-instance module, adds a name which is used on interface or sub-interface types to identify an associated network instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [RFC7277].

The interface related augmentations are as follows:

```
module: ietf-logical-network-element
augment /if:interfaces/if:interface:
   +--rw bind-lne-name?   string

module: ietf-network-instance
augment /if:interfaces/if:interface:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv4:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv6:
   +--rw bind-network-instance-name?   string
```

The following is an example of envisioned combined usage. The interfaces container includes a number of commonly used components as examples:

```
            +--rw if:interfaces
            |  +--rw interface* [name]
            |     +--rw name                        string
            |     +--rw lne:bind-lne-name?          string
            |     +--rw ethernet
            |     |  +--rw ni:bind-network-instance-name? string
            |     |  +--rw aggregates
            |     |  +--rw rstp
            |     |  +--rw lldp
            |     |  +--rw ptp
            |     +--rw vlans
            |     +--rw tunnels
            |     +--rw ipv4
            |     |  +--rw ni:bind-network-instance-name? string
            |     |  +--rw arp
            |     |  +--rw icmp
            |     |  +--rw vrrp
            |     |  +--rw dhcp-client
            |     +--rw ipv6
            |        +--rw ni:bind-network-instance-name? string
            |        +--rw vrrp
            |        +--rw icmpv6
            |        +--rw nd
            |        +--rw dhcpv6-client
```

The [RFC7223] defined interface model is structured to include all interfaces in a flat list, without regard to logical or virtual instances (e.g., VRFs) supported on the device. The bind-lne-name and bind-network-instance-name leaves provide the association between an interface and its associated LNE and NI (e.g., VRF or VSI).

2.2. System Management

[Editor’s note: need to discuss and resolve relationship between this structure and RFC7317 and determine if 7317 is close enough to simply use as is.]

System management is expected to reuse definitions contained in [RFC7317]. It is expected to be instantiated per device and LNE. Its structure is shown below:

```
    module: example-network-device
       +--rw system-management
       |  +--rw system-management-global
       |  +--rw system-management-protocol* [type]
       |     +--rw type    identityref
```

System-management-global is used for configuration information and state that is independent of a particular management protocol. System-management-protocol is a list of management protocol specific elements. The type-specific sub-modules are expected to be defined.

The following is an example of envisioned usage:

```
module: example-network-device
   +--rw system-management
      +--rw system-management-global
      |  +--rw statistics-collection
      |  ...
      +--rw system-management-protocol* [type]
      |  +--rw type=syslog
      |  +--rw type=dns
      |  +--rw type=ntp
      |  +--rw type=ssh
      |  +--rw type=tacacs
      |  +--rw type=snmp
      |  +--rw type=netconf
```

2.3. Network Services

A device may provide different network services to other devices, for example a device my act as a DHCP server. The model may be instantiated per device, LNE, and NI. An identityref is used to identify the type of specific service being provided and its associated configuration and state information. The defined structure is as follows:

```
module: example-network-device
   +--rw network-services
   |  +--rw network-service* [type]
   |     +--rw type    identityref
```

The following is an example of envisioned usage: Examples shown below include a device-based Network Time Protocol (NTP) server, a Domain Name System (DNS) server, and a Dynamic Host Configuration Protocol (DHCP) server:

```
module: example-network-device
   +--rw network-services
      +--rw network-service* [type]
         +--rw type=ntp-server
         +--rw type=dns-server
         +--rw type=dhcp-server
```

2.4. OAM Protocols

OAM protocols that may run within the context of a device are grouped within the oam-protocols model. The model may be instantiated per device, LNE, and NI. An identifyref is used to identify the information and state that may relate to a specific OAM protocol. The defined structure is as follows:

```
module: example-network-device
   +--rw oam-protocols
      +--rw oam-protocol* [type]
         +--rw type    identityref
```

The following is an example of envisioned usage. Examples shown below include Bi-directional Forwarding Detection (BFD), Ethernet Connectivity Fault Management (CFM), and Two-Way Active Measurement Protocol (TWAMP):

```
module: example-network-device
   +--rw oam-protocols
      +--rw oam-protocol* [type]
         +--rw type=bfd
         +--rw type=cfm
         +--rw type=twamp
```

2.5. Routing

Routing protocol and IP forwarding configuration and operation information is modeled via a routing model, such as the one defined in [I-D.ietf-netmod-routing-cfg].

The routing module is expected to include all IETF defined control plane protocols, such as BGP, OSPF, LDP and RSVP-TE. It is also expected to support configuration and operation of or more routing information bases (RIB). A RIB is a list of routes complemented with administrative data. Finally, policy is expected to be represented within each control plane protocol and RIB.

The anticipated structure is as follows:

```
module: example-network-device
    +--rw rt:routing                [I-D.ietf-netmod-routing-cfg]
       +--rw control-plane-protocol* [type]
       |  +--rw type      identityref
       |  +--rw policy
       +--rw rib* [name]
          +--rw name           string
          +--rw description?   string
          +--rw policy
```

2.6. MPLS

MPLS data plane related information is grouped together, as with the previously discussed modules, is unaware of VRFs/NIs. The model may be instantiated per device, LNE, and NI. MPLS control plane protocols are expected to be included in Section 2.5. MPLS may reuse and build on [I-D.openconfig-mpls-consolidated-model] or other emerging models and has an anticipated structure as follows:

```
module: example-network-device
     +--rw mpls
        +--rw global
        +--rw lsps* [type]
           +--rw type    identityref
```

Type refers to LSP type, such as static, traffic engineered or routing congruent. The following is an example of such usage:

```
module: example-network-device
     +--rw mpls
        +--rw global
        +--rw lsps* [type]
            +--rw type=static
            +--rw type=constrained-paths
            +--rw type=igp-congruent
```

3. Security Considerations

The network-device model structure described in this document does not define actual configuration and state data, hence it is not directly responsible for security risks.

Each of the component models that provide the corresponding configuration and state data should be considered sensitive from a security standpoint since they generally manipulate aspects of network configurations. Each component model should be carefully evaluated to determine its security risks, along with mitigations to reduce such risks.

LNE portion is TBD

NI portion is TBD

4. IANA Considerations

This YANG model currently uses a temporary ad-hoc namespace. If it is placed or redirected for the standards track, an appropriate namespace URI will be registered in the "IETF XML Registry" [RFC3688]. The YANG structure modules will be registered in the "YANG Module Names" registry [RFC6020].

5. Network Device Model Structure

```
<CODE BEGINS> file "example-network-device@2017-03-13.yang"
module example-network-device {

  yang-version "1";

  // namespace
  namespace "urn:example:network-device";

  prefix "nd";

  // import some basic types

  // meta
  organization "IETF RTG YANG Design Team Collaboration
                with OpenConfig";

  contact
      "Routing Area YANG Architecture Design Team -
       <rtg-dt-yang-arch@ietf.org>";

  description
    "This module describes a model structure for YANG
     configuration and operational state data models. Its intent is
     to describe how individual device protocol and feature models
     fit together and interact.";

  revision "2017-03-13" {
    description
      "IETF Routing YANG Design Team Meta-Model";
    reference "TBD";
  }

  // extension statements
```

```
     // identity statements

     identity oam-protocol-type {
         description
             "Base identity for derivation of OAM protocols";
     }

     identity network-service-type {
         description
             "Base identity for derivation of network services";
     }

      identity system-management-protocol-type {
         description
             "Base identity for derivation of system management
              protocols";
      }

      identity oam-service-type {
         description
             "Base identity for derivation of Operations,
              Administration, and Maintenance (OAM) services.";
      }

      identity control-plane-protocol-type {
         description
             "Base identity for derivation of control-plane protocols";
      }

      identity mpls-lsp-type {
         description
             "Base identity for derivation of MPLS LSP typs";
      }

     // typedef statements

     // grouping statements

     grouping ribs {
       description
         "Routing Information Bases (RIBs) supported by a
          network-instance";
       container ribs {
           description
               "RIBs supported by a network-instance";
           list rib {
               key "name";
               min-elements "1";
```

```
            description
                "Each entry represents a RIB identified by the
               'name' key. All routes in a RIB must belong to the
                same address family.

                For each routing instance, an implementation should
                provide one system-controlled default RIB for each
                supported address family.";
            leaf name {
                type string;
                description
                    "The name of the RIB.";
            }
            reference "draft-ietf-netmod-routing-cfg";
            leaf description {
                type string;
                description
                    "Description of the RIB";
            }
            // Note that there is no list of interfaces within
            container policy {
                description "Policy specific to RIB";
            }
        }
      }
    }

    // top level device definition statements
    container ietf-yang-library {
      description
        "YANG Module Library as defined in
         draft-ietf-netconf-yang-library";
    }

    container interfaces {
      description
       "Interface list as defined by RFC7223/RFC7224";
    }

    container hardware {
      description
        "Hardware / vendor-specific data relevant to the platform.
        This container is an anchor point for platform-specific
        configuration and operational state data.  It may be further
        organized into chassis, line cards, ports, etc.  It is
        expected that vendor or platform-specific augmentations
        would be used to populate this part of the device model";
    }
```

```
    container qos {
      description "QoS features, for example policing, shaping, etc.";
    }

    container system-management {
        description
          "System management for physical or virtual device.";
        container system-management-global {
            description "System management - with reuse of RFC 7317";
        }
        list system-management-protocol {
            key "type";
            leaf type {
                type identityref {
                    base system-management-protocol-type;
                }
                mandatory true;
                description
                    "Syslog, ssh, TACAC+, SNMP, NETCONF, etc.";
            }
            description "List of system management protocol
                         configured for a logical network
                         element.";
        }
    }

    container network-services {
        description
            "Container for list of configured network
             services.";
        list network-service {
            key "type";
            description
                "List of network services configured for a
                 network instance.";
            leaf type {
                type identityref {
                    base network-service-type;
                }
                mandatory true;
                description
                    "The network service type supported within
                     a network instance, e.g., NTP server, DNS
                     server, DHCP server, etc.";
            }
        }
    }
```

```
    container oam-protocols {
        description
            "Container for configured OAM protocols.";
        list oam-protocol {
            key "type";
            leaf type {
                type identityref {
                    base oam-protocol-type;
                }
                mandatory true;
                description
                    "The Operations, Administration, and
                     Maintenance (OAM) protocol type, e.g., BFD,
                     TWAMP, CFM, etc.";
            }
            description
                "List of configured OAM protocols.";
        }
    }

    container routing {
      description
        "The YANG Data Model for Routing Management revised to be
         Network Instance / VRF independent. ";
      // Note that there is no routing or network instance
      list control-plane-protocol {
          key "type";
          description
              "List of control plane protocols configured for
               a network instance.";
          leaf type {
              type identityref {
                  base control-plane-protocol-type;
              }
              mandatory true;
              description
                  "The control plane protocol type, e.g., BGP,
                   OSPF IS-IS, etc";
          }
          container policy {
              description
                  "Protocol specific policy,
                  reusing [RTG-POLICY]";
          }
      }
      list rib {
          key "name";
          description
```

```
                "Each entry represents a RIB identified by the
               'name' key. All routes in a RIB must belong to the
                same address family.

                For each routing instance, an implementation should
                provide one system-controlled default RIB for each
                supported address family.";
            leaf name {
                type string;
                mandatory true;
                description
                    "The name of the RIB.";
            }
            reference "draft-ietf-netmod-routing-cfg";
            leaf description {
                type string;
                description
                    "Description of the RIB";
            }
            // Note that there is no list of interfaces within
            container policy {
                description "Policy specific to RIB";
            }
        }
    }

    container mpls {
        description "MPLS and TE configuration";
        container global {
            description "Global MPLS configuration";
        }
        list lsps {
            key "type";
            description
                "List of LSP types.";
            leaf type {
                type identityref {
                    base mpls-lsp-type;
                }
                mandatory true;
                description
                    "MPLS and Traffic Engineering protocol LSP types,
                     static, LDP/SR (igp-congruent),
                     RSVP TE (constrained-paths) , etc.";
            }
        }
    }
```

```
     container ieee-dot1Q {
       description
         "The YANG Data Model for VLAN bridges as defined by the IEEE";
     }

     container ietf-acl {
       description "Packet Access Control Lists (ACLs) as specified
                      in draft-ietf-netmod-acl-model";
     }

     container ietf-key-chain {
       description "Key chains as specified in
                    draft-ietf-rtgwg-yang-key-chain;";
     }

     container logical-network-element {
       description
         "This module is used to support multiple logical network
          elements on a single physical or virtual system.";
     }

     container network-instance {
       description
         "This module is used to support multiple network instances
          within a single physical or virtual device.  Network
          instances are commonly know as VRFs (virtual routing
          and forwarding) and VSIs (virtual switching instances).";
     }
     // rpc statements

     // notification statements

   }
   <CODE ENDS>
```

6. References

6.1. Normative References

[I-D.ietf-rtgwg-lne-model]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "YANG Logical Network Elements", draft-ietf-rtgwg-lne-model-01 (work in progress), October 2016.

[I-D.ietf-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "YANG Network Instances", draft-ietf-rtgwg-ni-model-01 (work in progress), October 2016.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <http://www.rfc-editor.org/info/rfc4026>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

6.2. Informative References

[I-D.ietf-netmod-acl-model]
Bogdanovic, D., Koushik, K., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-09 (work in progress), October 2016.

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-04 (work in progress), January 2016.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-25 (work in progress), November 2016.

[I-D.ietf-rtgwg-yang-key-chain]
Lindem, A., Qu, Y., Yeung, D., Chen, I., Zhang, Z., and Y. Yang, "Routing Key Chain YANG Data Model", draft-ietf-rtgwg-yang-key-chain-15 (work in progress), February 2017.

[I-D.openconfig-mpls-consolidated-model]
George, J., Fang, L., eric.osborne@level3.com, e., and R. Shakir, "MPLS / TE Model for Service Provider Networks", draft-openconfig-mpls-consolidated-model-02 (work in progress), October 2015.

[I-D.openconfig-netmod-model-structure]
Shaikh, A., Shakir, R., D'Souza, K., and L. Fang, "Operational Structure and Organization of YANG Models", draft-openconfig-netmod-model-structure-00 (work in progress), March 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[IEEE-8021Q]
Holness, M., "IEEE 802.1Q YANG Module Specifications", IEEE-Draft http://www.ieee802.org/1/files/public/docs2015/new-mholness-yang-8021Q-0515-v04.pdf, May 2015.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

Appendix A. Acknowledgments

This document is derived from draft-openconfig-netmod-model-structure-00. We thank the Authors of that document and acknowledge their indirect contribution to this work. The authors include: Anees Shaikh, Rob Shakir, Kevin D'Souza, Luyuan Fang, Qin Wu, Stephane Litkowski and Gang Yan.

This work was discussed in and produced by the Routing Area Yang Architecture design team. Members at the time of writing included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Gang Yan.

The identityref approach was proposed by Mahesh Jethanandani.

The RFC text was produced using Marshall Rose's xml2rfc tool.

Authors' Addresses

Acee Lindem (editor)
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Lou Berger (editor)
LabN Consulting, L.L.C.

Email: lberger@labn.net

Dean Bogdanovic

Email: ivandean@gmail.com

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2018

L. Berger
LabN Consulting, L.L.C.
C. Hopps
Deutsche Telekom
A. Lindem
Cisco Systems
D. Bogdanovic

X. Liu
Jabil

March 19, 2018

YANG Model for Logical Network Elements
draft-ietf-rtgwg-lne-model-10

Abstract

This document defines a logical network element YANG module. This module can be used to manage the logical resource partitioning that may be present on a network device. Examples of common industry terms for logical resource partitioning are Logical Systems or Logical Routers. The YANG model in this document conforms to the Network Management Datastore Architecture as defined in RFCXXXX.

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 https://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 September 20, 2018.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] module to support the creation of logical network elements on a network device. A logical network element (LNE) is an independently managed virtual device made up of resources allocated to it from the host or parent network device. An LNE running on a host network device conceptually parallels a virtual machine running on a host system. Using host-virtualization terminology one could refer to an LNE as a "Guest", and the containing network-device as the "Host". While LNEs may be implemented via host-virtualization technologies this is not a requirement. The YANG model in this document conforms to the Network

Management Datastore Architecture defined in the [I-D.ietf-netmod-revised-datastores].

This document also defines the necessary augmentations for allocating host resources to a given LNE. As the interface management model [I-D.ietf-netmod-rfc7223bis] is the only a module that currently defines host resources, this document currently defines only a single augmentation to cover the assignment of interfaces to an LNE. Future modules that define support for the control of host device resources are expected to, where appropriate, provide parallel support for the assignment of controlled resources to LNEs.

As each LNE is an independently managed device, each will have its own set of YANG modeled data that is independent of the host device and other LNEs. For example, multiple LNEs may all have their own "Tunnel0" interface defined which will not conflict with each other and will not exist in the host's interface model. An LNE will have its own management interfaces possibly including independent instances of netconf/restconf/etc servers to support configuration of their YANG models. As an example of this independence, an implementation may choose to completely rename assigned interfaces, so on the host the assigned interface might be called "Ethernet0/1" while within the LNE it might be called "eth1".

In addition to standard management interfaces, a host device implementation may support accessing LNE configuration and operational YANG models directly from the host system. When supported, such access is accomplished through a yang-schema-mount mount point [I-D.ietf-netmod-schema-mount] under which the root level LNE YANG models may be accessed.

Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

1.1. Terminology

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with terms and concepts of YANG [RFC7950] and YANG Schema Mount [I-D.ietf-netmod-schema-mount].

This document uses the graphical representation of data models defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Overview

In this document, we consider network devices that support protocols and functions defined within the IETF Routing Area, e.g, routers, firewalls, and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs), each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

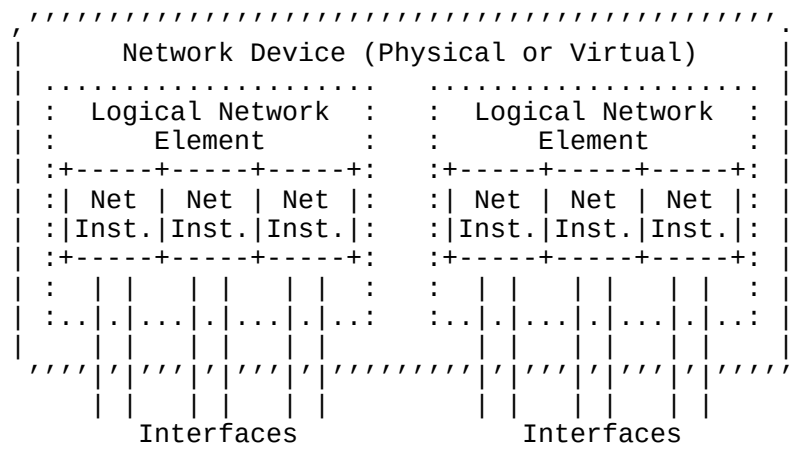


Figure 1: Module Element Relationships

A model for LNEs is described in Section 3 and the model for NIs is covered in [I-D.ietf-rtgwg-ni-model].

The interface management model [I-D.ietf-netmod-rfc7223bis] is an existing model that is impacted by the definition of LNEs and network instances. This document and [I-D.ietf-rtgwg-ni-model] define augmentations to the interface module to support LNEs and NIs. Similar elements, although perhaps only for LNEs, may also need to be included as part of the definition of the future hardware and QoS modules.

Interfaces are a crucial part of any network device’s configuration and operational state. They generally include a combination of raw

physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [I-D.ietf-netmod-rfc7223bis]. The logical-network-element module augments existing interface management model by adding an identifier which is used on interfaces to identify an associated LNE.

The interface related augmentation is as follows:

```
    module: ietf-logical-network-element
        augment /if:interfaces/if:interface:
          +--rw bind-lne-name?   ->
                /logical-network-elements/logical-network-element/name
```

The interface model defined in [I-D.ietf-netmod-rfc7223bis] is structured to include all interfaces in a flat list, without regard to logical assignment of resources supported on the device. The bind-lne-name and leaf provides the association between an interface and its associated LNE. Note that as currently defined, to assign an interface to both an LNE and NI, the interface would first be assigned to the LNE and then within that LNE's interface module, the LNE's representation of that interface would be assigned to an NI using the mechanisms defined in [I-D.ietf-rtgwg-ni-model].

3. Logical Network Elements

Logical network elements support the ability of some devices to partition resources into independent logical routers and/or switches. Device support for multiple logical network elements is implementation specific. Systems without such capabilities need not include support for the logical-network-element module. In physical devices, some hardware features are shared across partitions, but control plane (e.g., routing) protocol instances, tables, and configuration are managed separately. For example, in logical routers or VNFs, this may correspond to establishing multiple logical instances using a single software installation. The model supports configuration of multiple instances on a single device by creating a list of logical network elements, each with their own configuration and operational state related to routing and switching protocols.

The LNE model can be represented as:

```
module: ietf-logical-network-element
    +--rw logical-network-elements
       +--rw logical-network-element* [name]
          +--rw name           string
          +--rw managed?       boolean
          +--rw description?   string
          +--mp root
  augment /if:interfaces/if:interface:
    +--rw bind-lne-name?
            -> /logical-network-elements/logical-network-element/name

  notifications:
    +---n bind-lne-name-failed
       +--ro name             -> /if:interfaces/interface/name
       +--ro bind-lne-name
       |       -> /if:interfaces/interface/lne:bind-lne-name
       +--ro error-info?      string
```

'name' identifies the logical network element. 'managed' indicates if the server providing the host network device will provide the client LNE information via the 'root' structure. The root of an LNE's specific data is the schema mount point 'root'. bind-lne-name is used to associated an interface with an LNE and bind-lne-name-failed is used in certain failure cases.

An LNE root MUST contain at least the YANG library [RFC7895] and Interfaces [I-D.ietf-netmod-rfc7223bis] modules.

3.1. LNE Instantiation and Resource Assignment

Logical network elements may be controlled by clients using existing list operations. When list entries are created, a new LNE is instantiated. The models mounted under an LNE root are expected to be dependent on the server implementation. When a list entry is deleted, an existing LNE is destroyed. For more information, see [RFC7950] Section 7.8.6.

Once instantiated, host network device resources can be associated with the new LNE. As previously mentioned, this document augments ietf-interfaces with the bind-lne-name leaf to support such associations for interfaces. When an bind-lne-name is set to a valid LNE name, an implementation MUST take whatever steps are internally necessary to assign the interface to the LNE or provide an error message (defined below) with an indication of why the assignment failed. It is possible for the assignment to fail while processing the set, or after asynchronous processing. Error notification in the latter case is supported via a notification.

On a successful interface assignment to an LNE, an implementation MUST also make the resource available to the LNE by providing a system created interface to the LNE. The name of the system created interface is a local matter and may be identical or completely different, and mapped from and to, the name used in the context of the host device. The system created interface SHOULD be exposed via the LNE-specific instance of the interfaces module [I-D.ietf-netmod-rfc7223bis].

3.2. LNE Management - LNE View

Each LNE instance is expected to support management functions from within the context of the LNE root, via a server that provides information with the LNE's root exposed as device root. Management functions operating within the context of an LNE are accessed through the LNE's standard management interfaces, e.g., NETCONF and SNMP. Initial configuration, much like the initial configuration of the host device, is a local implementation matter.

When accessing an LNE via the LNE's management interface, a network-device representation will be presented, but its scope will be limited to the specific LNE. Normal YANG/NETCONF mechanisms, together with the required YANG library [RFC7895] instance, can be used to identify the available modules. Each supported module will be presented as a top level module. Only LNE associated resources will be reflected in resource related modules, e.g., interfaces, hardware, and perhaps QoS. From the management perspective, there will be no difference between the available LNE view (information) and a physical network device.

3.3. LNE Management - Host Network Device View

There are multiple implementation approaches possible to enable a network device to support the logical-network-element module and multiple LNEs. Some approaches will allow the management functions operating at network device level to access LNE configuration and operational information, while others will not. Similarly, even when LNE management from the network device is supported by the implementation, it may be prohibited by user policy.

Independent of the method selected by an implementation, the 'managed' boolean mentioned above is used to indicate when LNE management from the network device context is possible. When the 'managed' boolean is 'false', the LNE cannot be managed by the host system and can only be managed from within the context of the LNE as described in the previous section, Section 3.2. Attempts to access information below a root node whose associated 'managed' boolean is set to 'false' MUST result in the error message indicated below. In

some implementations, it may not be possible to change this value. For example, when an LNE is implemented using virtual machine and traditional hypervisor technologies, it is likely that this value will be restricted to a 'false' value.

It is an implementation choice if the information can be accessed and modified from within the context of the LNE, or even the context of the host device. When the 'managed' boolean is 'true', LNE information SHALL be accessible from the context of the host device. When the associated schema-mount definition has the 'config' leaf set to 'true', then LNE information SHALL also be modifiable from the context of the host device. When LNE information is available from both the host device and from within the context of the LNE, the same information MUST be made available via the 'root' element, with paths modified as described in [I-D.ietf-netmod-schema-mount].

An implementation MAY represent an LNE's schema using either the 'inline' or 'shared-schema' approaches defined in [I-D.ietf-netmod-schema-mount]. The choice of which to use is completely an implementation choice. The inline case is anticipated to be generally used in the cases where the 'managed' will always be 'false'. The 'shared-schema' approach is expected to be be most useful in the case where all LNEs share the same schema. When 'shared-schema' is used with an LNE mount point, the YANG library rooted in the LNE's mount point MUST match the associated schema defined according to the ietf-yang-schema-mount module.

Beyond the two modules that will always be present for an LNE, as an LNE is a network device itself, all modules that may be present at the top level network device MAY also be present for the LNE. The list of available modules is expected to be implementation dependent. As is the method used by an implementation to support LNEs. Appendix B provide example uses of LNEs.

4. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

LNE information represents device and network configuration information. As such, the security of this information is important, but it is fundamentally no different than any other interface or device configuration information that has already been covered in other documents such as [I-D.ietf-netmod-rfc7223bis], [RFC7317] and [I-D.ietf-netmod-rfc8022bis].

The vulnerable "config true" parameters and subtrees are the following:

/logical-network-elements/logical-network-element: This list specifies the logical network element and the related logical device configuration.

/logical-network-elements/logical-network-element/managed: While this leaf is contained in the previous list, it is worth particular attention as it controls whether information under the LNE mount point is accessible by both the host device and within the LNE context. There may be extra sensitivity to this leaf in environments where an LNE is managed by a different party than the host device, and that party does not wish to share LNE information with the operator of the host device.

/if:interfaces/if:interface/bind-lne-name: This leaf indicates the LNE instance to which an interface is assigned. Implementations should pay particular attention to when changes to this leaf are permitted as removal of an interface from an LNE can have major impact on the LNEs operation as it is similar to physically removing an interface from the device. Implementations can reject an reassignment using the previously described error message generation.

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

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-logical-network-element

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-logical-network-element
namespace:    urn:ietf:params:xml:ns:yang:ietf-logical-network-element
prefix:       lne
reference:    RFC XXXX
```

6. Logical Network Element Model

The structure of the model defined in this document is described by the YANG module below.

```
<CODE BEGINS> file "ietf-logical-network-element@2018-03-20.yang"
module ietf-logical-network-element {
  yang-version 1.1;

  // namespace

  namespace
     "urn:ietf:params:xml:ns:yang:ietf-logical-network-element";
  prefix lne;

  // import some basic types

  import ietf-interfaces {
    prefix if;
    reference "draft-ietf-netmod-rfc7223bis:
               A YANG Data Model for Interface Management";
  }
  import ietf-yang-schema-mount {
    prefix yangmnt;
    reference "draft-ietf-netmod-schema-mount: YANG Schema Mount";
    // RFC Ed.: Please replace this draft name with the corresponding
    // RFC number
  }

  organization
    "IETF Routing Area (rtgwg) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Author:   Lou Berger
               <mailto:lberger@labn.net>
     Author:   Christan Hopps
               <mailto:chopps@chopps.org>
     Author:   Acee Lindem
```

```
                <mailto:acee@cisco.com>
       Author:  Dean Bogdanovic
                <mailto:ivandean@gmail.com>";
    description
      "This module is used to support multiple logical network
       elements on a single physical or virtual system.

       Copyright (c) 2017 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.: please update TBD

    revision 2018-03-20 {
      description
        "Initial revision.";
      reference "RFC XXXX";
    }

    // top level device definition statements

    container logical-network-elements {
      description
        "Allows a network device to support multiple logical
         network element (device) instances.";
      list logical-network-element {
        key "name";
        description
          "List of logical network elements.";
        leaf name {
          type string;
          description
            "Device-wide unique identifier for the
             logical network element.";
        }
        leaf managed {
          type boolean;
```

```
          default "true";
          description
            "True if the host can access LNE information
             using the root mount point.  This value
             my not be modifiable in all implementations.";
        }
        leaf description {
          type string;
          description
            "Description of the logical network element.";
        }
        container "root" {
          description
            "Container for mount point.";
          yangmnt:mount-point "root" {
            description
              "Root for models supported per logical
               network element.  This mount point may or may not
               be inline based on the server implementation.  It
               SHALL always contain a YANG library and interfaces
               instance.

               When the associated 'managed' leaf is 'false' any
               operation that attempts to access information below
               the root SHALL fail with an error-tag of
               'access-denied' and an error-app-tag of
               'lne-not-managed'.";
          }
        }
      }
    }

    // augment statements

    augment "/if:interfaces/if:interface" {
      description
        "Add a node for the identification of the logical network
         element associated with an interface. Applies to
         interfaces that can be assigned on a per logical network
         element basis.

         Note that a standard error will be returned if the
         identified leafref isn't present.  If an interfaces
         cannot be assigned for any other reason, the operation
         SHALL fail with an error-tag of 'operation-failed' and an
         error-app-tag of 'lne-assignment-failed'.  A meaningful
         error-info that indicates the source of the assignment
         failure SHOULD also be provided.";
```

```
      leaf bind-lne-name {
        type leafref {
          path
           "/logical-network-elements/logical-network-element/name";
        }
        description
          "Logical network element ID to which interface is bound.";
      }
    }

    // notification statements

    notification bind-lne-name-failed {
      description
        "Indicates an error in the association of an interface to an
         LNE. Only generated after success is initially returned when
         bind-lne-name is set.";
      leaf name {
        type leafref {
          path "/if:interfaces/if:interface/if:name";
        }
        mandatory true;
        description
          "Contains the interface name associated with the
           failure.";
      }
      leaf bind-lne-name {
        type leafref {
          path "/if:interfaces/if:interface/lne:bind-lne-name";
        }
        mandatory true;
        description
          "Contains the bind-lne-name associated with the
           failure.";
      }
      leaf error-info {
        type string;
        description
          "Optionally, indicates the source of the assignment
           failure.";
      }
    }
  }
  <CODE ENDS>
```

7. References

7.1. Normative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

[I-D.ietf-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-rfc7223bis-03 (work in progress), January 2018.

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-08 (work in progress), October 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[I-D.ietf-netmod-entity]
Bierman, A., Bjorklund, M., Dong, J., and D. Romascanu, "A YANG Data Model for Hardware Management", draft-ietf-netmod-entity-08 (work in progress), January 2018.

[I-D.ietf-netmod-rfc8022bis]
Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", draft-ietf-netmod-rfc8022bis-11 (work in progress), January 2018.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

[I-D.ietf-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

[I-D.ietf-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Network Instances", draft-ietf-rtgwg-ni-model-11 (work in progress), March 2018.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

Appendix A. Acknowledgments

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Yan Gang. Useful review comments were also received by Martin Bjorklund, John Scudder, Dan Romascanu and Taylor Yu.

This document was motivated by, and derived from, [I-D.ietf-rtgwg-device-model].

The RFC text was produced using Marshall Rose’s xml2rfc tool.

Thanks to Alvaro Retana for IESG review.

Appendix B. Examples

The following subsections provide example uses of LNEs.

B.1. Example: Host Device Managed LNE

This section describes an example of the LNE model using schema mount to achieve the parent management. An example device supports multiple instances of LNEs (logical routers), each of which supports features of layer 2 and layer 3 interfaces [I-D.ietf-netmod-rfc7223bis], routing information base [I-D.ietf-netmod-rfc8022bis], and OSPF protocol. Each of these features is specified by a YANG model, and they are combined using YANG Schema Mount as shown below. Not all possible mounted modules are shown. For example implementations could also mount the model defined in [I-D.ietf-netmod-entity].

```
module: ietf-logical-network-element
   +--rw logical-network-elements
      +--rw logical-network-element* [name]
         +--rw name           string
         +--mp root
            +--ro yanglib:modules-state/
            |  +--ro module-set-id    string
            |  +--ro module* [name revision]
            |     +--ro name                yang:yang-identifier
            +--rw sys:system/
            |  +--rw contact?          string
            |  +--rw hostname?         inet:domain-name
            |  +--rw authentication {authentication}?
            |     +--rw user-authentication-order*   identityref
            |     +--rw user* [name] {local-users}?
            |        +--rw name              string
            |        +--rw password?         ianach:crypt-hash
            |        +--rw authorized-key* [name]
            |           +--rw name         string
            |           +--rw algorithm    string
            |           +--rw key-data     binary
            +--ro sys:system-state/
            |     ...
            +--rw rt:routing/
            |  +--rw router-id?                yang:dotted-quad
            |  +--rw control-plane-protocols
            |     +--rw control-plane-protocol* [type name]
            |        +--rw ospf:ospf/
            |          +--rw areas
```

```
               |             +--rw area* [area-id]
               |                +--rw interfaces
               |                   +--rw interface* [name]
               |                      +--rw name if:interface-ref
               |                      +--rw cost?   uint16
               +--rw if:interfaces/
                  +--rw interface* [name]
                     +--rw name            string
                     +--rw ip:ipv4!/
                     |  +--rw address* [ip]
                     |  ...

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                          string
            +--rw lne:bind-lne-name?            string
            +--ro oper-status        enumeration

   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name         string
      |           +--rw algorithm    string
      |           +--rw key-data     binary
      +--ro system-state
         +--ro platform
         |  +--ro os-name?      string
         |  +--ro os-release?   string
```

To realize the above schema, the example device implements the following schema mount instance:

```
"ietf-yang-schema-mount:schema-mounts": {
  "mount-point": [
    {
      "module": "ietf-logical-network-element",
      "label": "root",
      "shared-schema":  {}
    }
  ]
}
```

By using the implementation of the YANG schema mount, an operator can create instances of logical routers. An interface can be assigned to a logical router, so that the logical router has the permission to access this interface. The OSPF protocol can then be enabled on this assigned interface.

For this implementation, a parent management session has access to the schemas of both the parent hosting system and the child logical routers. In addition, each child logical router can grant its own management sessions, which have the following schema:

```
   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name         string
      |           +--rw algorithm    string
      |           +--rw key-data     binary
      +--ro system-state
         +--ro platform
            +--ro os-name?      string
            +--ro os-release?   string

   module: ietf-routing
      rw-- routing
         +--rw router-id?                 yang:dotted-quad
         +--rw control-plane-protocols
            +--rw control-plane-protocol* [type name]
               +--rw ospf:ospf/
                 +--rw areas
                    +--rw area* [area-id]
                       +--rw interfaces
                          +--rw interface* [name]
                             +--rw name    if:interface-ref
                             +--rw cost?   uint16

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                        string
            +--ro oper-status        enumeration
```

B.1.1. Configuration Data

The following shows an example where two customer specific LNEs are configured:

```
{
  "ietf-logical-network-element:logical-network-elements": {
    "logical-network-element": [
      {
        "name": "cust1",
        "root": {
          "ietf-system:system": {
            "authentication": {
              "user": [
                {
                  "name": "john",
                  "password": "$0$password"
                }
              ]
            }
          },
          "ietf-routing:routing": {
            "router-id": "192.0.2.1",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "af": "ipv4",
                    "areas": {
                      "area": [
                        {
                          "area-id": "203.0.113.1",
                          "interfaces": {
                            "interface": [
                              {
                                "name": "eth1",
                                "cost": 10
                              }
                            ]
                          }
                        }
                      ]
                    }
                  }
                }
              ]
            }
          },
          "ietf-interfaces:interfaces": {
            "interfaces": {
              "interface": [
```

```
                {
                  "name": "eth1",
                  "ip:ipv4": {
                    "address": [
                      {
                        "ip": "192.0.2.11",
                        "prefix-length": 24,
                      }
                    ]
                  },
                  "ip:ipv6": {
                    "address": [
                      {
                        "ip": "2001:db8:0:2::11",
                        "prefix-length": 64,
                      }
                    ]
                  }
                }
              ]
            }
          }
        }
      },
      {
        "name": "cust2",
        "root": {
          "ietf-system:system": {
            "authentication": {
              "user": [
                {
                  "name": "john",
                  "password": "$0$password"
                }
              ]
            }
          }
          "ietf-routing:routing": {
            "router-id": "192.0.2.2",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "af": "ipv4",
                    "areas": {
                      "area": [
```

```
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth1",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            },
            "ietf-interfaces:interfaces": {
              "interfaces": {
                {
                  "name": "eth1",
                  "ip:ipv4": {
                    "address": [
                      {
                        "ip": "192.0.2.11",
                        "prefix-length": 24,
                      }
                    ]
                  }
                }
              ]
            }
          }
        }
      ]
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.10",
                  "prefix-length": 24,
```

```
                }
              ]
            },
            "ip:ipv6": {
              "address": [
                {
                  "ip": "2001:db8:0:2::10",
                  "prefix-length": 64,
                }
              ]
            }
          },
          {
            "name": "cust1:eth1",
            "lne:bind-lne-name": "cust1"
          },
          {
            "name": "cust2:eth1",
            "lne:bind-lne-name": "cust2"
          }
        ]
      }
    },

    "ietf-system:system": {
      "authentication": {
        "user": [
          {
            "name": "root",
            "password": "$0$password"
          }
        ]
      }
    }
  }
```

B.1.2. State Data

The following shows possible state data associated the above configuration data:

```
  {
    "ietf-logical-network-element:logical-network-elements": {
      "logical-network-element": [
        {
          "name": "cust1",
          "root": {
            "ietf-yang-library:modules-state": {
```

```
              "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
              "module": [
                {
                  "name": "iana-if-type",
                  "revision": "2014-05-08",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:iana-if-type",
                  "conformance-type": "import"
                },
                {
                  "name": "ietf-inet-types",
                  "revision": "2013-07-15",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-inet-types",
                  "conformance-type": "import"
                },
                {
                  "name": "ietf-interfaces",
                  "revision": "2014-05-08",
                  "feature": [
                    "arbitrary-names",
                    "pre-provisioning"
                  ],
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-interfaces",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ip",
                  "revision": "2014-06-16",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ip",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ospf",
                  "revision": "2018-03-03",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ospf",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-routing",
                  "revision": "2018-03-13",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-routing",
                  "conformance-type": "implement"
                },
```

```
          {
            "name": "ietf-system",
            "revision": "2014-08-06",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-system",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-yang-library",
            "revision": "2016-06-21",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-yang-library",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-yang-types",
            "revision": "2013-07-15",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-yang-types",
            "conformance-type": "import"
          }
        ]
      },
      "ietf-system:system-state": {
        "ietf-system:system-state": {
          "platform": {
            "os-name": "NetworkOS"
          }
        }
      },
      "ietf-routing:routing": {
        "router-id": "192.0.2.1",
        "control-plane-protocols": {
          "control-plane-protocol": [
            {
              "type": "ietf-routing:ospf",
              "name": "1",
              "ietf-ospf:ospf": {
                "af": "ipv4",
                "areas": {
                  "area": [
                    {
                      "area-id": "203.0.113.1",
                      "interfaces": {
                        "interface": [
                          {
                            "name": "eth1",
                            "cost": 10
```

```
                                  }
                                ]
                              }
                            }
                          ]
                        }
                      }
                    }
                  ]
                }
              },
              "ietf-interfaces:interfaces": {
                "interfaces": {
                  "interface": [
                    {
                      "name": "eth1",
                      "type": "iana-if-type:ethernetCsmacd",
                      "oper-status": "up",
                      "phys-address": "00:01:02:A1:B1:C1",
                      "statistics": {
                        "discontinuity-time": "2017-06-26T12:34:56-05:00"
                      },
                      "ip:ipv4": {
                        "address": [
                          {
                            "ip": "192.0.2.11",
                            "prefix-length": 24,
                          }
                        ]
                      }
                    }
                  ]
                }
              }
            },
            {
              "name": "cust2",
              "root": {
                "ietf-yang-library:modules-state": {
                  "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
                  "module": [
                    {
                      "name": "iana-if-type",
                      "revision": "2014-05-08",
                      "namespace":
                      "urn:ietf:params:xml:ns:yang:iana-if-type",
                      "conformance-type": "import"
```

```
                },
                {
                  "name": "ietf-inet-types",
                  "revision": "2013-07-15",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-inet-types",
                  "conformance-type": "import"
                },
                {
                  "name": "ietf-interfaces",
                  "revision": "2014-05-08",
                  "feature": [
                    "arbitrary-names",
                    "pre-provisioning"
                  ],
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-interfaces",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ip",
                  "revision": "2014-06-16",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ip",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ospf",
                  "revision": "2018-03-03",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ospf",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-routing",
                  "revision": "2018-03-13",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-routing",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-system",
                  "revision": "2014-08-06",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-system",
                  "conformance-type": "implement"
                },
                {
```

```
                "name": "ietf-yang-library",
                "revision": "2016-06-21",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                "conformance-type": "implement"
              },
              {
                "name": "ietf-yang-types",
                "revision": "2013-07-15",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-yang-types",
                "conformance-type": "import"
              }
            ]
          }
          "ietf-system:system-state": {
            "ietf-system:system-state": {
              "platform": {
                "os-name": "NetworkOS"
              }
            }
          },
          "ietf-routing:routing": {
            "router-id": "192.0.2.2",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "af": "ipv4",
                    "areas": {
                      "area": [
                        {
                          "area-id": "203.0.113.1",
                          "interfaces": {
                            "interface": [
                              {
                                "name": "eth1",
                                "cost": 10
                              }
                            ]
                          }
                        }
                      ]
                    }
                  }
                }
```

```
              ]
            }
          }
          "ietf-interfaces:interfaces": {
            "interfaces": {
              {
                "name": "eth1",
                "type": "iana-if-type:ethernetCsmacd",
                "oper-status": "up",
                "phys-address": "00:01:02:A1:B1:C2",
                "statistics": {
                  "discontinuity-time": "2017-06-26T12:34:56-05:00"
                },
                "ip:ipv4": {
                  "address": [
                    {
                      "ip": "192.0.2.11",
                      "prefix-length": 24,
                    }
                  ]
                }
              }
            ]
          }
        }
      }
    ]
  },

  "ietf-interfaces:interfaces": {
    "interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C0",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.10",
                "prefix-length": 24,
              }
            ]
          }
```

```
          },
          {
            "name": "cust1:eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C1",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
          },
          {
            "name": "cust2:eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
          }
        ]
      }
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-yang-library:modules-state": {
      "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
      "module": [
        {
          "name": "iana-if-type",
          "revision": "2014-05-08",
          "namespace":
          "urn:ietf:params:xml:ns:yang:iana-if-type",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "conformance-type": "import"
        },
        {
          "name": "ietf-interfaces",
```

```
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-logical-network-element",
          "revision": "2017-03-13",
          "feature": [
            "bind-lne-name"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-logical-network-element",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ospf",
          "revision": "2018-03-03",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ospf",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
```

```
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-logical-network-element",
          "label": "root",
          "shared-schema": {}
        }
      ]
    }
  }
```

B.2. Example: Self Managed LNE

This section describes an example of the LNE model using schema mount to achieve child independent management. An example device supports multiple instances of LNEs (logical routers), each of them has the features of layer 2 and layer 3 interfaces [I-D.ietf-netmod-rfc7223bis], routing information base [I-D.ietf-netmod-rfc8022bis], and the OSPF protocol. Each of these features is specified by a YANG model, and they are put together by the YANG Schema Mount as following:

```
         module: ietf-logical-network-element
      +--rw logical-network-elements
         +--rw logical-network-element* [name]
            +--rw name           string
            +--mp root
               // The internal modules of the LNE are not visible to
               // the parament management.
               // The child manages its modules, including ietf-routing
               // and ietf-interfaces

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                          string
            +--rw lne:bind-lne-name?            string
            +--ro oper-status         enumeration

   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name          string
      |           +--rw algorithm     string
      |           +--rw key-data      binary
      +--ro system-state
         +--ro platform
         |  +--ro os-name?      string
         |  +--ro os-release?   string
```

To realize the above schema, the device implements the following schema mount instance:

```
"ietf-yang-schema-mount:schema-mounts": {
  "mount-point": [
    {
      "module": "ietf-logical-network-element",
      "label": "root",
      "inline": {}
    }
  ]
}
```

By using the implementation of the YANG schema mount, an operator can create instances of logical routers, each with their logical router specific in-line modules. An interface can be assigned to a logical router, so that the logical router has the permission to access this interface. The OSPF protocol can then be enabled on this assigned interface. Each logical router independently manages its own set of modules, which may or may not be the same as other logical routers. The following is an example of schema set implemented on one particular logical router:

```
   module: ietf-yang-library
      +--ro modules-state
         +--ro module-set-id    string
         +--ro module* [name revision]
            +--ro name                yang:yang-identifier

   module: ietf-system
      +--rw system
      |  +--rw contact?          string
      |  +--rw hostname?         inet:domain-name
      |  +--rw authentication {authentication}?
      |     +--rw user-authentication-order*   identityref
      |     +--rw user* [name] {local-users}?
      |        +--rw name              string
      |        +--rw password?         ianach:crypt-hash
      |        +--rw authorized-key* [name]
      |           +--rw name         string
      |           +--rw algorithm    string
      |           +--rw key-data     binary
      +--ro system-state
         +--ro platform
         |  +--ro os-name?      string
         |  +--ro os-release?   string

   module: ietf-routing
      +--rw routing
         +--rw router-id?                 yang:dotted-quad
         +--rw control-plane-protocols
            +--rw control-plane-protocol* [type name]
               +--rw ospf:ospf/
                  +--rw areas
                    +--rw area* [area-id]
                       +--rw interfaces
                          +--rw interface* [name]
                             +--rw name    if:interface-ref
                             +--rw cost?   uint16

   module: ietf-interfaces
      +--rw interfaces
         +--rw interface* [name]
            +--rw name                        string
            +--ro oper-status        enumeration
```

B.2.1. Configuration Data

Each of the child virtual routers is managed through its own sessions and configuration data.

B.2.1.1. Logical Network Element 'vnf1'

The following shows an example configuration data for the LNE name "vnf1":

```
{
  "ietf-system:system": {
    "authentication": {
      "user": [
        {
          "name": "john",
          "password": "$0$password"
        }
      ]
    }
  },
  "ietf-routing:routing": {
    "router-id": "192.0.2.1",
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "ietf-routing:ospf",
          "name": "1",
          "ietf-ospf:ospf": {
            "af": "ipv4",
            "areas": {
              "area": [
                {
                  "area-id": "203.0.113.1",
                  "interfaces": {
                    "interface": [
                      {
                        "name": "eth1",
                        "cost": 10
                      }
                    ]
                  }
                }
              ]
            }
          }
        }
      ]
    }
  },
  "ietf-interfaces:interfaces": {
    "interfaces": {
      "interface": [
```

```
          {
            "name": "eth1",
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          }
        ]
      }
    }
  }
```

B.2.1.2. Logical Network Element 'vnf2'

The following shows an example configuration data for the LNE name "vnf2":

```
{
  "ietf-system:system": {
    "authentication": {
      "user": [
        {
          "name": "john",
          "password": "$0$password"
        }
      ]
    }
  },
  "ietf-routing:routing": {
    "router-id": "192.0.2.2",
    "control-plane-protocols": {
      "control-plane-protocol": [
        {
          "type": "ietf-routing:ospf",
          "name": "1",
          "ietf-ospf:ospf": {
            "af": "ipv4",
            "areas": {
              "area": [
                {
                  "area-id": "203.0.113.1",
                  "interfaces": {
                    "interface": [
                      {
```

```
                          "name": "eth1",
                          "cost": 10
                        }
                      ]
                    }
                  }
                ]
              }
            }
          ]
        }
      },
      "ietf-interfaces:interfaces": {
    "interfaces": {
          "interface": [
            {
              "name": "eth1",
              "ip:ipv4": {
                "address": [
                  {
                    "ip": "192.0.2.11",
                    "prefix-length": 24,
                  }
                ]
              }
            }
          ]
        }
      }
    }
```

B.2.2. State Data

The following sections shows possible state data associated the above configuration data. Note that there are three views: the host device's, and each LNE's.

B.2.2.1. Host Device

The following shows state data for the device hosting the example LNEs:

```
{
  "ietf-logical-network-element:logical-network-elements": {
    "logical-network-element": [
      {
        "name": "vnf1",
```

```
          "root": {
          }
        },
        {
          "name": "vnf2",
          "root": {
          }
        }
      ]
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C0",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.10",
                  "prefix-length": 24,
                }
              ]
            }
          },
          {
            "name": "vnf1:eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C1",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
          },
          {
            "name": "vnf2:eth2",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            }
```

```
          }
        ]
      }
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-yang-library:modules-state": {
      "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
      "module": [
        {
          "name": "iana-if-type",
          "revision": "2014-05-08",
          "namespace":
          "urn:ietf:params:xml:ns:yang:iana-if-type",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "conformance-type": "import"
        },
        {
          "name": "ietf-interfaces",
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-logical-network-element",
```

```
          "revision": "2017-03-13",
          "feature": [
            "bind-lne-name"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-logical-network-element",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-logical-network-element",
          "label": "root",
          "inline": {}
        }
      ]
    }
```

```
   }
```

B.2.2.2. Logical Network Element 'vnf1'

The following shows state data for the example LNE with name "vnf1":

```
   {
     "ietf-yang-library:modules-state": {
       "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
       "module": [
         {
           "name": "iana-if-type",
           "revision": "2014-05-08",
           "namespace":
           "urn:ietf:params:xml:ns:yang:iana-if-type",
           "conformance-type": "import"
         },
         {
           "name": "ietf-inet-types",
           "revision": "2013-07-15",
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-inet-types",
           "conformance-type": "import"
         },
         {
           "name": "ietf-interfaces",
           "revision": "2014-05-08",
           "feature": [
             "arbitrary-names",
             "pre-provisioning"
           ],
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-interfaces",
           "conformance-type": "implement"
         },
         {
           "name": "ietf-ip",
           "revision": "2014-06-16",
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-ip",
           "conformance-type": "implement"
         },
         {
           "name": "ietf-ospf",
           "revision": "2018-03-03",
           "namespace":
           "urn:ietf:params:xml:ns:yang:ietf-ospf",
           "conformance-type": "implement"
```

```
        },
        {
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-routing:routing": {
      "router-id": "192.0.2.1",
      "control-plane-protocols": {
        "control-plane-protocol": [
          {
            "type": "ietf-routing:ospf",
            "name": "1",
            "ietf-ospf:ospf": {
              "af": "ipv4",
              "areas": {
```

```
                "area": [
                  {
                    "area-id": "203.0.113.1",
                    "interfaces": {
                      "interface": [
                        {
                          "name": "eth1",
                          "cost": 10
                        }
                      ]
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C1",
            "statistics": {
              "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          }
        ]
      }
    }
  }
```

B.2.2.3. Logical Network Element 'vnf2'

The following shows state data for the example LNE with name "vnf2":

```
{
  "ietf-yang-library:modules-state": {
    "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
    "module": [
      {
        "name": "iana-if-type",
        "revision": "2014-05-08",
        "namespace":
        "urn:ietf:params:xml:ns:yang:iana-if-type",
        "conformance-type": "import"
      },
      {
        "name": "ietf-inet-types",
        "revision": "2013-07-15",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-inet-types",
        "conformance-type": "import"
      },
      {
        "name": "ietf-interfaces",
        "revision": "2014-05-08",
        "feature": [
          "arbitrary-names",
          "pre-provisioning"
        ],
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-interfaces",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ip",
        "revision": "2014-06-16",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-ip",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ospf",
        "revision": "2018-03-03",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-ospf",
        "conformance-type": "implement"
      },
      {
```

```
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    },

    "ietf-routing:routing": {
      "router-id": "192.0.2.2",
      "control-plane-protocols": {
        "control-plane-protocol": [
          {
            "type": "ietf-routing:ospf",
            "name": "1",
            "ietf-ospf:ospf": {
              "af": "ipv4",
              "areas": {
                "area": [
                  {
```

```
                    "area-id": "203.0.113.1",
                    "interfaces": {
                      "interface": [
                        {
                          "name": "eth1",
                          "cost": 10
                        }
                      ]
                    }
                  }
                ]
              }
            }
          }
        ]
      }
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
              "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          }
        ]
      }
    }
  }
```

Authors' Addresses

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Dean Bogdanovic

Email: ivandean@gmail.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2018

L. Berger
LabN Consulting, L.L.C.
C. Hopps
Deutsche Telekom
A. Lindem
Cisco Systems
D. Bogdanovic

X. Liu
Jabil

March 19, 2018

YANG Model for Network Instances
draft-ietf-rtgwg-ni-model-12

Abstract

This document defines a network instance module. This module can be used to manage the virtual resource partitioning that may be present on a network device. Examples of common industry terms for virtual resource partitioning are Virtual Routing and Forwarding (VRF) instances and Virtual Switch Instances (VSIs).

The YANG model in this document conforms to the Network Management Datastore Architecture defined in I-D.ietf-netmod-revised-datastores.

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 https://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 September 20, 2018.

Table of Contents

1. Introduction

This document defines the second of two new modules that are defined to support the configuration and operation of network-devices that allow for the partitioning of resources from both, or either, management and networking perspectives. Both leverage the YANG functionality enabled by YANG Schema Mount [I-D.ietf-netmod-schema-mount].

The YANG model in this document conforms to the Network Management Datastore Architecture defined in the [I-D.ietf-netmod-revised-datastores].

The first form of resource partitioning provides a logical partitioning of a network device where each partition is separately managed as essentially an independent network element which is 'hosted' by the base network device. These hosted network elements are referred to as logical network elements, or LNEs, and are supported by the logical-network-element module defined in [I-D.ietf-rtgwg-lne-model]. That module is used to identify LNEs and associate resources from the network-device with each LNE. LNEs themselves are represented in YANG as independent network devices; each accessed independently. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

The second form, which is defined in this document, provides support for what is commonly referred to as Virtual Routing and Forwarding (VRF) instances as well as Virtual Switch Instances (VSI), see [RFC4026] and [RFC4664]. In this form of resource partitioning, multiple control plane and forwarding/bridging instances are provided by and managed via a single (physical or logical) network device. This form of resource partitioning is referred to as a Network Instance and is supported by the network-instance module defined below. Configuration and operation of each network-instance is always via the network device and the network-instance module.

One notable difference between the LNE model and the NI model is that the NI model provides a framework for VRF and VSI management. This document envisions the separate definition of VRF and VSI, i.e., L3 and L2 VPN, technology specific models. An example of such can be found in the emerging L3VPN model defined in [I-D.ietf-bess-l3vpn-yang] and the examples discussed below.

1.1. Terminology

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with terms and concepts of YANG [RFC7950] and YANG Schema Mount [I-D.ietf-netmod-schema-mount].

This document uses the graphical representation of data models defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Overview

In this document, we consider network devices that support protocols and functions defined within the IETF, e.g, routers, firewalls, and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs) each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a network instances (NIs). This breakdown is represented in Figure 1.

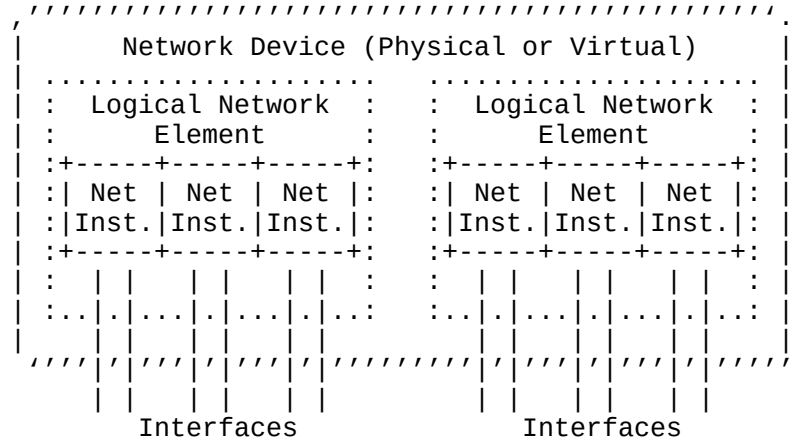


Figure 1: Module Element Relationships

A model for LNEs is described in [I-D.ietf-rtgwg-lne-model] and the model for NIs is covered in this document in Section 3.

The current interface management model [I-D.ietf-netmod-rfc7223bis] is impacted by the definition of LNEs and NIs. This document and [I-D.ietf-rtgwg-lne-model] define augmentations to the interface module to support LNEs and NIs.

The network instance model supports the configuration of VRFs and VSIs. Each instance is supported by information that relates to the device, for example the route target used when advertising VRF routes via the mechanisms defined in [RFC4364], and information that relates to the internal operation of the NI, for example for routing protocols [I-D.ietf-netmod-rfc8022bis] and OSPF [I-D.ietf-ospf-yang].

This document defines the network-instance module that provides a basis for the management of both types of information.

NI information that relates to the device, including the assignment of interfaces to NIs, is defined as part of this document. The defined module also provides a placeholder for the definition of NI-technology specific information both at the device level and for NI internal operation. Information related to NI internal operation is supported via schema mount [I-D.ietf-netmod-schema-mount] and mounting appropriate modules under the mount point. Well known mount points are defined for L3VPN, L2VPN, and L2+L3VPN NI types.

3. Network Instances

The network instance container is used to represent virtual routing and forwarding instances (VRFs) and virtual switching instances (VSIs). VRFs and VSIs are commonly used to isolate routing and switching domains, for example to create virtual private networks, each with their own active protocols and routing/switching policies. The model supports both core/provider and virtual instances. Core/provider instance information is accessible at the top level of the server, while virtual instance information is accessible under the root schema mount points.

```
module: ietf-network-instance
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name           string
        +--rw enabled?       boolean
        +--rw description?   string
        +--rw (ni-type)?
        +--rw (root-type)
           +--:(vrf-root)
           |  +--mp vrf-root
           +--:(vsi-root)
           |  +--mp vsi-root
           +--:(vv-root)
              +--mp vv-root
augment /if:interfaces/if:interface:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name
augment /if:interfaces/if:interface/ip:ipv4:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name
augment /if:interfaces/if:interface/ip:ipv6:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name

notifications:
  +---n bind-ni-name-failed
     +--ro name          -> /if:interfaces/interface/name
     +--ro interface
     |  +--ro bind-ni-name?
     |                   -> /if:interfaces/interface/ni:bind-ni-name
     +--ro ipv4
     |  +--ro bind-ni-name?
     |          -> /if:interfaces/interface/ip:ipv4/ni:bind-ni-name
     +--ro ipv6
     |  +--ro bind-ni-name?
     |          -> /if:interfaces/interface/ip:ipv6/ni:bind-ni-name
     +--ro error-info?   string
```

A network instance is identified by a 'name' string. This string is used both as an index within the network-instance module and to associate resources with a network instance as shown above in the interface augmentation. The ni-type and root-type choice statements are used to support different types of L2 and L3 VPN technologies. The bind-ni-name-failed notification is used in certain failure cases.

3.1. NI Types and Mount Points

The network-instance module is structured to facilitate the definition of information models for specific types of VRFs and VSIs using augmentations. For example, the information needed to support

VPLS, VxLAN and EVPN based L2VPNs are likely to be quite different. Example models under development that could be restructured to take advantage on NIs include, for L3VPNs [I-D.ietf-bess-l3vpn-yang] and for L2VPNs [I-D.ietf-bess-l2vpn-yang].

Documents defining new YANG models for the support of specific types of network instances should augment the network instance module. The basic structure that should be used for such augmentations include a case statement, with containers for configuration and state data and finally, when needed, a type specific mount point. Generally ni types, are expected to not need to define type specific mount points, but rather reuse one of the well known mount point, as defined in the next section. The following is an example type specific augmentation:

```
 augment "/ni:network-instances/ni:network-instance/ni:ni-type" {
   case l3vpn {
     container l3vpn {
         ...
     }
     container l3vpn-state {
         ...
     }
   }
 }
```

3.1.1. Well Known Mount Points

YANG Schema Mount, [I-D.ietf-netmod-schema-mount], identifies mount points by name within a module. This definition allows for the definition of mount points whose schema can be shared across ni-types. As discussed above, ni-types largely differ in the configuration information needed in the core/top level instance to support the NI, rather than in the information represented within an NI. This allows the use of shared mount points across certain NI types.

The expectation is that there are actually very few different schema that need to be defined to support NIs on an implementation. In particular, it is expected that the following three forms of NI schema are needed, and each can be defined with a well known mount point that can be reused by future modules defining ni-types.

The three well known mount points are:

vrf-root
: vrf-root is intended for use with L3VPN type ni-types.

vsi-root
vsi-root is intended for use with L2VPN type ni-types.

vv-root
vv-root is intended for use with ni-types that simultaneously support L2VPN bridging and L3VPN routing capabilities.

Future model definitions should use the above mount points whenever possible. When a well known mount point isn't appropriate, a model may define a type specific mount point via augmentation.

3.1.2. NI Type Example

The following is an example of an L3VPN VRF using a hypothetical augmentation to the networking instance schema defined in [I-D.ietf-bess-l3vpn-yang]. More detailed examples can be found in Appendix B.

```
module: ietf-network-instance
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name           string
        +--rw enabled?       boolean
        +--rw description?   string
        +--rw (ni-type)?
        |  +--:(l3vpn)
        |     +--rw l3vpn:l3vpn
        |     |  ... // config data
        |     +--ro l3vpn:l3vpn-state
        |     |  ... // state data
        +--rw (root-type)
           +--:(vrf-root)
              +--mp vrf-root
                 +--rw rt:routing/
                 |  +--rw router-id?                 yang:dotted-quad
                 |  +--rw control-plane-protocols
                 |     +--rw control-plane-protocol* [type name]
                 |     +--rw ospf:ospf/
                 |          +--rw area* [area-id]
                 |             +--rw interfaces
                 |                +--rw interface* [name]
                 |                   +--rw name if:interface-ref
                 |                   +--rw cost?   uint16
                 +--ro if:interfaces@
                 |  ...
```

This shows YANG Routing Management [I-D.ietf-netmod-rfc8022bis] and YANG OSPF [I-D.ietf-ospf-yang] as mounted modules. The mounted

modules can reference interface information via a parent-reference to the containers defined in [I-D.ietf-netmod-rfc7223bis].

3.2. NIs and Interfaces

Interfaces are a crucial part of any network device's configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity). The interface management model is defined by [I-D.ietf-netmod-rfc7223bis].

As shown below, the network-instance module augments the existing interface management model by adding a name which is used on interface or sub-interface types to identify an associated network instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [I-D.ietf-netmod-rfc7277bis].

The following is an example of envisioned usage. The interfaces container includes a number of commonly used components as examples:

```
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name                          string
   |     +--rw ip:ipv4!
   |     |  +--rw ip:enabled?                      boolean
   |     |  +--rw ip:forwarding?                   boolean
   |     |  +--rw ip:mtu?                          uint16
   |     |  +--rw ip:address* [ip]
   |     |  |  +--rw ip:ip              inet:ipv4-address-no-zone
   |     |  |  +--rw (ip:subnet)
   |     |  |     +--:(ip:prefix-length)
   |     |  |     |  +--rw ip:prefix-length?   uint8
   |     |  |     +--:(ip:netmask)
   |     |  |        +--rw ip:netmask?         yang:dotted-quad
   |     |  +--rw ip:neighbor* [ip]
   |     |  |  +--rw ip:ip                   inet:ipv4-address-no-zone
   |     |  |  +--rw ip:link-layer-address yang:phys-address
   |     |  +--rw ni:bind-network-instance-name?   string
   |     +--rw ni:bind-network-instance-name?   string
```

The [I-D.ietf-netmod-rfc7223bis] defined interface model is structured to include all interfaces in a flat list, without regard to virtual instances (e.g., VRFs) supported on the device. The bind-

network-instance-name leaf provides the association between an interface and its associated NI (e.g., VRF or VSI). Note that as currently defined, to assign an interface to both an LNE and NI, the interface would first be assigned to the LNE using the mechanisms defined in [I-D.ietf-rtgwg-lne-model] and then within that LNE's interface module, the LNE's representation of that interface would be assigned to an NI.

3.3. Network Instance Management

Modules that may be used to represent network instance information will be available under the ni-type specific 'root' mount point. The "shared-schema" method defined in the "ietf-yang-schema-mount" module [I-D.ietf-netmod-schema-mount] MUST be used to identify accessible modules. A future version of this document could relax this requirement. Mounted modules SHOULD be defined with access, via the appropriate schema mount parent-references [I-D.ietf-netmod-schema-mount], to device resources such as interfaces. An implementation MAY choose to restrict parent referenced information to information related to a specific instance, e.g., only allowing references to interfaces that have a "bind-network-instance-name" which is identical to the instance's "name".

All modules that represent control-plane and data-plane information may be present at the 'root' mount point, and be accessible via paths modified per [I-D.ietf-netmod-schema-mount]. The list of available modules is expected to be implementation dependent, as is the method used by an implementation to support NIs.

For example, the following could be used to define the data organization of the example NI shown in Section 3.1.2:

```
  "ietf-yang-schema-mount:schema-mounts": {
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "vrf-root",
        "shared-schema": {
          "parent-reference": [
            "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
          ]
        }
      }
    ]
  }
```

Module data identified according to the ietf-yang-schema-mount module will be instantiated under the mount point identified under "mount-

point". These modules will be able to reference information for nodes belonging to top-level modules that are identified under "parent-reference". Parent referenced information is available to clients via their top level paths only, and not under the associated mount point.

To allow a client to understand the previously mentioned instance restrictions on parent referenced information, an implementation MAY represent such restrictions in the "parent-reference" leaf-list. For example:

```
  "namespace": [
    {
      "prefix": "if",
      "uri": "urn:ietf:params:xml:ns:yang:ietf-interfaces"
    },
    {
      "prefix": "ni",
      "uri": "urn:ietf:params:xml:ns:yang:ietf-network-instance"
    }
  ],
  "mount-point": [
    {
      "module": "ietf-network-instance",
      "label": "vrf-root",
      "shared-schema": {
        "parent-reference": [
          "/if:interfaces/if:interface
             [ni:bind-network-instance-name = current()/../ni:name]",
          "/if:interfaces/if:interface/ip:ipv4
             [ni:bind-network-instance-name = current()/../ni:name]",
          "/if:interfaces/if:interface/ip:ipv6
             [ni:bind-network-instance-name = current()/../ni:name]"
        ]
      }
    }
  ],
```

The same such "parent-reference" restrictions for non-NMDA implementations can be represented based on the [RFC7223] and [RFC7277] as:

```
    "namespace": [
      {
        "prefix": "if",
        "uri": "urn:ietf:params:xml:ns:yang:ietf-interfaces"
      },
      {
        "prefix": "ni",
        "uri": "urn:ietf:params:xml:ns:yang:ietf-network-instance"
      }
    ],
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "vrf-root",
        "shared-schema": {
          "parent-reference": [
            "/if:interfaces/if:interface
               [ni:bind-network-instance-name = current()/../ni:name]",
            "/if:interfaces-state/if:interface
               [if:name = /if:interfaces/if:interface
                 [ni:bind-ni-name = current()/../ni:name]/if:name]",
            "/if:interfaces/if:interface/ip:ipv4
               [ni:bind-network-instance-name = current()/../ni:name]",
            "/if:interfaces-state/if:interface/ip:ipv4
               [if:name = /if:interfaces/if:interface/ip:ipv4
                 [ni:bind-ni-name = current()/../ni:name]/if:name]",
            "/if:interfaces/if:interface/ip:ipv6
               [ni:bind-network-instance-name = current()/../ni:name]",
            "/if:interfaces-state/if:interface/ip:ipv6
               [if:name = /if:interfaces/if:interface/ip:ipv4
                 [ni:bind-ni-name = current()/../ni:name]/if:name]"
          ]
        }
      }
    ],
```

3.4. Network Instance Instantiation

Network instances may be controlled by clients using existing list operations. When a list entry is created, a new instance is instantiated. The models mounted under an NI root are expected to be dependent on the server implementation. When a list entry is deleted, an existing network instance is destroyed. For more information, see [RFC7950] Section 7.8.6.

Once instantiated, host network device resources can be associated with the new NI. As previously mentioned, this document augments ietf-interfaces with the bind-ni-name leaf to support such

associations for interfaces. When a bind-ni-name is set to a valid NI name, an implementation MUST take whatever steps are internally necessary to assign the interface to the NI or provide an error message (defined below) with an indication of why the assignment failed. It is possible for the assignment to fail while processing the set operation, or after asynchronous processing. Error notification in the latter case is supported via a notification.

4. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are two different sets of security considerations to consider in the context of this document. One set is security related to information contained within mounted modules. The security considerations for mounted modules are not substantively changed based on the information being accessible within the context of an NI. For example, when considering the modules defined in [I-D.ietf-netmod-rfc8022bis], the security considerations identified in that document are equally applicable, whether those modules are accessed at a server’s root or under an NI instance’s root node.

The second area for consideration is information contained in the NI module itself. NI information represents network configuration and route distribution policy information. As such, the security of this information is important, but it is fundamentally no different than any other interface or routing configuration information that has already been covered in [I-D.ietf-netmod-rfc7223bis] and [I-D.ietf-netmod-rfc8022bis].

The vulnerable "config true" parameters and subtrees are the following:

/network-instances/network-instance: This list specifies the network instances and the related control plane protocols configured on a device.

/if:interfaces/if:interface/*/bind-network-instance-name: This leaf indicates the NI instance to which an interface is assigned.

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.

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-network-instance

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-network-instance
namespace:    urn:ietf:params:xml:ns:yang:ietf-network-instance
prefix:       ni
reference:    RFC XXXX
```

6. Network Instance Model

The structure of the model defined in this document is described by the YANG module below.

```
<CODE BEGINS> file "ietf-network-instance@2018-03-20.yang"
module ietf-network-instance {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-instance";
  prefix ni;

  // import some basic types

  import ietf-interfaces {
    prefix if;
    reference "draft-ietf-netmod-rfc7223bis: A YANG Data Model
               for Interface Management";
  }
  import ietf-ip {
    prefix ip;
```

```
      reference "draft-ietf-netmod-rfc7277bis: A YANG Data Model
                 for IP Management";
    }
    import ietf-yang-schema-mount {
      prefix yangmnt;
      reference "draft-ietf-netmod-schema-mount: YANG Schema Mount";
      // RFC Ed.: Please replace this draft name with the
      // corresponding RFC number
    }

    organization
      "IETF Routing Area (rtgwg) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>

       Author:   Lou Berger
                 <mailto:lberger@labn.net>
       Author:   Christan Hopps
                 <mailto:chopps@chopps.org>
       Author:   Acee Lindem
                 <mailto:acee@cisco.com>
       Author:   Dean Bogdanovic
                 <mailto:ivandean@gmail.com>";
    description
      "This module is used to support multiple network instances
       within a single physical or virtual device.  Network
       instances are commonly known as VRFs (virtual routing
       and forwarding) and VSIs (virtual switching instances).

       Copyright (c) 2017 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.: please update TBD

    revision 2018-03-20 {
```

```
      description
        "Initial revision.";
      reference "RFC TBD";
    }

    // top level device definition statements

    container network-instances {
      description
        "Network instances each of which consists of a
         VRFs (virtual routing and forwarding) and/or
         VSIs (virtual switching instances).";
      reference "draft-ietf-rtgwg-rfc8022bis - A YANG Data Model
                 for Routing Management";
      list network-instance {
        key "name";
        description
          "List of network-instances.";
        leaf name {
          type string;
          mandatory true;
          description
            "device scoped identifier for the network
             instance.";
        }
        leaf enabled {
          type boolean;
          default "true";
          description
            "Flag indicating whether or not the network
             instance is enabled.";
        }
        leaf description {
          type string;
          description
            "Description of the network instance
             and its intended purpose.";
        }
        choice ni-type {
          description
            "This node serves as an anchor point for different types
             of network instances.  Each 'case' is expected to
             differ in terms of the information needed in the
             parent/core to support the NI, and may differ in their
             mounted schema definition. When the mounted schema is
             not expected to be the same for a specific type of NI
             a mount point should be defined.";
        }
```

```
        choice root-type {
          mandatory true;
          description
            "Well known mount points.";
          container vrf-root {
            description
              "Container for mount point.";
            yangmnt:mount-point "vrf-root" {
              description
                "Root for L3VPN type models. This will typically
                 not be an inline type mount point.";
            }
          }
          container vsi-root {
            description
              "Container for mount point.";
            yangmnt:mount-point "vsi-root" {
              description
                "Root for L2VPN type models. This will typically
                 not be an inline type mount point.";
            }
          }
          container vv-root {
            description
              "Container for mount point.";
            yangmnt:mount-point "vv-root" {
              description
                "Root models that support both L2VPN type bridging
                 and L3VPN type routing. This will typically
                 not be an inline type mount point.";
            }
          }
        }
      }
    }

    // augment statements

    augment "/if:interfaces/if:interface" {
      description
        "Add a node for the identification of the network
         instance associated with the information configured
         on a interface.

         Note that a standard error will be returned if the
         identified leafref isn't present.  If an interfaces cannot
         be assigned for any other reason, the operation SHALL fail
         with an error-tag of 'operation-failed' and an
```

```
          error-app-tag of 'ni-assignment-failed'.  A meaningful
          error-info that indicates the source of the assignment
          failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
          }
          description
            "Network Instance to which an interface is bound.";
        }
      }
      augment "/if:interfaces/if:interface/ip:ipv4" {
        description
          "Add a node for the identification of the network
           instance associated with the information configured
           on an IPv4 interface.

           Note that a standard error will be returned if the
           identified leafref isn't present.  If an interfaces cannot
           be assigned for any other reason, the operation SHALL fail
           with an error-tag of 'operation-failed' and an
           error-app-tag of 'ni-assignment-failed'.  A meaningful
           error-info that indicates the source of the assignment
           failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
          }
          description
            "Network Instance to which IPv4 interface is bound.";
        }
      }
      augment "/if:interfaces/if:interface/ip:ipv6" {
        description
          "Add a node for the identification of the network
           instance associated with the information configured
           on an IPv6 interface.

           Note that a standard error will be returned if the
           identified leafref isn't present.  If an interfaces cannot
           be assigned for any other reason, the operation SHALL fail
           with an error-tag of 'operation-failed' and an
           error-app-tag of 'ni-assignment-failed'.  A meaningful
           error-info that indicates the source of the assignment
           failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
```

```
        }
        description
          "Network Instance to which IPv6 interface is bound.";
      }
    }

    // notification statements

    notification bind-ni-name-failed {
      description
        "Indicates an error in the association of an interface to an
         NI. Only generated after success is initially returned when
         bind-ni-name is set.

         Note: some errors may need to be reported for multiple
         associations, e.g., a single error may need to be reported
         for an IPv4 and an IPv6 bind-ni-name.

         At least one container with a bind-ni-name leaf MUST be
         included in this notification.";
      leaf name {
        type leafref {
          path "/if:interfaces/if:interface/if:name";
        }
        mandatory true;
        description
          "Contains the interface name associated with the
          failure.";
      }
      container interface {
        description
          "Generic interface type.";
        leaf bind-ni-name {
          type leafref {
            path "/if:interfaces/if:interface/ni:bind-ni-name";
          }
          description
            "Contains the bind-ni-name associated with the
             failure.";
        }
      }
      container ipv4 {
        description
          "IPv4 interface type.";
        leaf bind-ni-name {
          type leafref {
            path "/if:interfaces/if:interface"
              + "/ip:ipv4/ni:bind-ni-name";
```

```
          }
          description
            "Contains the bind-ni-name associated with the
             failure.";
        }
      }
      container ipv6 {
        description
          "IPv6 interface type.";
        leaf bind-ni-name {
          type leafref {
            path "/if:interfaces/if:interface"
              + "/ip:ipv6/ni:bind-ni-name";
          }
          description
            "Contains the bind-ni-name associated with the
             failure.";
        }
      }
      leaf error-info {
        type string;
        description
          "Optionally, indicates the source of the assignment
           failure.";
      }
    }
  }
  <CODE ENDS>
```

7. References

7.1. Normative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

[I-D.ietf-netmod-rfc7223bis]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-rfc7223bis-03 (work in progress), January 2018.

[I-D.ietf-netmod-rfc7277bis]
Bjorklund, M., "A YANG Data Model for IP Management", draft-ietf-netmod-rfc7277bis-03 (work in progress), January 2018.

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-08 (work in progress), October 2017.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-08 (work in progress), February 2018.

[I-D.ietf-bess-l3vpn-yang]
Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS L3 VPNs", draft-ietf-bess-l3vpn-yang-02 (work in progress), October 2017.

[I-D.ietf-netmod-rfc8022bis]
Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", draft-ietf-netmod-rfc8022bis-11 (work in progress), January 2018.

[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-10 (work in progress), March 2018.

[I-D.ietf-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

[I-D.ietf-rtgwg-lne-model]
Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", draft-ietf-rtgwg-lne-model-09 (work in progress), March 2018.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <https://www.rfc-editor.org/info/rfc4026>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <https://www.rfc-editor.org/info/rfc4664>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <https://www.rfc-editor.org/info/rfc7277>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

Appendix A. Acknowledgments

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Yan Gang. Useful review comments were also received by Martin Bjorklund and John Scudder.

This document was motivated by, and derived from, [I-D.ietf-rtgwg-device-model].

Thanks for AD and IETF last call comments from Alia Atlas, Liang Xia, Benoit Claise, and Adam Roach.

The RFC text was produced using Marshall Rose’s xml2rfc tool.

Appendix B. Example NI usage

The following subsections provide example uses of NIs.

B.1. Configuration Data

The following shows an example where two customer specific network instances are configured:

```
{
  "ietf-network-instance:network-instances": {
    "network-instance": [
```

```
        {
          "name": "vrf-red",
          "vrf-root": {
            "ietf-routing:routing": {
              "router-id": "192.0.2.1",
              "control-plane-protocols": {
                "control-plane-protocol": [
                  {
                    "type": "ietf-routing:ospf",
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth1",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        },
        {
          "name": "vrf-blue",
          "vrf-root": {
            "ietf-routing:routing": {
              "router-id": "192.0.2.2",
              "control-plane-protocols": {
                "control-plane-protocol": [
                  {
                    "type": "ietf-routing:ospf",
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
```

```
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth2",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        ]
      },

      "ietf-interfaces:interfaces": {
        "interfaces": {
          "interface": [
            {
              "name": "eth0",
              "ip:ipv4": {
                "address": [
                  {
                    "ip": "192.0.2.10",
                    "prefix-length": 24,
                  }
                ]
              },
              "ip:ipv6": {
                "address": [
                  {
                    "ip": "2001:db8:0:2::10",
                    "prefix-length": 64,
                  }
                ]
              }
            },
            {
              "name": "eth1",
              "ip:ipv4": {
```

```
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          },
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          },
          "ni:bind-network-instance-name": "vrf-red"
        },
        {
          "name": "eth2",
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          },
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          },
          "ni:bind-network-instance-name": "vrf-blue"
        }
      ]
    }
  },

  "ietf-system:system": {
    "authentication": {
      "user": [
        {
          "name": "john",
          "password": "$0$password"
        }
```

```
        ]
      }
    }
  }
```

B.2. State Data - Non-NMDA Version

The following shows state data for the configuration example above based on [RFC7223], [RFC7277], and [RFC8022].

```
{
  "ietf-network-instance:network-instances": {
    "network-instance": [
      {
        "name": "vrf-red",
        "vrf-root": {
          "ietf-yang-library:modules-state": {
            "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
            "module": [
              {
                "name": "ietf-yang-library",
                "revision": "2016-06-21",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                "conformance-type": "implement"
              },
              {
                "name": "ietf-ospf",
                "revision": "2018-03-03",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-ospf",
                "conformance-type": "implement"
              },
              {
                "name": "ietf-routing",
                "revision": "2018-03-13",
                "namespace":
                "urn:ietf:params:xml:ns:yang:ietf-routing",
                "conformance-type": "implement"
              }
            ]
          },
          "ietf-routing:routing-state": {
            "router-id": "192.0.2.1",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
```

```
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth1",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        },
        {
          "name": "vrf-blue",
          "vrf-root": {
            "ietf-yang-library:modules-state": {
              "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
              "module": [
                {
                  "name": "ietf-yang-library",
                  "revision": "2016-06-21",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-ospf",
                  "revision": "2018-03-03",
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-ospf",
                  "conformance-type": "implement"
                },
                {
                  "name": "ietf-routing",
                  "revision": "2018-03-13",
```

```
                  "namespace":
                  "urn:ietf:params:xml:ns:yang:ietf-routing",
                  "conformance-type": "implement"
                }
              ]
            },
            "ietf-routing:routing-state": {
              "router-id": "192.0.2.2",
              "control-plane-protocols": {
                "control-plane-protocol": [
                  {
                    "type": "ietf-routing:ospf",
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "af": "ipv4",
                      "areas": {
                        "area": [
                          {
                            "area-id": "203.0.113.1",
                            "interfaces": {
                              "interface": [
                                {
                                  "name": "eth2",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  }
                ]
              }
            }
          }
        }
      ]
    },

    "ietf-interfaces:interfaces-state": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C0",
```

```
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.10",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::10",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C1",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
```

```
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
            "ip:ipv6": {
              "address": [
                {
                  "ip": "2001:db8:0:2::11",
                  "prefix-length": 64,
                }
              ]
            }
          }
        ]
      }
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    }

    "ietf-yang-library:modules-state": {
      "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
      "module": [
        {
          "name": "iana-if-type",
          "revision": "2014-05-08",
          "namespace":
          "urn:ietf:params:xml:ns:yang:iana-if-type",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "conformance-type": "import"
```

```
        },
        {
          "name": "ietf-interfaces",
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-interfaces",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ip",
          "revision": "2014-06-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-ip",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-network-instance",
          "revision": "2018-02-03",
          "feature": [
            "bind-network-instance-name"
          ],
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-network-instance",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-ospf",
          "revision": "2018-03-03",
          "namespace": "urn:ietf:params:xml:ns:yang:ietf-ospf",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-routing",
          "revision": "2018-03-13",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-routing",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-system",
          "revision": "2014-08-06",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-system",
          "conformance-type": "implement"
```

```
        },
        {
          "name": "ietf-yang-library",
          "revision": "2016-06-21",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-library",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-network-instance",
          "label": "vrf-root",
          "shared-schema": {
            "parent-reference": [
              "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
            ]
          }
        }
      ]
    }
  }
```

B.3. State Data - NMDA Version

The following shows state data for the configuration example above based on [I-D.ietf-netmod-rfc7223bis], [I-D.ietf-netmod-rfc7277bis], and [I-D.ietf-netmod-rfc8022bis].

```
{
  "ietf-network-instance:network-instances": {
```

```
    "network-instance": [
      {
        "name": "vrf-red",
        "vrf-root": {
          "ietf-yang-library:yang-library": {
            "checksum": "41e2ab5dc325f6d86f743e8da3de323f1a61a801",
            "module-set": [
              {
                "name": "ni-modules",
                "module": [
                  {
                    "name": "ietf-yang-library",
                    "revision": "2016-06-21",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-ospf",
                    "revision": "2018-03-03",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-ospf",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-routing",
                    "revision": "2018-03-13",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-routing",
                    "conformance-type": "implement"
                  }
                ],
                "import-only-module": [
                  {
                    "name": "ietf-inet-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types"
                  },
                  {
                    "name": "ietf-yang-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types"
                  },
                  {
                    "name": "ietf-datastores",
                    "revision": "2018-02-14",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-datastores"
                  }
```

```
            ]
          }
        ],
        "schema": [
          {
            "name": "ni-schema",
            "module-set": [ "ni-modules" ]
          }
        ],
        "datastore": [
          {
            "name": "ietf-datastores:running",
            "schema": "ni-schema"
          },
          {
            "name": "ietf-datastores:operational",
            "schema": "ni-schema"
          }
        ]
      },
      "ietf-routing:routing": {
        "router-id": "192.0.2.1",
        "control-plane-protocols": {
          "control-plane-protocol": [
            {
              "type": "ietf-routing:ospf",
              "name": "1",
              "ietf-ospf:ospf": {
                "af": "ipv4",
                "areas": {
                  "area": [
                    {
                      "area-id": "203.0.113.1",
                      "interfaces": {
                        "interface": [
                          {
                            "name": "eth1",
                            "cost": 10
                          }
                        ]
                      }
                    }
                  ]
                }
              }
            }
          ]
        }
```

```
          }
        }
      },
      {
        "name": "vrf-blue",
        "vrf-root": {
          "ietf-yang-library:yang-library": {
            "checksum": "41e2ab5dc325f6d86f743e8da3de323f1a61a801",
            "module-set": [
              {
                "name": "ni-modules",
                "module": [
                  {
                    "name": "ietf-yang-library",
                    "revision": "2016-06-21",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-yang-library",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-ospf",
                    "revision": "2018-03-03",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-ospf",
                    "conformance-type": "implement"
                  },
                  {
                    "name": "ietf-routing",
                    "revision": "2018-03-13",
                    "namespace":
                    "urn:ietf:params:xml:ns:yang:ietf-routing",
                    "conformance-type": "implement"
                  }
                ],
                "import-only-module": [
                  {
                    "name": "ietf-inet-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-types"
                  },
                  {
                    "name": "ietf-yang-types",
                    "revision": "2013-07-15",
                    "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-types"
                  },
                  {
                    "name": "ietf-datastores",
                    "revision": "2018-02-14",
```

```
                  "namespace": "urn:ietf:params:xml:ns:yang:ietf-datastores"
                }
              ]
            }
          ],
          "schema": [
            {
              "name": "ni-schema",
              "module-set": [ "ni-modules" ]
            }
          ],
          "datastore": [
            {
              "name": "ietf-datastores:running",
              "schema": "ni-schema"
            },
            {
              "name": "ietf-datastores:operational",
              "schema": "ni-schema"
            }
          ]
        },
        "ietf-routing:routing": {
          "router-id": "192.0.2.2",
          "control-plane-protocols": {
            "control-plane-protocol": [
              {
                "type": "ietf-routing:ospf",
                "name": "1",
                "ietf-ospf:ospf": {
                  "af": "ipv4",
                  "areas": {
                    "area": [
                      {
                        "area-id": "203.0.113.1",
                        "interfaces": {
                          "interface": [
                            {
                              "name": "eth2",
                              "cost": 10
                            }
                          ]
                        }
                      }
                    ]
                  }
                }
              }
```

```
              ]
            }
          }
        }
      }
    ]
  },

  "ietf-interfaces:interfaces": {
    "interfaces": {
      "interface": [
        {
          "name": "eth0",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C0",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.10",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::10",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth1",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C1",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
```

```
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          }
        },
        {
          "name": "eth2",
          "type": "iana-if-type:ethernetCsmacd",
          "oper-status": "up",
          "phys-address": "00:01:02:A1:B1:C2",
          "statistics": {
             "discontinuity-time": "2017-06-26T12:34:56-05:00"
          },
          "ip:ipv4": {
            "address": [
              {
                "ip": "192.0.2.11",
                "prefix-length": 24,
              }
            ]
          }
          "ip:ipv6": {
            "address": [
              {
                "ip": "2001:db8:0:2::11",
                "prefix-length": 64,
              }
            ]
          }
        }
      ]
    }
  },

  "ietf-system:system-state": {
    "platform": {
      "os-name": "NetworkOS"
    }
  }
```

```
  "ietf-yang-library:modules-state": {
    "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
    "module": [
      {
        "name": "iana-if-type",
        "revision": "2014-05-08",
        "namespace":
        "urn:ietf:params:xml:ns:yang:iana-if-type",
        "conformance-type": "import"
      },
      {
        "name": "ietf-inet-types",
        "revision": "2013-07-15",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-inet-types",
        "conformance-type": "import"
      },
      {
        "name": "ietf-interfaces",
        "revision": "2018-01-09",
        "feature": [
          "arbitrary-names",
          "pre-provisioning"
        ],
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-interfaces",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ip",
        "revision": "2018-01-09",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-ip",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-network-instance",
        "revision": "2018-02-03",
        "feature": [
          "bind-network-instance-name"
        ],
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-network-instance",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-ospf",
        "revision": "2017-10-30",
```

```
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-ospf",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-routing",
        "revision": "2018-01-25",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-routing",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-system",
        "revision": "2014-08-06",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-system",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-yang-library",
        "revision": "2016-06-21",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-yang-library",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-yang-schema-mount",
        "revision": "2017-05-16",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
        "conformance-type": "implement"
      },
      {
        "name": "ietf-yang-types",
        "revision": "2013-07-15",
        "namespace":
        "urn:ietf:params:xml:ns:yang:ietf-yang-types",
        "conformance-type": "import"
      }
    ]
  },

  "ietf-yang-schema-mount:schema-mounts": {
    "mount-point": [
      {
        "module": "ietf-network-instance",
        "label": "vrf-root",
        "shared-schema": {
          "parent-reference": [
```

```
            "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
          ]
        }
      }
    ]
  }
}
```

Authors' Addresses

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Dean Bogdanovic

Email: ivandean@gmail.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2017

X. Liu
Kuatro Technologies
I. Bryskin
Huawei Technologies
V. Beeram
Juniper Networks
T. Saad
Cisco Systems Inc
H. Shah
Ciena
O. Gonzalez de Dios
Telefonica
October 24, 2016

A YANG Data Model for Configuration Scheduling
draft-liu-netmod-yang-schedule-02

Abstract

This document describes a data model for configuration scheduling.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 24, 2017.

Table of Contents

1. Introduction

This document introduces a YANG [RFC6020] data model for configuration scheduling. This model can be used together with other YANG data models to specify a schedule applied on a configuration data node, so that the configuration data can take effect according to the schedule.

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 [RFC6020] and are not redefined here:

- o augment
- o data model
- o data node

2. Motivation

Some applications benefit from resource scheduling to allow operators to plan ahead of time. Traffic engineering is one of such examples [RFC7399]. When configuration and state models are designed for such applications, it has been considered that certain data objects need to be configured according to predefined schedules. In other situations, operators need to de-configure certain data objects at predefined schedules for the purposes such as maintenance. These data objects are interpreted and implemented by the applicable applications.

3. Configuration Scheduling YANG Data Model Overview

This document defines a YANG data model that specifies configuration schedules for other YANG data models. For each targeted configuration data object or a group of configuration data objects, an entry is specified along with requested schedules using this configuration schedule model. The application implementing the targeted schema nodes implements the configuration schedules, configuring or de-configuring the specified objects according to the specified schedules. The model schema of the targeted application does not need changes, so the data model described in this document can be used for any data model. The configuration scheduling YANG data model has the following structure:

```
module: ietf-schedule
   +--rw configuration-schedules
      +--rw target* [object]
         +--rw object        yang:xpath1.0
         +--rw data-value?   string
         +--rw schedules
         |  +--rw schedule* [schedule-id]
         |     +--rw schedule-id            uint32
         |     +--rw inclusive-exclusive?   enumeration
         |     +--rw start?                 yang:date-and-time
         |     +--rw schedule-duration?     string
         |     +--rw repeat-interval?       string
         +--ro state
```

```
              +--ro next-event
                 +--ro start?       yang:date-and-time
                 +--ro duration?    string
                 +--ro operation?   enumerationmodule: ietf-schedule
```

4. Usage Example

The following model defines a list of TE (Traffic Engineering) links which can be configured with specified schedules:

```
module: example
   +--rw te-links
      +--rw te-link* [id]
         +--rw id           string
```

The following configuration requests that

- link-1 is configured weekly for five one-day periods, starting from 2016-09-12T23:20:50.52Z.
- link-2 is de-configured for two hours, starting from 2016-09-15T01:00:00.00Z.

```
<configuration-schedules>
  <target xmlns:ex=""urn:example">
    <object>/ex:te-links/ex:te-link[ex:link-id='link-1']</object>
    <schedules>
      <schedule>
   <schedule-id>11<schedule-id>
   <start>2016-09-12T23:20:50.52Z</start>
   <schedule-duration>P1D</schedule-duration>
   <repeat-interval>R5/P1W</repeat-interval>
      </schedule>
    </schedules>
  </target>
  <target xmlns:ex=""urn:example">
    <object>/ex:te-links/ex:te-link[ex:link-id='link-2']</object>
    <schedules>
      <schedule>
   <schedule-id>12<schedule-id>
   <inclusive-exclusive>exclusive</inclusive-exclusive>
   <start>2016-09-15T01:00:00.00Z</start>
   <schedule-duration>P2H</schedule-duration>
```

```
        </schedule>
      </schedules>
    </target>
  </configuration-schedules>
```

5. Configuration Scheduling YANG Module

```
  <CODE BEGINS> file "ietf-schedule@2016-10-11.yang"
  module ietf-schedule {
    yang-version 1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-schedule";
    // replace with IANA namespace when assigned

    prefix "sch";

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

    organization "TBD";
    contact "TBD";
    description
      "The model allows time scheduling parameters to be specified.";

    revision "2016-10-11" {
      description "Initial revision";
      reference "TBD";
    }

    /*
     * Groupings
     */

    grouping schedule-config-attributes {
      description
        "A group of attributes for a schedule.";

      leaf inclusive-exclusive {
        type enumeration {
          enum inclusive {
            description "The schedule element is inclusive.";
```

```
          }
          enum exclusive {
            description "The schedule element is exclusive.";
          }
        }
        default "inclusive";
        description
          "Whether the list item is inclusive or exclusive.";
      }
      leaf start {
        type yang:date-and-time;
        description "Start time.";
      }
      leaf schedule-duration {
        type string {
          pattern
            'P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?(\d+M)?(\d+S)?';
        }
        description "Schedule duration in ISO 8601 format.";
      }
      leaf repeat-interval {
        type string {
          pattern
            'R\d*/P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?(\d+M)?'
            + '(\d+S)?';
        }
        description "Repeat interval in ISO 8601 format.";
      }
    } // schedule-config-attributes

    grouping schedule-state-attributes {
      description
        "State attributes for a schedule.";
      container next-event {
        description
          "The state information of the nexte scheduled event.";
        leaf start {
          type yang:date-and-time;
          description "Start time.";
        }
        leaf duration {
```

```
          type string {
            pattern
              'P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?(\d+M)?(\d+S)?';
          }
          description "Schedule duration in ISO 8601 format.";
        }
        leaf operation {
          type enumeration {
            enum configure {
              description
                "Create the configuration data.";
            }
            enum deconfigure {
              description
                "Remove the configuration data.";
            }
            enum set {
              description
                "Set the specified configuration data.";
            }
            enum reset {
              description
                "Revert the specified configuration data back to the
                 original value.";
            }
          }
          description "Operation type.";
        }
      } // next-event
    } // schedule-state-attributes

    grouping schedules {
      description
        "A list of schedules defining when a particular
         configuration takes effect.";
      container schedules {
        description
          "Container of a schedule list defining when a particular
           configuration takes effect.";
        list schedule {
          key "schedule-id";
```

```
          description "A list of schedule elements.";
          leaf schedule-id {
            type uint32;
            description "Identifies the schedule element.";
          }
          uses schedule-config-attributes;
        }
      }
    } // schedules

    /*
     * Configuration data nodes
     */
    container configuration-schedules {
      description
        "Serves as top-level container for a list of configuration
         schedules.";
      list target {
        key "object";
        description
          "A list of targets that configuration schedules are
           applied.";
        leaf object {
          type yang:xpath1.0;
          description
            "Xpath defining the data items of interest.";
        }
        leaf data-value {
          type string;
          description
            "The ephemeral value applied to the leaf data node
             specified by data-objects.
             This is applicable when object is a leaf.";
        }
        uses schedules;
        container state {
          config false;
          description
            "Operational state data.";
          uses schedule-state-attributes;
        } // state
```

```
      } // target
    } // configuration-schedules
  }
  <CODE ENDS>
```

6. Security Considerations

The configuration, state, action and notification data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241]. The data-model by itself does not create any security implications. The security considerations for the NETCONF protocol are applicable. The NETCONF protocol used for sending the data supports authentication and encryption.

7. References

7.1. 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.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC7399] Farrel, A. and King, D., "Unanswered Questions in the Path Computation Element Architecture", RFC 7399, October 2014.

7.2. Informative References

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.

Authors' Addresses

Xufeng Liu
Kuatro Technologies
8281 Greensboro Drive, Suite 200
McLean, VA 22102
USA

Email: xliu@kuatrotech.com

Igor Bryskin
Huawei Technologies
Email: Igor.Bryskin@huawei.com

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

Tarek Saad
Cisco Systems Inc
Email: tsaad@cisco.com

Himanshu Shah
Ciena
Email: hshah@ciena.com

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2017

M. Bjorklund, Ed.
Tail-f Systems
J. Schoenwaelder
Jacobs University
P. Shafer
K. Watsen
Juniper
R. Wilton
Cisco

October 27, 2016

A Revised Conceptual Model for YANG Datastores
draft-nmdsdt-netmod-revised-datastores-00

Abstract

Datastores are a fundamental concept binding the YANG data modeling language to protocols transporting data defined in YANG data models, such as NETCONF or RESTCONF. This document defines a revised conceptual model of datastores based on the experience gained with the initial simpler model and addressing requirements that were not well supported in the initial model.

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 30, 2017.

Table of Contents

1. Introduction

This document provides a revised architectural framework for datastores as they are used by network management protocols such as NETCONF [RFC6241], RESTCONF [I-D.ietf-netconf-restconf] and the YANG [RFC7950] data modeling language. Datastores are a fundamental concept binding management data models to network management protocols and agreement on a common architectural model of datastores ensures that data models can be written in a network management

protocol agnostic way. This architectural framework identifies a set of conceptual datastores but it does not mandate that all network management protocols expose all these conceptual datastores. Furthermore, the architecture does not detail how data is encoded by network management protocols.

2. Background

NETCONF [RFC6241] provides the following definitions:

- datastore: A conceptual place to store and access information. A datastore might be implemented, for example, using files, a database, flash memory locations, or combinations thereof.

- configuration datastore: The datastore holding the complete set of configuration data that is required to get a device from its initial default state into a desired operational state.

YANG 1.1 [RFC7950] provides the following refinements when NETCONF is used with YANG (which is the usual case but note that NETCONF was defined before YANG did exist):

- datastore: When modeled with YANG, a datastore is realized as an instantiated data tree.

- configuration datastore: When modeled with YANG, a configuration datastore is realized as an instantiated data tree with configuration data.

RFC 6244 defined operational state data as follows:

- Operational state data is a set of data that has been obtained by the system at runtime and influences the system’s behavior similar to configuration data. In contrast to configuration data, operational state is transient and modified by interactions with internal components or other systems via specialized protocols.

Section 4.3.3 of RFC 6244 discusses operational state and among other things mentions the option to consider operational state as being stored in another datastore. Section 4.4 of this document then concludes that at the time of the writing, modeling state as a separate data tree is the recommended approach.

Implementation experience and requests from operators [I-D.ietf-netmod-opstate-reqs], [I-D.openconfig-netmod-opstate] indicate that the datastore model initially designed for NETCONF and refined by YANG needs to be extended. In particular, the notion of intended configuration and applied configuration has developed.

Furthermore, separating operational state data from configuration data in a separate branch in the data model has been found operationally complicated. The relationship between the branches is not machine readable and filter expressions operating on configuration data and on related operational state data are different.

3. Terminology

This document defines the following terms:

- configuration data: Data that determines how a device behaves. Configuration data can originate from different sources. In YANG 1.1, configuration data is the "config true" nodes.

- static configuration data: Configuration data that is eventually persistent and used to get a device from its initial default state into its desired operational state.

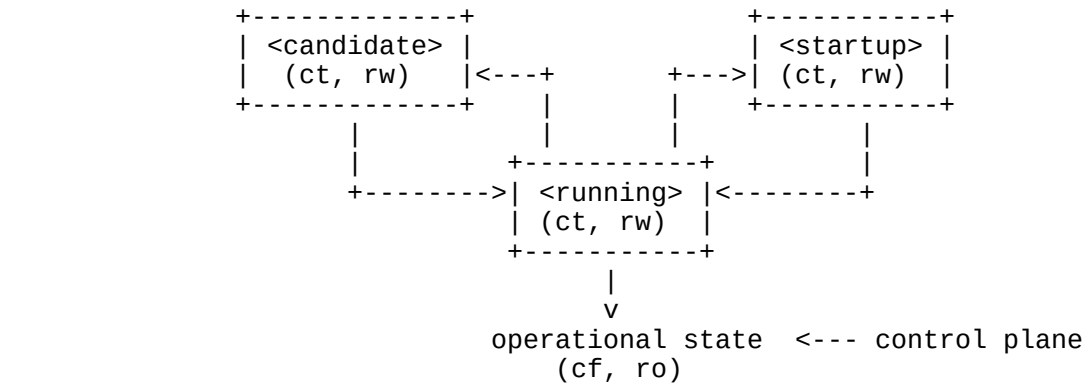
- dynamic configuration data: Configuration data that is obtained dynamically during the operation of a device through interaction with other systems and not persistent.

- system configuration data: Configuration data that is supplied by the device itself.

- data-model-defined configuration data: Configuration data that is not explicitly provided but for which a value defined in the data model is used. In YANG 1.1, such data can be defined with the "default" statement or in "description" statements.

4. Original Model of Datastores

The following drawing shows the original model of datastores as it is currently used by NETCONF [RFC6241]:



ct = config true; cf = config false
rw = read-write; ro = read-only
boxes denote datastores

Note that read-only (ro) and read-write (rw) is to be understood at a conceptual level. In NETCONF, for example, support for the <candidate> and <startup> datastores is optional and the <running> datastore does not have to be writable. Furthermore, the <startup> datastore can only be modified by copying <running> to <startup> in the standardized NETCONF datastore editing model. The RESTCONF protocol does not expose these differences and instead provides only a writable unified datastore, which hides whether edits are done through a <candidate> datastore or by directly modifying the <running> datastore or via some other implementation specific mechanism. RESTCONF also hides how configuration is made persistent. Note that implementations may also have additional datastores that can propagate changes to the <running> datastore. NETCONF explicitly mentions so called named datastores.

Some observations:

- Operational state has not been defined as a datastore although there were proposals in the past to introduce an operational state datastore.

- The NETCONF <get/> operation returns the content of the <running> configuration datastore together with the operational state. It is therefore necessary that config false data is in a different branch than the config true data if the operational state data can have a different lifetime compared to configuration data or if configuration data is not immediately or successfully applied.

- Several implementations have proprietary mechanisms that allow clients to store inactive data in the <running> datastore; this

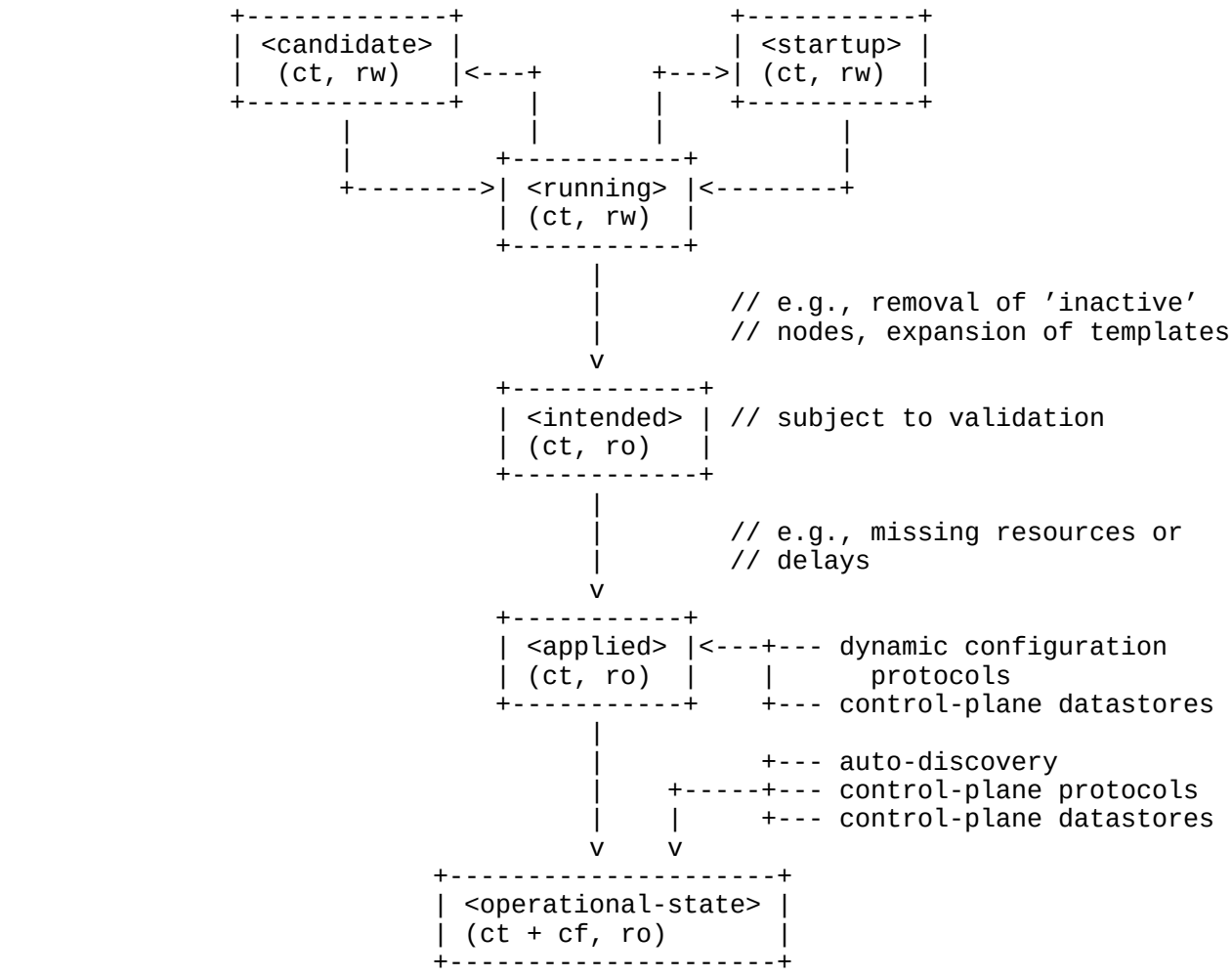
inactive data is only exposed to clients that indicate that they support the concept of inactive data; clients not indicating support for inactive data receive the content of the <running> datastore with the inactive data removed. Inactive data is conceptually removed during validation.

- Some implementations have proprietary mechanisms that allow clients to define configuration templates in <running>. These templates are expanded automatically by the system, and the resulting configuration is applied internally.

- Some operators have reported that it is essential for them to be able to retrieve the configuration that has actually been successfully applied, which may be a subset or a superset of the <running> configuration.

5. Revised Model of Datastores

Below is a new conceptual model of datastores extending the original model in order reflect the experience gained with the original model.



ct = config true; cf = config false
rw = read-write; ro = read-only
boxes denote datastores

The model foresees control-plane datastores that are by definition not part of the persistent configuration of a device. In some contexts, these have been termed ephemeral datastores since the information is ephemeral, i.e., lost upon reboot. The control-plane datastores interact with the rest of the system through the <applied> or <operational-state> datastores, depending on the type of data they contain. Note that the ephemeral datastore discussed in I2RS documents maps to a control-plane datastore in the revised datastore model described here.

5.1. The <intended> datastore

The <intended> datastore is a read-only datastore that consists of config true nodes. It is tightly coupled to <running>. When data is written to <running>, the data that is to be validated is also conceptually written to <intended>. Validation is performed on the contents of <intended>.

On a traditional NETCONF implementation, <running> and <intended> are always the same.

Currently there are no standard mechanisms defined that affect <intended> so that it would have different contents than <running>, but this architecture allows for such mechanisms to be defined.

One example of such a mechanism is support for marking nodes as inactive in <running>. Inactive nodes are not copied to <intended>, and are thus not taken into account when validating the configuration.

Another example is support for templates. Templates are expanded when copied into <intended>, and the result is validated.

5.2. The <applied> datastore

The <applied> datastore is a read-only datastore that consists of config true nodes. It contains the currently active configuration on the device. This data can come from several sources; from <intended>, from dynamic configuration protocols (e.g., DHCP), or from control-plane datastores.

As data flows into the <applied> and <operational-state> datastores, it is conceptually marked with a metadata annotation ([RFC7952]) that indicates its origin. The "origin" metadata annotation is defined in Section 8. The values are YANG identities. The following identities are defined:

```
  +-- origin
      +-- static
      +-- dynamic
      +-- data-model
      +-- system
```

These identities can be further refined, e.g., there might be an identity "dhcp" derived from "dynamic".

The <applied> datastore contains the subset of the instances in the <operational-state> datastore where the "origin" values are derived from or equal to "static" or "dynamic".

5.2.1. Missing Resources

Sometimes some parts of <intended> configuration refer to resources that are not present and hence parts of the <intended> configuration cannot be applied. A typical example is an interface configuration that refers to an interface that is not currently present. In such a situation, the interface configuration remains in <intended> but the interface configuration will not appear in <applied>.

5.2.2. System-controlled Resources

Sometimes resources are controlled by the device and such system controlled resources appear in (and disappear from) the <operational-state> dynamically. If a system controlled resource has matching configuration in <intended> when it appears, the system will try to apply the configuration, which causes the configuration to appear in <applied> eventually (if application of the configuration was successful).

5.3. The <operational-state> datastore

The <operational-state> datastore is a read-only datastore that consists of config true and config false nodes. In the original NETCONF model the operational state only had config false nodes. The reason for incorporating config true nodes here is to be able to expose all operational settings without having to replicate definitions in the data models.

The <operational-state> datastore contains all configura data actually used by the system, i.e., all applied configuration, system configuration and data-model-defined configuration. This data is marked with the "origin" metadata annotation. In addition, the <operational-state> datastore also contains state data.

In the <operational-state> datastore, semantic constraints defined in the data model are not applied. See Appendix B.

6. Implications

6.1. Implications on NETCONF

- A mechanism is needed to announce support for <intended>, <applied>, and <operational-state>.

- Support for <intended>, <applied>, and <operational-state> should be optional to implement.

- For systems supporting <intended> or <applied> configuration datastores, the <get-config/> operation may be used to retrieve data stored in these new datastores.

- A new operation should be added to retrieve the operational state data store (e.g., <get-state/>). An alternative is to define a new operation to retrieve data from any datastore (e.g., <get-data> with the name of the datastore as a parameter). In principle <get-config/> could work but it would be a confusing name.

- The <get/> operation will be deprecated since it returns data from two datastores that may overlap in the revised datastore model.

6.1.1. Migration Path

A common approach in current data models is to have two separate trees "/foo" and "/foo-state", where the former contains config true nodes, and the latter config false nodes. A data model that is designed for the revised architectural framework presented in this document will have a single tree "/foo" with a combination of config true and config false nodes.

A server that implements the <operational-state> datastore can implement a module of the old design. In this case, some instances are probably reported both in the "/foo" tree and in the "/foo-state" tree.

A server that does not implement the <operational-state> datastore can implement a module of the new design, but with limited functionality. Specifically, it may not be possible to retrieve all operationally used instances (e.g., dynamically configured or system-controlled). The same limitation applies to a client that does not implement the <operational-state> datastore, but talks to a server that implements it.

6.2. Implications on RESTCONF

- The {+restconf}/data resource represents the combined configuration and state data resources that can be accessed by a client. This is effectively bundling <running> together with <operational-state>, much like the <get/> operation of NETCONF. This design should be deprecated.

- A new query parameter is needed to indicate that data from <operational-state> is requested.

6.3. Implications on YANG

- Some clarifications may be needed if this revised model is adopted. YANG currently describes validation in terms of the <running> configuration datastore while it really happens on the <intended> configuration datastore.

6.4. Implications on Data Models

- Since the NETCONF <get/> operation returns the content of the <running> configuration datastore and the operational state together in one tree, data models were often forced to branch at the top-level into a config true branch and a structurally similar config false branch that replicated some of the config true nodes and added state nodes. With the revised datastore model this is not needed anymore since the different datastores handle the different lifetimes of data objects. Introducing this model together with the deprecation of the <get/> operation makes it possible to write simpler models.

- There may be some differences in the value set of some nodes that are used for both configuration and state. At this point of time, these are considered to be rare cases that can be dealt with using different nodes for the configured and state values.

- It is important to design data models with clear semantics that work equally well for instantiation in a configuration datastore and instantiation in the <operational-state> datastore.

7. Data Model Design Guidelines

7.1. Auto-configured or Auto-negotiated Values

Sometimes configuration leafs support special values that instruct the system to automatically configure a value. An example is an MTU that is configured to 'auto' to let the system determine a suitable MTU value. Another example is Ethernet auto-negotiation of link speed. In such a situation, it is recommended to model this as two separate leafs, one config true leaf for the input to the auto-negotiation process, and one config false leaf for the output from the process.

8. Data Model

```
<CODE BEGINS> file "ietf-yang-architecture@2016-10-13.yang"

module ietf-yang-architecture {
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-architecture";
  prefix arch;

  import ietf-yang-metadata {
    prefix md;
  }

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

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>

     WG List:  <mailto:netmod@ietf.org>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>";

  description
    "This YANG module defines an 'origin' metadata annotation,
     and a set of identities for the origin value.  The 'origin'
     metadata annotation is used to mark data in the applied
     and operational state datastores with information on where
     the data originated.

     Copyright (c) 2016 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
     (http://www.rfc-editor.org/info/rfcxxxx); see the RFC itself
     for full legal notices.";

  revision 2016-10-13 {
    description
      "Initial revision.";
    reference
```

```
        "RFC XXXX: A Revised Conceptual Model for YANG Datastores";
    }

    /*
     * Identities
     */

    identity origin {
      description
        "Abstract base identitiy for the origin annotation.";
    }

    identity static {
      base origin;
      description
        "Denotes data from static configuration (e.g., <intended>).";
    }

    identity dynamic {
      base origin;
      description
        "Denotes data from dynamic configuration protocols
         or dynamic datastores (e.g., DHCP).";
    }

    identity system {
      base origin;
      description
        "Denotes data created by the system independently of what
         has been configured.";
    }

    identity data-model {
      base origin;
      description
        "Denotes data that does not have an explicitly configured
         value, but has a default value in use.  Covers both simple
         defaults and complex defaults.";
    }

    /*
     * Metadata annotations
     */

    md:annotation origin {
      type identityref {
        base origin;
      }
```

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

9. IANA Considerations

TBD

10. Security Considerations

This document discusses a conceptual model of datastores for network management using NETCONF/RESTCONF and YANG. It has no security impact on the Internet.

11. Acknowledgments

This document grew out of many discussions that took place since 2010. Several Internet-Drafts ([I-D.bjorklund-netmod-operational], [I-D.wilton-netmod-opstate-yang], [I-D.ietf-netmod-opstate-reqs], [I-D.kwatsen-netmod-opstate], [I-D.openconfig-netmod-opstate]) and [RFC6244] touched on some of the problems of the original datastore model. The following people were authors to these Internet-Drafts or otherwise actively involved in the discussions that led to this document:

- Lou Berger, LabN Consulting, L.L.C., <lberger@labn.net>
- Andy Bierman, YumaWorks, <andy@yumaworks.com>
- Marcus Hines, Google, <hines@google.com>
- Christian Hopps, Deutsche Telekom, <chopps@chopps.org>
- Acee Lindem, Cisco Systems, <acee@cisco.com>
- Ladislav Lhotka, CZ.NIC, <lhotka@nic.cz>
- Thomas Nadeau, Brocade Networks, <tnadeau@lucidvision.com>
- Anees Shaikh, Google, <aashaikh@google.com>
- Rob Shakir, Google, <robjs@google.com>

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

12. References

12.1. Normative References

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-17 (work in progress), September 2016.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <http://www.rfc-editor.org/info/rfc7952>.

12.2. Informative References

[I-D.bjorklund-netmod-operational]
Bjorklund, M. and L. Lhotka, "Operational Data in NETCONF and YANG", draft-bjorklund-netmod-operational-00 (work in progress), October 2012.

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-04 (work in progress), January 2016.

[I-D.kwatsen-netmod-opstate]
Watsen, K., Bierman, A., Bjorklund, M., and J. Schoenwaelder, "Operational State Enhancements for YANG, NETCONF, and RESTCONF", draft-kwatsen-netmod-opstate-02 (work in progress), February 2016.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[I-D.wilton-netmod-opstate-yang]
Wilton, R., ""With-config-state" Capability for NETCONF/RESTCONF", draft-wilton-netmod-opstate-yang-02 (work in progress), December 2015.

[RFC6244] Shafer, P., "An Architecture for Network Management Using NETCONF and YANG", RFC 6244, DOI 10.17487/RFC6244, June 2011, <http://www.rfc-editor.org/info/rfc6244>.

Appendix A. Example Data

In this example, the following fictional module is used:

```
module example-system {
  yang-version 1.1;
  namespace urn:example:system;
  prefix sys;

  import ietf-inet-types {
    prefix inet;
  }

  container system {
    leaf hostname {
      type string;
    }

    list interface {
      key name;

      leaf name {
        type string;
      }

      container auto-negotiation {
        leaf enabled {
          type boolean;
          default true;
        }
        leaf speed {
          type uint32;
          units mbps;
          description
            "The advertised speed, in mbps.";
        }
      }

      leaf speed {
```

```
          type uint32;
          units mbps;
          config false;
          description
            "The speed of the interface, in mbps.";
        }

        list address {
          key ip;

          leaf ip {
            type inet:ip-address;
          }
          leaf prefix-length {
            type uint8;
          }
        }
      }
    }
  }
```

The operator has configured the host name and two interfaces, so the contents of <intended> is:

```
<system xmlns="urn:example:system">

  <hostname>foo</hostname>

  <interface>
    <name>eth0</name>
    <auto-negotiation>
      <speed>1000</speed>
    </auto-negotiation>
    <address>
      <ip>2001:db8::10</ip>
      <prefix-length>32</prefix-length>
    </address>
  </interface>

  <interface>
    <name>eth1</name>
    <address>
      <ip>2001:db8::20</ip>
      <prefix-length>32</prefix-length>
    </address>
  </interface>

</system>
```

The system has detected that the hardware for one of the configured interfaces ("eth1") is not yet present, so the configuration for that interface is not applied. Further, the system has received a host name and an additional IP address for "eth0" over DHCP. This is reflected in <applied>:

```
<system
    xmlns="urn:example:system"
    xmlns:arch="urn:ietf:params:xml:ns:yang:ietf-yang-architecture">

  <hostname arch:origin="arch:dynamic">bar</hostname>

  <interface arch:origin="arch:static">
    <name>eth0</name>
    <auto-negotiation>
      <speed>1000</speed>
    </auto-negotiation>
    <address>
      <ip>2001:db8::10</ip>
      <prefix-length>32</prefix-length>
    </address>
    <address arch:origin="arch:dynamic">
      <ip>2001:db8::1:100</ip>
      <prefix-length>32</prefix-length>
    </address>
  </interface>

</system>
```

In <operational-state>, all data from <applied> is present, in addition to a default value, a loopback interface automatically added by the system, and the result of the "speed" auto-negotiation:

```
   <system
       xmlns="urn:example:system"
       xmlns:arch="urn:ietf:params:xml:ns:yang:ietf-yang-architecture">

     <hostname arch:origin="arch:dynamic">bar</hostname>

     <interface arch:origin="arch:static">
       <name>eth0</name>
       <auto-negotiation>
         <enabled arch:origin="arch:data-model">true</enabled>
         <speed>1000</speed>
       </auto-negotiation>
       <speed>100</speed>
       <address>
         <ip>2001:db8::10</ip>
         <prefix-length>32</prefix-length>
       </address>
       <address arch:origin="arch:dynamic">
         <ip>2001:db8::1:100</ip>
         <prefix-length>32</prefix-length>
       </address>
     </interface>

     <interface arch:origin="arch:system">
       <name>lo0</name>
       <address>
         <ip>::1</ip>
         <prefix-length>128</prefix-length>
       </address>
     </interface>

   </system>
```

Appendix B. Open Issues

1. Do we need another DS <active> inbetween <running> and <intended>? This DS would allow a client to see all active nodes, including unexpanded templates.

2. How do we handle semantical constraints in <operational-state>? Are they just ignored? Do we need a new YANG statement to define if a "must" constraints applies to the <operational-state>?

3. Should it be possible to ask for <applied> in RESTCONF?

4. Better name for "static configuration"?

5. Better name for "intended"?

Authors' Addresses

Martin Bjorklund (editor)
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Phil Shafer
Juniper

Email: phil@juniper.net

Kent Watsen
Juniper

Email: kwatsen@juniper.net

Rob Wilton
Cisco

Email: rwilton@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2017

X. Liu
Kuatro Technologies
Y. Qu
A. Lindem
Cisco Systems
C. Hopps
Deutsche Telekom
L. Berger
LabN Consulting, L.L.C.

October 28, 2016

Routing Area Common YANG Data Types
draft-rtgyangdt-rtgwg-routing-types-00

Abstract

This document defines a collection of common data types using YANG data modeling language. These derived common types are designed to be imported by other modules defined in the routing area.

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 May 1, 2017.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] [RFC7950] is a data modeling language used to model configuration data, state data, Remote Procedure Calls, and notifications for network management protocols. 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 the common types applicable for modeling in the routing area.

1.1. Requirements Language

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 BCP 14, RFC 2119 [RFC2119].

1.2. Terminology

The terminology for describing YANG data models is found in [RFC7950].

2. Overview

This document defines the following data types:

router-id
: Router Identifiers are commonly used to identify a nodes in routing and other control plane protocols. An example usage of router-id can be found in [I-D.ietf-ospf-yang].

address-family
: This type defines values for use in address family identifiers. The values are based on the IANA Address Family Numbers Registry [1]. An example usage can be found in [I-D.ietf-idr-bgp-model].

route-target
: Route Targets (RTs) are commonly used to control the distribution of virtual routing and forwarding (VRF) information, see [RFC4364], in support of virtual private networks (VPNs). An example usage can be found in [I-D.ietf-idr-bgp-model] and

route-distinguisher
: Route Distinguishers (RDs) are commonly used to identify separate routes in support of virtual private networks (VPNs). For example, in [RFC4364], RDs are commonly used to identify independent VPNs and VRFs, and more generally, to identify multiple routes to the same prefix. An example usage can be found in [I-D.ietf-idr-bgp-model].

ieee-bandwidth
: Bandwidth in IEEE 754 floating point 32-bit binary format [IEEE754]. Commonly used in Traffic Engineering control plane protocols. An example of where this type may/will be used is [I-D.ietf-ospf-yang].

link-access-type
: This type identifies the IGP link type. An example of where this type may/will be used is [I-D.ietf-ospf-yang].

multicast-source-ipv4-addr-type
: IPv4 source address type for use in multicast control protocols. This type also allows the indication of wildcard sources, i.e., "*". An example of where this type may/will be used is [I-D.ietf-pim-yang].

multicast-source-ipv6-addr-type
: IPv6 source address type for use in multicast control protocols. This type also allows the indication of wildcard sources, i.e.,

"*". An example of where this type may/will be used is [I-D.ietf-pim-yang].

timer-multiplier
This type is used in conjunction with a timer-value type. It is generally used to indicate define the number of timer-value intervals that may expire before a specific event must occur. Examples of this include the arrival of any BFD packets, see [RFC5880] Section 6.8.4, or hello_interval in [RFC3209]. Example of where this type may/will be used is [I-D.ietf-idr-bgp-model] and [I-D.ietf-teas-yang-rsvp].

timer-value-seconds16
This type covers timers which can be set in seconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint16 (2 octets). An example of where this type may/will be used is [I-D.ietf-ospf-yang].

timer-value-seconds32
This type covers timers which can be set in seconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint32 (4 octets). An example of where this type may/will be used is [I-D.ietf-teas-yang-rsvp].

timer-value-milliseconds
This type covers timers which can be set in milliseconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint32 (4 octets). Examples of where this type may/will be used include [I-D.ietf-teas-yang-rsvp] and [I-D.ietf-bfd-yang].

3. YANG Module

<CODE BEGINS> file "ietf-routing-types@2016-10-28.yang"

```
module ietf-routing-types {

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

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

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

  organization "IETF Routing Area Working Group (rtgwg)";
```

```
     contact
         "Routing Area Working Group - <rtgwg@ietf.org>";

     description
       "This module contains a collection of YANG data types
        considered generally useful for routing protocols.";

     revision 2016-10-28 {
       description
         "Initial revision.";
       reference
         "RFC TBD: Routing YANG Data Types";
     }

     /*** collection of types related to routing ***/
     typedef router-id {
       type yang:dotted-quad;
       description
         "A 32-bit number in the dotted quad format assigned to each
          router. This number uniquely identifies the router within an
          Autonomous System.";
     }

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

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

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

     //The rest of the values deinfed in the IANA registry

     identity nsap {
       base address-family;
       description
         "Address family from IANA registry.";
```

```
  }
  identity hdlc {
    base address-family;
    description
      "(8-bit multidrop)
       Address family from IANA registry.";
  }
  identity bbn1822 {
    base address-family;
    description
      "AHIP (BBN report #1822)
       Address family from IANA registry.";
  }
  identity ieee802 {
    base address-family;
    description
      "(includes all 802 media plus Ethernet canonical format)
       Address family from IANA registry.";
  }
  identity e163 {
    base address-family;
    description
      "Address family from IANA registry.";
  }
  identity e164 {
    base address-family;
    description
      "SMDS, Frame Relay, ATM
       Address family from IANA registry.";
  }
  identity f69 {
    base address-family;
    description
      "(Telex)
       Address family from IANA registry.";
  }
  identity x121 {
    base address-family;
    description
      "(X.25, Frame Relay)
       Address family from IANA registry.";
  }
  identity ipx {
    base address-family;
    description
      "Address family from IANA registry.";
  }
  identity appletalk {
```

```
      base address-family;
      description
        "Address family from IANA registry.";
    }
    identity decnet-iv {
      base address-family;
      description
        "Decnet IV
         Address family from IANA registry.";
    }
    identity vines {
      base address-family;
      description
        "Banyan Vines
         Address family from IANA registry.";
    }
    identity e164-nsap {
      base address-family;
      description
        "E.164 with NSAP format subaddress
         Address family from IANA registry.";
    }
    identity dns {
      base address-family;
      description
        "Domain Name System
         Address family from IANA registry.";
    }
    identity dn {
      base address-family;
      description
        "Distinguished Name
         Address family from IANA registry.";
    }
    identity as-num {
      base address-family;
      description
        "AS Number
         Address family from IANA registry.";
    }
    identity xtp-v4 {
      base address-family;
      description
        "XTP over IPv4
         Address family from IANA registry.";
    }
    identity xtp-v6 {
      base address-family;
```

```
      description
        "XTP over IPv6
         Address family from IANA registry.";
    }
    identity xtp {
      base address-family;
      description
        "XTP native mode XTP
         Address family from IANA registry.";
    }
    identity fc-port {
      base address-family;
      description
        "Fibre Channel World-Wide Port Name
         Address family from IANA registry.";
    }
    identity fc-node {
      base address-family;
      description
        "Fibre Channel World-Wide Node Name
         Address family from IANA registry.";
    }
    identity gwid {
      base address-family;
      description
        "Address family from IANA registry.";
    }
    identity l2vpn {
      base address-family;
      description
        "Address family from IANA registry.";
    }
    identity mpls-tp-section-eid {
      base address-family;
      description
        "MPLS-TP Section Endpoint Identifier
         Address family from IANA registry.";
    }
    identity mpls-tp-lsp-eid {
      base address-family;
      description
        "MPLS-TP LSP Endpoint Identifier
         Address family from IANA registry.";
    }
    identity mpls-tp-pwe-eid {
      base address-family;
      description
        "MPLS-TP Pseudowire Endpoint Identifier
```

```
          Address family from IANA registry.";
      }
      identity mt-v4 {
        base address-family;
        description
          "Multi-Topology IPv4.
           Address family from IANA registry.";
      }
      identity mt-v6 {
        base address-family;
        description
          "Multi-Topology IPv6.
           Address family from IANA registry.";
      }

      /*** collection of types related to VPN ***/
      typedef route-target {
        type string {
          pattern
            '(0:(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d):(429496729[0-5]|42949672[0-8]\d|'
          + '4294967[01]\d{2}|429496[0-6]\d{3}|42949[0-5]\d{4}|'
          + '4294[0-8]\d{5}|429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|'
          + '[0-3]?\d{0,8}\d))|'
          + '(1:(((\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])\.){3}(\d|[1-9]\d|'
          + '1\d{2}|2[0-4]\d|25[0-5])):(6553[0-5]|655[0-2]\d|'
          + '65[0-4]\d{2}|6[0-4]\d{3}|[0-5]?\d{0,3}\d))|'
          + '(2:(429496729[0-5]|42949672[0-8]\d|4294967[01]\d{2}|'
          + '429496[0-6]\d{3}|42949[0-5]\d{4}|4294[0-8]\d{5}|'
          + '429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|[0-3]?\d{0,8}\d):'
          + '(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d))';
        }
        description
          "Route target has a similar format to route distinguisher.
           A route target consists of three fields:
           a 2-byte type field, an administrator field,
           and an assigned number field.
           According to the data formats for type 0, 1, and 2 defined in
           RFC4360, the encoding pattern is defined as:

           0:2-byte-asn:4-byte-number
           1:4-byte-ipv4addr:2-byte-number
           2:4-byte-asn:2-byte-number.

           Some valid examples are: 0:100:100, 1:1.1.1.1:100, and
           2:1234567890:203.";
        reference
```

```
        "RFC4360: BGP Extended Communities Attribute.";
      }

      typedef route-distinguisher {
        type string {
          pattern
            '(0:(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d):(429496729[0-5]|42949672[0-8]\d|'
          + '4294967[01]\d{2}|429496[0-6]\d{3}|42949[0-5]\d{4}|'
          + '4294[0-8]\d{5}|429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|'
          + '[0-3]?\d{0,8}\d))|'
          + '(1:(((\d|[1-9]\d|1\d{2}|2[0-4]\d|25[0-5])\.){3}(\d|[1-9]\d|'
          + '1\d{2}|2[0-4]\d|25[0-5])):(6553[0-5]|655[0-2]\d|'
          + '65[0-4]\d{2}|6[0-4]\d{3}|[0-5]?\d{0,3}\d))|'
          + '(2:(429496729[0-5]|42949672[0-8]\d|4294967[01]\d{2}|'
          + '429496[0-6]\d{3}|42949[0-5]\d{4}|4294[0-8]\d{5}|'
          + '429[0-3]\d{6}|42[0-8]\d{7}|4[01]\d{8}|[0-3]?\d{0,8}\d):'
          + '(6553[0-5]|655[0-2]\d|65[0-4]\d{2}|6[0-4]\d{3}|'
          + '[0-5]?\d{0,3}\d))|'
          + '(([3-9a-fA-F]|[1-9a-fA-F][\da-fA-F]{1,3}):'
          + '[\da-fA-F]{1,12})';
        }
        description
          "Route distinguisher has a similar format to route target.
           An route distinguisher consists of three fields:
           a 2-byte type field, an administrator field,
           and an assigned number field.
           According to the data formats for type 0, 1, and 2 defined in
           RFC4364, the encoding pattern is defined as:

           0:2-byte-asn:4-byte-number
           1:4-byte-ipv4addr:2-byte-number
           2:4-byte-asn:2-byte-number.
           2-byte-other-hex-number:6-byte-hex-number

           Some valid examples are: 0:100:100, 1:1.1.1.1:100, and
           2:1234567890:203.";
        reference
          "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
      }

      /*** collection of types common to protocols ***/
      typedef ieee-bandwidth {
        type string {
          pattern
            '0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
          + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?[pP](\+)?(12[0-7]|'
          + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}';
```

```
      }
      description
        "Bandwidth in IEEE 754 floating point 32-bit binary format:
         (-1)**(S) * 2**(Exponent-127) * (1 + Fraction),
         where Exponent uses 8 bits, and Fraction uses 23 bits.
         The units are bytes per second.
         The encoding format is the external hexadecimal-significand
         character sequences specified in IEEE 754 and C99,
         restricted to be normalized, non-negative, and non-fraction:
         0x1.hhhhhhp{+}d or 0X1.HHHHHHP{+}D
         where 'h' and 'H' are hexadecimal digits, 'd' and 'D' are
         integers in the range of [0..127].
         When six hexadecimal digits are used for 'hhhhhh' or 'HHHHHH',
         the least significant digit must be an even number.
         'x' and 'X' indicate hexadecimal; 'p' and 'P' indicate power
         of two.
         Some examples are: 0x0p0, 0x1p10, and 0x1.abcde2p+20";
      reference
        "IEEE Std 754-2008: IEEE Standard for Floating-Point
         Arithmetic.";
    }

    typedef link-access-type {
      type enumeration {
        enum "broadcast" {
          description
          "Specify broadcast multi-access network.";
        }
        enum "non-broadcast" {
          description
          "Specify Non-Broadcast Multi-Access (NBMA) network.";
        }
        enum "point-to-multipoint" {
          description
          "Specify point-to-multipoint network.";
        }
        enum "point-to-point" {
          description
          "Specify point-to-point network.";
        }
      }
      description
        "Link access type.";
    }

    typedef multicast-source-ipv4-addr-type {
      type union {
        type enumeration {
```

```
          enum '*' {
            description
            "Any source address.";
          }
        }
        type inet:ipv4-address;
      }
      description
        "Multicast source IP address type.";
    }

    typedef multicast-source-ipv6-addr-type {
      type union {
        type enumeration {
          enum '*' {
            description
            "Any source address.";
          }
        }
        type inet:ipv6-address;
      }
      description
        "Multicast source IP address type.";
    }

    typedef timer-multiplier {
      type uint8;
      description
        "The number of timer value intervals that should be
         interpreted as a failure.";
    }

    typedef timer-value-seconds16 {
      type union {
        type uint16 {
          range "1..65535";
          }
        type enumeration {
          enum "infinity" {
            description "The timer is set to infinity.";
          }
          enum "no-expiry" {
            description "The timer is not set.";
          }
        }
      }
      units seconds;
      description "Timer value type, in seconds (16 bit range).";
```

```
    }

    typedef timer-value-seconds32 {
      type union {
        type uint32 {
          range "1..4294967295";
          }
        type enumeration {
          enum "infinity" {
            description "The timer is set to infinity.";
          }
          enum "no-expiry" {
            description "The timer is not set.";
          }
        }
      }
      units seconds;
      description "Timer value type, in seconds (32 bit range).";
    }

    typedef timer-value-milliseconds {
      type union {
        type uint32{
          range "1..4294967295";
          }
        type enumeration {
          enum "infinity" {
            description "The timer is set to infinity.";
          }
          enum "no-expiry" {
            description "The timer is not set.";
          }
        }
      }
      units milliseconds;
      description "Timer value type, in milliseconds.";
    }
  }
  <CODE ENDS>
```

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

5. 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 [RFC7950] apply for this document as well.

6. Acknowledgements

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Ebben Aries, Lou Berger, Qin Wu, Rob Shakir, Xufeng Liu, and Yingzhen Qu.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

7.2. Informative References

[IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE Std 754-2008, August 2008.

[I-D.ietf-bfd-yang] Zheng, L., Rahman, R., Networks, J., Jethanandani, M., and G. Mirsky, "Yang Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-03 (work in progress), July 2016.

[I-D.ietf-idr-bgp-model] Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Zhdankin, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-02 (work in progress), July 2016.

[I-D.ietf-ospf-yang] Yeung, D., Qu, Y., Zhang, Z., Bogdanovic, D., and K. Koushik, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-05 (work in progress), July 2016.

[I-D.ietf-pim-yang] Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and f. hu, "A YANG data model for Protocol-Independent Multicast (PIM)", draft-ietf-pim-yang-03 (work in progress), October 2016.

[I-D.ietf-teas-yang-rsvp] Beeram, V., Saad, T., Gandhi, R., Liu, X., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-04 (work in progress), October 2016.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <http://www.rfc-editor.org/info/rfc3209>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <http://www.rfc-editor.org/info/rfc5880>.

7.3. URIs

[1] http://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml

Authors' Addresses

Xufeng Liu
Kuatro Technologies
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

EMail: xliu@kuatrotech.com

Yingzhen Qu
Cisco Systems
170 West Tasman Drive
San Jose CA 95134
USA

EMail: yiqu@cisco.com

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

EMail: acee@cisco.com

Christian Hopps
Deutsche Telekom

EMail: chopps@chopps.org

Lou Berger
LabN Consulting, L.L.C.

EMail: lberger@labn.net

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 24, 2017

R. Wilton, Ed.
D. Ball
T. Singh
Cisco Systems
S. Sivaraj
Juniper Networks
October 21, 2016

Sub-interface VLAN YANG Data Models
draft-wilton-netmod-intf-vlan-yang-04

Abstract

This document defines YANG modules to add support for classifying traffic received on interfaces as Ethernet/VLAN framed packets to sub-interfaces based on the fields available in the Ethernet/VLAN frame headers. These modules allow IETF forwarding protocols (such as IPv6 and VPLS) to interoperate with VLAN tagged traffic orginated from an IEEE 802.1Q compliant bridge. Primarily the classification is based on VLAN identifiers in the 802.1Q VLAN tags, but the model also has support for matching on some other layer 2 frame header fields and is designed to be extensible to match on other arbitrary header fields.

The model differs from an IEEE 802.1Q bridge model in that the configuration is interface/sub-interface based as opposed to being based on membership of an 802.1Q VLAN bridge.

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 24, 2017.

Table of Contents

1. Introduction

This document defines two YANG [RFC6020] modules that augment the encapsulation choice YANG element defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang] and the generic interfaces data model defined in [RFC7223]. The two modules provide configuration nodes to support classification of Ethernet/VLAN traffic to sub-interfaces, that can have interface based feature and service configuration applied to them.

The purpose of these models is to allow IETF defined forwarding protocols, such as IPv6 [RFC2460], Ethernet Pseudo Wires [RFC4448] and VPLS [RFC4761] [RFC4762] to be configurable via YANG when interoperating with VLAN tagged traffic received from an IEEE 802.1Q compliant bridge.

In the case of layer 2 Ethernet services, the flexible encapsulation module also supports flexible rewriting of the VLAN tags contained the in frame header.

For reference, a comparison between the sub-interface based YANG model documented in this draft and an IEEE 802.1Q bridge model is described in Appendix A.

In summary, the YANG modules defined in this internet draft are:

if-l3-vlan.yang - Defines the model for basic classification of VLAN tagged traffic to L3 transport services

flexible-encapsulation.yang - Defines the model for flexible classification of Ethernet/VLAN traffic to L2 transport services

1.1. Terminology

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 RFC 2119 [RFC2119].

Sub-interface: A sub-interface is a small augmentation of a regular interface in the standard YANG module for Interface Management that represents a subset of the traffic handled by its parent interface. As such, it supports both configuration and operational data using any other YANG models that augment or reference interfaces in the normal way. It is defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

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:

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

2. Objectives

The primary aim of the YANG modules contained in this draft is to provide the core model that is required to implement VLAN transport services on router based devices that is fully compatible with IEEE 802.1Q complaint bridges.

A secondary aim is for the modules to be structured in such a way that they can be cleanly extended in future.

2.1. Interoperability with IEEE 802.1Q compliant bridges

The modules defined in this document are designed to fully interoperate with IEEE 802.1Q compliant bridges. In particular, the models are restricted to only matching, adding, or rewriting the 802.1Q VLAN tags in frames in ways that are compatible with IEEE 802.1Q compliant bridges.

2.2. Extensibility

The modules are structured in such a way that they can be sensibly extended. In particular:

The tag stack is represented as a list to allow a tag stack of more than two tags to be supported if necessary in future.

The tag stack list elements allow other models, or vendors, to include additional forms of tag matching and rewriting. The

intention, however, is that it should not be necessary to have any vendor specific extensions to any of the YANG models defined in this document to implement standard Ethernet and VLAN services.

3. L3 Interface VLAN Model

The L3 Interface VLAN model provides appropriate leaves for termination of an 802.1Q VLAN tagged segment to a sub-interface based L3 service. It allows for termination of traffic with up to two 802.1Q VLAN tags.

The "if-l3-vlan" YANG module has the following structure:

```
augment /if:interfaces/if:interface/if-cmn:encapsulation/
  if-cmn:encaps-type:
   +--:(vlan)
      +--rw vlan
         +--rw tags
            +--rw tag* [index]
               +--rw index        uint8
               +--rw dot1q-tag
                  +--rw tag-type    dot1q-tag-type
                  +--rw vlan-id     dot1q-vlan-id
```

4. Flexible Encapsulation Model

The Flexible Encapsulation model is designed to allow for the flexible provisioning of layer 2 services. It provides the capability to classify Ethernet/VLAN frames received on an Ethernet trunk interface to sub-interfaces based on the fields available in the layer 2 headers. Once classified to sub-interfaces, it provides the capability to selectively modify fields within the layer 2 headers before the frame is handed off to the appropriate forwarding code for further handling.

The model supports a common core set of layer 2 header matches based on the 802.1Q tag type and VLAN Ids contained within the header up to a tag stack depth of two tags.

The model supports flexible rewrites of the layer 2 frame header for data frames as they are processed on the interface. It defines a set of standard tag manipulations that allow for the insertion, removal, or rewrite of one or two 802.1Q VLAN tags. The expectation is that manipulations are generally implemented in a symmetrical fashion, i.e. if a manipulation is performed on ingress traffic on an interface then the reverse manipulation is always performed on egress

traffic out of the same interface. However, the model also allows for asymmetrical rewrites, which may be required to implement some forwarding models (such as E-Tree).

The structure of the model is currently limited to matching or rewriting a maximum of two 802.1Q tags in the frame header but has been designed to be easily extensible to matching/rewriting three or more VLAN tags in future, if required.

The final aim for the model design is for it to be cleanly extensible to add in additional match and rewrite criteria of the layer 2 header, such as matching on the source or destination MAC address, PCP or DEI fields in the 802.1Q tags, or the EtherType of the frame payload. Rewrites can also be extended to allow for modification of other fields within the layer 2 frame header.

The "flexible-encapsulation" YANG module has the following structure:

```
augment /if:interfaces/if:interface/if-cmn:encapsulation/
  if-cmn:encaps-type:
   +--:(flexible) {flexible-encapsulation-rewrites}?
      +--rw flexible
         +--rw match
         |  +--rw (match-type)
         |     +--:(default)
         |     |  +--rw default?           empty
         |     +--:(untagged)
         |     |  +--rw untagged?          empty
         |     +--:(priority-tagged)
         |     |  +--rw priority-tagged
         |     |     +--rw tag-type?    dot1q:dot1q-tag-type
         |     +--:(vlan-tagged)
         |        +--rw vlan-tagged
         |           +--rw tag* [index]
         |           |  +--rw index        uint8
         |           |  +--rw dot1q-tag
         |           |     +--rw tag-type    dot1q-tag-type
         |           |     +--rw vlan-id     union
         |           +--rw match-exact-tags?   empty
         +--rw rewrite {flexible-rewrites}?
            +--rw (direction)?
               +--:(symmetrical)
               |  +--rw symmetrical
               |     +--rw tag-rewrite {tag-rewrites}?
               |        +--rw pop-tags?    uint8
               |        +--rw push-tags* [index]
               |           +--rw index        uint8
```

```
                  |           +--rw dot1q-tag
                  |              +--rw tag-type    dot1q-tag-type
                  |              +--rw vlan-id     dot1q-vlan-id
                  +--:(asymmetrical) {asymmetric-rewrites}?
                     +--rw ingress
                     |  +--rw tag-rewrite {tag-rewrites}?
                     |     +--rw pop-tags?    uint8
                     |     +--rw push-tags* [index]
                     |        +--rw index        uint8
                     |        +--rw dot1q-tag
                     |           +--rw tag-type    dot1q-tag-type
                     |           +--rw vlan-id     dot1q-vlan-id
                     +--rw egress
                        +--rw tag-rewrite {tag-rewrites}?
                           +--rw pop-tags?    uint8
                           +--rw push-tags* [index]
                              +--rw index        uint8
                              +--rw dot1q-tag
                                 +--rw tag-type    dot1q-tag-type
                                 +--rw vlan-id     dot1q-vlan-id
augment /if:interfaces/if:interface:
   +--rw flexible-encapsulation
      +--rw local-traffic-default-encaps
         +--rw tag* [index]
            +--rw index        uint8
            +--rw dot1q-tag
               +--rw tag-type    dot1q-tag-type
               +--rw vlan-id     dot1q-vlan-id
```

5. L3 Interface VLAN YANG Module

This YANG module augments the encapsultion container defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

```
<CODE BEGINS> file "ietf-if-l3-vlan@2016-10-21.yang"
module ietf-if-l3-vlan {
  namespace "urn:ietf:params:xml:ns:yang:ietf-if-l3-vlan";
  prefix if-l3-vlan;

  import ietf-interfaces {
    prefix if;
  }

  import iana-if-type {
    prefix ianaift;
  }
```

```
     import ieee802-dot1q-types {
       prefix dot1q-types;
     }

     import ietf-interfaces-common {
       prefix if-cmn;
     }

     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: Lou Berger
                  <mailto:lberger@labn.net>

        WG Chair: Kent Watsen
                  <mailto:kwatsen@juniper.net>

        Editor:   Robert Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "This YANG module models L3 VLAN sub-interfaces";

     revision 2016-10-21 {
       description "Latest draft revision";

       reference "Internet-Draft draft-wilton-netmod-intf-vlan-yang-04";
     }

     feature l3-vlan-sub-interfaces {
       description
         "This feature indicates that the device supports L3 VLAN
          sub-interfaces";
     }

     /*
      * Add support for the 802.1Q VLAN encapsulation syntax on layer 3
      * terminated VLAN sub-interfaces.
      */
     augment "/if:interfaces/if:interface/if-cmn:encapsulation/" +
             "if-cmn:encaps-type" {
       when "../if:type = 'ianaift:l2vlan' and
             ../if-cmn:transport-layer = 'layer-3'" {
         description "Applies only to VLAN sub-interfaces that are
```

```
                operating at layer 3";
      }
      if-feature l3-vlan-sub-interfaces;
      description "Augment the generic interface encapsulation with an
                   encapsulation for layer 3 VLAN sub-interfaces";

      /*
       * Matches a VLAN, or pair of VLAN Ids to classify traffic
       * into an L3 service.
       */
      case vlan {
        container vlan {
          description
          "Match VLAN tagged frames with specific VLAN Ids";
          container tags {
            description "Matches frames tagged with specific VLAN Ids";
            list tag {
              must 'index != 0 or ' +
                   'count(../tag/index) != 2 or ' +
                   'dot1q-tag/tag-type = "s-vlan"' {
                error-message
                  "When matching two tags, the outer tag must be of
                   S-VLAN tag type";
                description
                  "For IEEE 802.1Q interoperability, when matching two
                   tags, it is required that the outer tag is an
                   S-VLAN, and the inner tag is a C-VLAN";
              }

              must 'index != 1 or ' +
                   'count(../tag/index) != 2 or ' +
                   'dot1q-tag/tag-type = "c-vlan"' {
                error-message
                  "When matching two tags, the inner tag must be of
                   C-VLAN tag type";
                description
                  "For IEEE 802.1Q interoperability, when matching two
                   tags, it is required that the outer tag is an
                   S-VLAN, and the inner tag is a C-VLAN";
              }

              key "index";
              min-elements 1;
              max-elements 2;

              description "The tags to match, with the outermost tag to
                           match with index 0";
              leaf index {
```

```
                type uint8 {
                  range "0..1";
                }

                /*
                 * Only allow matching on an inner tag (at index 1), if
                 * also matching on the outer tag at the same time.
                 */
                must ". = 0 or
                      count(../../tag[index = 0]/index) > 0" {
                  error-message
                    "An inner tag can only be matched on when also
                     matching on an outer tag";
                  description
                    "Only allow matching on an inner tag, if also
                     matching on the outer tag at the same time";
                }

                description
                  "The index into the tag stack, outermost tag first";
              }

              uses dot1q-types:dot1q-tag-classifier;
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

6. Flexible Encapsulation YANG Module

This YANG module augments the encapsultion container defined in Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

This YANG module also augments the interface container defined in [RFC7223].

```
<CODE BEGINS> file "ietf-flexible-encapsulation@2016-10-21.yang"
module ietf-flexible-encapsulation {
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-flexible-encapsulation";

  prefix flex;
```

```
  import ietf-interfaces {
    prefix if;
  }

  import ietf-interfaces-common {
    prefix if-cmn;
  }

  import ieee802-dot1q-types {
    prefix dot1q-types;
  }

  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: Lou Berger
               <mailto:lberger@labn.net>

     WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

     Editor:   Robert Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This YANG module describes interface configuration for flexible
     VLAN matches and rewrites.";

  revision 2016-10-21 {
    description "Latest draft revision";

    reference
      "Internet-Draft draft-wilton-netmod-intf-vlan-yang-04";
  }

  feature flexible-encapsulation-rewrites {
    description
      "This feature indicates whether the network element supports
       flexible Ethernet encapsulation that allows for matching VLAN
       ranges and performing independent tag manipulations";
  }

  feature flexible-rewrites {
    description
```

```
        "This feature indicates whether the network element supports
          specifying flexible rewrite operations";
      }

      feature asymmetric-rewrites {
        description
          "This feature indicates whether the network element supports
           specifying different rewrite operations for the ingress
           rewrite operation and egress rewrite operation.";
      }

      feature tag-rewrites {
        description
          "This feature indicates whether the network element supports
           the flexible rewrite functionality specifying flexible tag
           rewrites";
      }

      /*
       * flexible-match grouping.
       *
       * This grouping represents a flexible match.
       *
       * The rules for a flexible match are:
       *     1. default, untagged, priority tag, or a stack of tags.
       *   - Each tag in the stack of tags matches:
       *      1. tag type (802.1Q or 802.1ad) +
       *      2. tag value:
       *        i. single tag
       *        ii. set of tag ranges/values.
       *        iii. "any" keyword
       */
      grouping flexible-match {
        description "Flexible match";
        choice match-type {
          mandatory true;
          description "Provides a choice of how the frames may be
                       matched";

          case default {
            description "Default match";
            leaf default {
              type empty;
              description
                "Default match.  Matches all traffic not matched to any
                 other peer sub-interface by a more specific
                 encapsulation.";
            } // leaf default
```

```
        } // case default

        case untagged {
          description "Match untagged Ethernet frames only";
          leaf untagged {
            type empty;
            description
              "Untagged match.  Matches all untagged traffic.";
          } // leaf untagged
        } // case untagged

        case priority-tagged {
          description "Match priority tagged Ethernet frames only";

          container priority-tagged {
            description "Priority tag match";
            leaf tag-type {
              type dot1q-types:dot1q-tag-type;
              description "The 802.1Q tag type of matched priority
                           tagged packets";
            }
          }
        }

        case vlan-tagged {
          container vlan-tagged {
            description "Matches VLAN tagged frames";
            list tag {
              must 'index != 0 or ' +
                   'count(../tag/index) != 2 or ' +
                   'dot1q-tag/tag-type = "s-vlan"' {
                error-message
                  "When matching two tags, the outer tag must be of
                   S-VLAN tag type";
                description
                  "For IEEE 802.1Q interoperability, when matching two
                   tags, it is required that the outer tag is an
                   S-VLAN, and the inner tag is a C-VLAN";
              }

              must 'index != 1 or ' +
                   'count(../tag/index) != 2 or ' +
                   'dot1q-tag/tag-type = "c-vlan"' {
                error-message
                  "When matching two tags, the inner tag must be of
                   C-VLAN tag type";
                description
                  "For IEEE 802.1Q interoperability, when matching two
```

```
                    tags, it is required that the outer tag is an
                    S-VLAN, and the inner tag is a C-VLAN";
                }

                key "index";
                min-elements 1;
                max-elements 2;
                description "The tags to match, with the outermost tag to
                             match assigned index 0";
                leaf index {
                  type uint8 {
                    range "0..1";
                  }

                  must ". = 0 or
                        count(../../tag[index = 0]/index) > 0" {
                    error-message "An inner tag can only be matched on
                                   when also matching on an outer tag";
                    description "Only allow matching on an inner tag, if
                                 also matching on the outer tags at the
                                 same time";
                  }
                  description
                    "The index into the tag stack, outermost tag first";
                }

                uses dot1q-types:dot1q-tag-ranges-or-any-classifier;
              }

              leaf match-exact-tags {
                type empty;
                description
                  "If set, indicates that all 802.1Q VLAN tags in the
                   Ethernet frame header must be explicitly matched, i.e.
                   the EtherType following the matched tags must not be a
                   802.1Q tag EtherType.  If unset then extra 802.1Q VLAN
                   tags are allowed.";
              }
            }
          }
        } // encaps-type
      }

      /*
       * Grouping for tag-rewrite that can be expressed either
       * symmetrically, or in the ingress and/or egress directions
       * independently.
       */
```

```
     grouping tag-rewrite {
       description "Flexible rewrite";
       leaf pop-tags {
         type uint8 {
           range 1..2;
         }
         description "The number of tags to pop (or translate if used in
                      conjunction with push-tags)";
       }

       list push-tags {
         must 'index != 0 or ' +
              'count(../push-tags/index) != 2 or ' +
              'dot1q-tag/tag-type = "s-vlan"' {
           error-message
             "When pushing two tags, the outer tag must be of
              S-VLAN tag type";
           description
             "For IEEE 802.1Q interoperability, when pushing two
              tags, it is required that the outer tag is an
              S-VLAN, and the inner tag is a C-VLAN";
         }

         must 'index != 1 or ' +
              'count(../push-tags/index) != 2 or ' +
              'dot1q-tag/tag-type = "c-vlan"' {
           error-message
             "When pushing two tags, the inner tag must be of
              C-VLAN tag type";
           description
             "For IEEE 802.1Q interoperability, when pushing two
              tags, it is required that the outer tag is an
              S-VLAN, and the inner tag is a C-VLAN";
         }

         key "index";
         max-elements 2;
         description "The number of tags to push (or translate if used
                      in conjunction with pop-tags)";
         /*
          * Server should order by increasing index.
          */
         leaf index {
           type uint8 {
             range 0..1;
           }

           /*
```

```
            * Only allow a push of an inner tag if an outer tag is also
            * being pushed.
            */
           must ". != 0 or
                 count(../../push-tags[index = 0]/index) > 0" {
             error-message "An inner tag can only be pushed if an outer
                            tag is also specified";
             description "Only allow a push of an inner tag if an outer
                          tag is also being pushed";
           }

           description "The index into the tag stack";
         }

         uses dot1q-types:dot1q-tag-classifier;
       }
     }

     /*
      * Grouping for all flexible rewrites of fields in the L2 header.
      *
      * This currently only includes flexible tag rewrites, but is
      * designed to be extensible to cover rewrites of other fields in
      * the L2 header if required.
      */
     grouping flexible-rewrite {
       description "Flexible rewrite";

       /*
        * Tag rewrite.
        *
        * All tag rewrites are formed using a combination of pop-tags
        * and push-tags operations.
        */
       container tag-rewrite {
         if-feature tag-rewrites;
         description "Tag rewrite.  Translate operations are expressed
                      as a combination of tag push and pop operations.";
         uses tag-rewrite;
       }
     }
     augment "/if:interfaces/if:interface/if-cmn:encapsulation/" +
             "if-cmn:encaps-type" {
       when "../if:type = 'if-cmn:ethSubInterface' and
             ../if-cmn:transport-layer = 'layer-2'" {
         description "Applies only to Ethernet sub-interfaces that are
                      operating at transport layer 2";
       }
```

```
      description
        "Add flexible match and rewrite for VLAN sub-interfaces";

      /*
       * A flexible encapsulation allows for the matching of ranges and
       * sets of VLAN Ids.  The structure is also designed to be
       * extended to allow for matching/rewriting other fields within
       * the L2 frame header if required.
       */
      case flexible {
        if-feature flexible-encapsulation-rewrites;
        description "Flexible encapsulation and rewrite";
        container flexible {
          description "Flexible encapsulation and rewrite";

          container match {
            description
              "The match used to classify frames to this interface";
            uses flexible-match;
          }

          container rewrite {
            if-feature flexible-rewrites;
            description "L2 frame rewrite operations";
            choice direction {
              description "Whether the rewrite policy is symmetrical or
                           asymmetrical";
              case symmetrical {
                container symmetrical {
                  uses flexible-rewrite;
                  description
                    "Symmetrical rewrite.  Expressed in the ingress
                     direction, but the reverse operation is applied
                     to egress traffic";
                }
              }

              /*
               * Allow asymmetrical rewrites to be specified.
               */
              case asymmetrical {
                if-feature asymmetric-rewrites;
                description "Asymmetrical rewrite";
                container ingress {
                  uses flexible-rewrite;
                  description "Ingress rewrite";
                }
                container egress {
```

```
                  uses flexible-rewrite;
                  description "Egress rewrite";
                }
              }
            }
          }
        }
      }
    }

    augment "/if:interfaces/if:interface" {
      when "if:type = 'if-cmn:ethSubInterface' and
            if-cmn:transport-layer = 'layer-2'" {
        description "Any L2 Ethernet sub-interfaces";
      }
      description "Add flexible encapsulation configuration for VLAN
                   sub-interfaces";

      /*
       * All flexible encapsulation specific interface configuration
       * (except for the actual encapsulation and rewrite) is contained
       * by a flexible-encapsulation container on the interface.
       */
      container flexible-encapsulation {
        description
          "All per interface flexible encapsulation related fields";

        /*
         * For encapsulations that match a range of VLANs (or Any),
         * allow configuration to specify the default VLAN tag values
         * to use for any traffic that is locally sourced from an
         * interface on the device.
         */
        container local-traffic-default-encaps {
          description "The VLAN tags to use by default for locally
                       sourced traffic";
          list tag {
            key "index";
            max-elements 2;

            description
              "The VLAN tags to use by locally sourced traffic";

            leaf index {
              type uint8 {
                range "0..1";
              }
```

```
              /*
               * Only allow an inner tag to be specified if an outer
               * tag has also been specified.
               */
              must ". = 0 or
                    count(../../tag[index = 0]/index) > 0" {
                error-message "An inner tag can only be specified if an
                               outer tag has also been specified";
                description "Ensure that an inner tag cannot be
                             specified without an outer tag'";
              }

              description "The index into the tag stack, outermost tag
                           assigned index 0";
            }

            uses dot1q-types:dot1q-tag-classifier;
          }
        }
      }
    }
  }
  <CODE ENDS>
```

7. Acknowledgements

The authors would particularly like to thank John Messenger, Glenn Parsons, and Dan Romascanu for their help progressing this draft.

The authors would also like to thank Eric Gray, Marc Holness, Neil Ketley, William Lupton, John Messenger, Glenn Parsons, Ludwig Pauwels, and members of the IEEE 802.1 WG for their helpful feedback on this draft.

8. ChangeLog

8.1. Version -04

- IEEE 802.1 specific types have been removed from the draft. These are now referenced from the 802.1Qcp draft YANG modules.

- Fixed errors in the xpath expressions.

8.2. Version -03

- Incorporates feedback received from presenting to the IEEE 802.1 WG.
- Updates the modules for double tag matches/rewrites to restrict the outer tag type to S-VLAN and inner tag type to C-VLAN.
- Updates the introduction to indicate primary use case is for IETF forwarding protocols.
- Updates the objectives to make IEEE 802.1Q bridge interoperability a key objective.

9. IANA Considerations

This document defines several new YANG module and the authors politely request that IANA assigns unique names to the YANG module files contained within this draft, and also appropriate URIs in the "IETF XML Registry".

10. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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:

10.1. if-l3-vlan.yang

The nodes in the if-l3-vlan YANG module are concerned with matching particular frames received on the network device to connect them to a layer 3 forwarding instance, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be lost because it is not being classified correctly, or is being classified to a separate sub-interface. The nodes, all under the subtree

/interfaces/interface/encapsulation/vlan, that are sensitive to this are:

- tags
- tags/index
- tags/index/tag-type
- tags/index/vlan-id

10.2. flexible-encapsulation.yang

There are many nodes in the flexible-encapsulation YANG module that are concerned with matching particular frames received on the network device, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be lost because it is not being classified correctly, or is being classified to a separate sub-interface. The nodes, all under the subtree /interfaces/interface/encapsulation/flexible/match, that are sensitive to this are:

- default
- untagged
- priority-tagged
- priority-tagged/tag-type
- vlan-tagged
- vlan-tagged/index
- vlan-tagged/index/dot1q-tag/vlan-type
- vlan-tagged/index/dot1q-tag/vlan-id
- vlan-tagged/match-exact-tags

There are also many modes in the flexible-encapsulation YANG module that are concerned with rewriting the fields in the L2 header for particular frames received on the network device, and as such adding/modifying/deleting these nodes has a high risk of causing traffic to be dropped or incorrectly processed on peer network devices, or it could cause layer 2 tunnels to go down due to a mismatch in negotiated MTU. The nodes, all under the subtree

/interfaces/interface/encapsulation/flexible/rewrite, that are sensitive to this are:

- symmetrical/tag-rewrite/pop-tags
- symmetrical/tag-rewrite/push-tags
- symmetrical/tag-rewrite/push-tags/index
- symmetrical/tag-rewrite/push-tags/dot1q-tag/tag-type
- symmetrical/tag-rewrite/push-tags/dot1q-tag/vlan-id
- asymmetrical/ingress/tag-rewrite/pop-tags
- asymmetrical/ingress/tag-rewrite/push-tags
- asymmetrical/ingress/tag-rewrite/push-tags/index
- asymmetrical/ingress/tag-rewrite/push-tags/dot1q-tag/tag-type
- asymmetrical/ingress/tag-rewrite/push-tags/dot1q-tag/vlan-id
- asymmetrical/egress/tag-rewrite/pop-tags
- asymmetrical/egress/tag-rewrite/push-tags
- asymmetrical/egress/tag-rewrite/push-tags/index
- asymmetrical/egress/tag-rewrite/push-tags/dot1q-tag/tag-type
- asymmetrical/egress/tag-rewrite/push-tags/dot1q-tag/vlan-id

Nodes in the flexible-encapsulation YANG module that are concerned with the VLAN tags to use for traffic sourced from the network element could cause protocol sessions (such as CFM) to fail if they are added, modified or deleted. The nodes, all under the subtree /interfaces/interface/flexible-encapsulation/local-traffic-default-encaps that are sensitive to this are:

- tag
- tag/index
- tag/dot1q-tag/tag-type
- tag/dot1q-tag/vlan-id

11. References

11.1. Normative References

[I-D.ietf-netmod-intf-ext-yang]
Wilton, R., Ball, D., tsingh@juniper.net, t., and S. Sivaraj, "Common Interface Extension YANG Data Models", draft-ietf-netmod-intf-ext-yang-01 (work in progress), July 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <http://www.rfc-editor.org/info/rfc7224>.

11.2. Informative References

[dot1Qcp] Holness, M., "802.1Qcp Bridges and Bridged Networks - Amendment: YANG Data Model", 2016.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <http://www.rfc-editor.org/info/rfc4448>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <http://www.rfc-editor.org/info/rfc4761>.

[RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <http://www.rfc-editor.org/info/rfc4762>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

Appendix A. Comparison with the IEEE 802.1Q Configuration Model

In addition to the sub-interface based YANG model proposed here, the IEEE 802.1Q working group is also developing a YANG model for the configuration of 802.1Q VLANs. This raises the valid question as to whether the models overlap and whether it is necessary or beneficial to have two different models for superficially similar constructs. This section aims to answer that question by summarizing and comparing the two models.

A.1. Sub-interface based configuration model overview

The key features of the sub-interface based configuration model can be summarized as:

- The model is primarily designed to enable layer 2 and layer 3 services on Ethernet interfaces that can be defined in a very flexible way to meet the varied requirements of service providers.

- Traffic is classified from an Ethernet-like interface to sub-interfaces based on fields in the layer 2 header. This is often based on VLAN Ids contained in the frame, but the model is extensible to other arbitrary fields in the frame header.

- Sub-interfaces are just a type of if:interface and hence support any feature configuration YANG models that can be applied generally to interfaces. For example, QoS or ACL models that reference if:interface can be applied to the sub-interfaces, or

the sub-interface can be used as an Access Circuit in L2VPN or L3VPN models that reference if:interface.

- In the sub-interface based configuration model, the classification of traffic arriving on an interface to a given sub-interface, based on fields in the layer 2 header, is completely independent of how the traffic is forwarded. The sub-interface can be referenced (via references to if:interface) by other models that specify how traffic is forwarded; thus sub-interfaces can support multiple different forwarding paradigms, including but not limited to: layer 3 (IPv4/IPv6), layer 2 pseudowires (over MPLS or IP), VPLS instances, EVPN instance.

- The model is flexible in the scope of the VLAN Identifier space. I.e. by default VLAN Ids can be scoped locally to a single Ethernet-like trunk interface, but the scope is determined by the forwarding paradigm that is used.

A.2. IEEE 802.1Q Bridge Configuration Model Overview

The key features of the IEEE 802.1Q bridge configuration model can be summarized as:

- Each VLAN bridge component has a set of Ethernet interfaces that are members of that bridge. Sub-interfaces are not used, nor required in the 802.1Q bridge model.

- Within a VLAN bridge component, the VLAN tag in the packet is used, along with the destination MAC address, to determine how to forward the packet. Other forwarding paradigms are not supported by the 802.1Q model.

- Classification of traffic to a VLAN bridge component is based only on the Ethernet interface that it arrived on.

- VLAN Identifiers are scoped to a VLAN bridge component. Often devices only support a single bridge component and hence VLANs are scoped globally within the device.

- Feature configuration is specified in the context of the bridge, or particular VLANs on a bridge.

A.3. Possible Overlap Between the Two Models

Both models can be used for configuring similar basic layer 2 forwarding topologies. The 802.1Q bridge configuration model is optimised for configuring Virtual LANs that span across enterprises and data centers.

The sub-interface model can also be used for configuring equivalent Virtual LAN networks that span across enterprises and data centers, but often requires more configuration to be able to configure the equivalent constructs to the 802.1Q bridge model.

The sub-interface model really excels when implementing flexible L2 and L3 services, where those services may be handled on the same physical interface, and where the VLAN Identifier is being solely used to identify the customer or service that is being provided rather than a Virtual LAN. The sub-interface model provides more flexibility as to how traffic can be classified, how features can be applied to traffic streams, and how the traffic is to be forwarded.

Conversely, the 802.1Q bridge model can also be use to implement L2 services in some scenarios, but only if the forwarding paradigm being used to implement the service is the native Ethernet forwarding specified in 802.1Q - other forwarding paradigms such as pseudowires or VPLS are not supported. The 802.1Q bridge model does not implement L3 services at all, although this can be partly mitigated by using a virtual L3 interface construct that is a separate logical Ethernet-like interface which is a member of the bridge.

In conclusion, it is valid for both of these models to exist since they have different deployment scenarios for which they are optimized. Devices may choose which of the models (or both) to implement depending on what functionality the device is being designed for.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@cisco.com

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net