CCAMP Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 April 2024

J. Ahlberg
Ericsson AB
S. Mansfield
Ericsson Inc
M. Ye
I. Busi
Huawei Technologies
X. Li
NEC Laboratories Europe
D. Spreafico
Nokia - IT

18 October 2023

A YANG Data Model for Interface Reference Topology
draft-ietf-ccamp-if-ref-topo-yang-01

Abstract

This document defines a YANG data model to provide a reference from a termination point in a topology model to interface management information.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://github.com/ietf-ccamp-wg/draft-ietf-ccamp-mw-topo-yang. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-ccamp-if-ref-topo-yang/.

Discussion of this document takes place on the CCAMP Working Group mailing list (mailto:ccamp@ietf.org), which is archived at https://datatracker.ietf.org/wg/ccamp/about/. Subscribe at https://www.ietf.org/mailman/listinfo/ccamp/.

Source for this draft and an issue tracker can be found at https://github.com/https://github.com/ietf-ccamp-wg/draft-ietf-ccamp-mw-topo-yang.

Status of This Memo

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

This Internet-Draft will expire on 20 April 2024.

Table of Contents

1. Introduction

This document defines a YANG data model to provide a reference from a termination point in a topology model to interface management information. It introduces a way to reference the information in a YANG data model for interface management [RFC8343] that could be useful for all types of termination points. The model augments "YANG Data Model for Traffic Engineering (TE) Topologies" defined in [RFC8795], which is based on "A YANG Data Model for Network Topologies" defined in [RFC8345].

The interface reference model is expected to be used between a Provisioning Network Controller (PNC) and a Multi Domain Service Coordinator(MDSC) [RFC8453]. Different use cases require access to different attributes and in order not to restrict what use cases can be supported, all attributes supported by the interface management model is with this model made accessible from the topology model.

1.1. Terminology and Definitions

The following acronyms are used in this document:

PNC Provisioning Network Controller

MDSC Multi Domain Service Coordinator

1.2. Tree Structure

A simplified graphical representation of the data model is used in chapter 3.1 of this document. The meaning of the symbols in these diagrams is defined in [RFC8340].

2. 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.

3. Termination Point to Interface Reference YANG Data Model

3.1. YANG Tree

```
module: ietf-tp-interface-reference-topology

augment /nw:networks/nw:network/nw:node/nt:termination-point/tet:te:
  +--rw tp-to-interface-path?   -> /if:interfaces/interface/name
```

3.2. Termination Point to Interface Reference YANG Data Module

```
<CODE BEGINS> file "ietf-tp-interface-reference-topology.yang"
module ietf-tp-interface-reference-topology {
  yang-version "1.1";
   namespace
  "urn:ietf:params:xml:ns:yang:ietf-tp-interface-reference-topology";

   prefix "ifref";

   import ietf-network {
     prefix "nw";
     reference "RFC 8345: A YANG Data Model for Network Topologies";
   }

   import ietf-network-topology {
     prefix "nt";
     reference "RFC 8345: A YANG Data Model for Network Topologies";
   }

   import ietf-te-topology {
     prefix "tet";
     reference "RFC 8795: YANG Data Model for Traffic Engineering
                (TE) Topologies";
   }

   import ietf-interfaces {
     prefix if;
     reference
       "RFC 8343";
   }

   organization
     "Internet Engineering Task Force (IETF) CCAMP WG";
   contact
     "WG List: <mailto:ccamp@ietf.org>

      Editor: Jonas Ahlberg
              <mailto:jonas.ahlberg@ericsson.com>
      Editor: Scott Mansfield
              <mailto:scott.mansfield@ericsson.com>
      Editor: Min Ye
              <mailto:amy.yemin@huawei.com>
      Editor: Italo Busi
              <mailto:Italo.Busi@huawei.com>
      Editor: Xi Li
              <mailto:Xi.Li@neclab.eu>
      Editor: Daniela Spreafico
```

```
                <mailto:daniela.spreafico@nokia.com>
      ";

    description
      "This is a module for defining a reference from a termination
       point in a te topology to a list element in interfaces
       as defined in RFC 8343.

      Copyright (c) 2023 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 Revised 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).

      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 2023-02-15 {
      description
      "First rough draft.";
      reference "";
    }

    /*
     * Groupings
     */
    grouping tp-to-interface-ref {

      description
        "Grouping used for reference between a termination point and
         an interface.";
      leaf tp-to-interface-path {
        type leafref {
          path '/if:interfaces/if:interface/if:name';
        }
        description
          "Leafref expression referencing a list element, identified
           by its name, in interfaces as defined in RFC 8343.";
      }
    }

    /*
     * Data nodes
```

```
       */

      augment "/nw:networks/nw:network/nw:node/nt:termination-point/"
              + "tet:te" {
        description
          "Augmentation to add possibility to reference an element
           in the list of interfaces as defined by RFC 8343.";
        uses tp-to-interface-ref;
      }
    }
```

<CODE ENDS>

4. Security Considerations

The YANG modules specified in this document define schemas 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 [RFC8446].

The NETCONF 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.

The YANG module specified in this document imports and augments the ietf-network and ietf-network-topology models defined in [RFC8345]. The security considerations from [RFC8345] are applicable to the module in this document.

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

* tp-to-interface-path: A malicious client could set an arbitrary path that could allow a client to retrieve incorrect information. Troubleshooting would be difficult because the bad path would not be detectable until the client tries to use the leaf to identify to radio link terminal.

5. IANA Considerations

IANA is asked to assign a new URI from the "IETF XML Registry" [RFC3688] as follows:

URI: urn:ietf:params:xml:ns:yang:ietf-tp-interface-reference-topology
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.

It is proposed that IANA should record YANG module names in the "YANG Module Names" registry [RFC6020] as follows:

```
Name: ietf-tp-interface-reference-topology
Maintained by IANA?: N
Namespace:
  urn:ietf:params:xml:ns:yang:ietf-interface-reference-topology
Prefix: ifref
Reference: RFC XXXX
```

6. References

6.1. 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, <https://www.rfc-editor.org/rfc/rfc2119>.

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

[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/rfc/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/rfc/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/rfc/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/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/rfc/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/rfc/rfc8341>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/rfc/rfc8343>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/rfc/rfc8345>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8795] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Gonzalez de Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", RFC 8795, DOI 10.17487/RFC8795, August 2020, <https://www.rfc-editor.org/rfc/rfc8795>.

6.2. Informative References

[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/rfc/rfc8340>.

[RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <https://www.rfc-editor.org/rfc/rfc8453>.

Appendix A. Examples of the Interface Reference Topology Model

This appendix provides some examples and illustrations of how the Interface Reference Topology Model can be used. There is one extended tree to illustrate the Model and a JSON based instantiation of the Interface Reference Model for a small network example.

A.1. A tree for a complete Interface Reference Topology Model

The tree below shows the leafs for extending a Network Topology Model defined in [RFC8345], Traffic Engineering (TE) Topologies model defined in [RFC8795] with a possibility to reference interface management information.

```
module: ietf-network
  +--rw networks
     +--rw network* [network-id]
        +--rw network-id                 network-id
        +--rw node* [node-id]
           +--rw node-id                node-id
           +--rw nt:termination-point* [tp-id]
              +--rw nt:tp-id            tp-id
              +--rw tet:te!
                 +--rw ifref:tp-to-interface-path?
                   -> /if:interfaces/interface/name
```

A.2. A JSON example

```
{
  "ietf-interfaces:interfaces": {
    "interface": [
      {
        "name": "L2Interface1",
        "description": "'Ethernet Interface 1'",
        "type": "iana-if-type:ethernetCsmacd"
      },
      {
        "name": "L2Interface2",
        "description": "'Ethernet Interface 2'",
        "type": "iana-if-type:ethernetCsmacd"
      }
    ]
  },
  "ietf-network:networks": {
    "network": [
      {
        "network-id": "L2-network",
        "network-types": {
```

```
            "ietf-te-topology:te-topology": {
              "ietf-eth-te-topology:eth-tran-topology": {}
            }
          },
          "node": [
            {
              "node-id": "L2-N1",
              "ietf-network-topology:termination-point": [
                {
                  "tp-id": "L2-N1-TP1",
                  "ietf-te-topology:te-tp-id": "10.10.10.1",
                  "ietf-te-topology:te": {
    "ietf-tp-interface-reference-topology:tp-to-interface-path":
       "L2Interface1"
                  }
                }
              ]
            },
            {
              "node-id": "L2-N2",
              "ietf-network-topology:termination-point": [
                {
                  "tp-id": "L2-N2-TP2",
                  "ietf-te-topology:te-tp-id": "10.10.10.2",
                  "ietf-te-topology:te": {
    "ietf-tp-interface-reference-topology:tp-to-interface-path":
       "L2Interface2"
                  }
                }
              ]
            }
          ],
          "ietf-network-topology:link": [
            {
              "link-id": "L2-N1-N2",
              "source": {
                "source-node": "L2-N1",
                "source-tp": "L2-N1-TP1"
              },
              "destination": {
                "dest-node": "L2-N2",
                "dest-tp": "L2-N2-TP2"
              }
            }
          ]
        }
      ]
    }
```

```
   }
```

Acknowledgments

This document was prepared using kramdown

The authors would like to thank ...

Authors' Addresses

Jonas Ahlberg
Ericsson AB
Lindholmspiren 11
SE-417 56 Goteborg
Sweden
Email: jonas.ahlberg@ericsson.com

Scott Mansfield
Ericsson Inc
Email: scott.mansfield@ericsson.com

Min Ye
Huawei Technologies
No.1899, Xiyuan Avenue
Chengdu
611731
China
Email: amy.yemin@huawei.com

Italo Busi
Huawei Technologies
Email: italo.busi@huawei.com

Xi Li
NEC Laboratories Europe
Kurfursten-Anlage 36
69115 Heidelberg
Germany
Email: Xi.Li@neclab.eu

Daniela Spreafico
Nokia - IT
Via Energy Park, 14
20871 Vimercate (MI)
Italy
Email: daniela.spreafico@nokia.com

netmod
Internet-Draft
Intended status: Standards Track
Expires: 19 April 2024

O. G. D. Dios
S. Barguil
Telefonica
M. Boucadair
Orange
Q. Wu
Huawei

17 October 2023

Extensions to the Access Control Lists (ACLs) YANG Model
draft-ietf-netmod-acl-extensions-03

Abstract

RFC 8519 defines a YANG data model for Access Control Lists (ACLs). This document discusses a set of extensions that fix many of the limitations of the ACL model as initially defined in RFC 8519.

The document also defines IANA-maintained modules for ICMP types and IPv6 extension headers.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Network Modeling Working Group mailing list (netmod@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/netmod/.

Source for this draft and an issue tracker can be found at https://github.com/boucadair/enhanced-acl-netmod.

Status of This Memo

This Internet-Draft will expire on 19 April 2024.

Table of Contents

1. Introduction

[RFC8519] defines Access Control Lists (ACLs) as a user-ordered set of filtering rules. The model targets the configuration of the filtering behavior of a device. However, the model structure, as defined in [RFC8519], suffers from a set of limitations. This document describes these limitations and proposes an enhanced ACL structure. The YANG module in this document is solely based on augmentations to the ACL YANG module defined in [RFC8519].

The motivation of such enhanced ACL structure is discussed in detail in Appendix D.

When managing ACLs, it is common for network operators to group match elements in pre-defined sets. The consolidation into group matches allows for reducing the number of rules, especially in large scale networks. If, for example, it is needed to find a match against 100 IP addresses (or prefixes), a single rule will suffice rather than creating individual Access Control Entries (ACEs) for each IP address (or prefix). In doing so, implementations would optimize the performance of matching lists vs multiple rules matching.

The enhanced ACL structure is also meant to facilitate the management of network operators. Instead of entering the IP address or port number literals, using user-named lists decouples the creation of the rule from the management of the sets. Hence, it is possible to remove/add entries to the list without redefining the (parent) ACL rule.

In addition, the notion of Access Control List (ACL) and defined sets is generalized so that it is not device-specific as per [RFC8519]. ACLs and defined sets may be defined at network / administrative domain level and associated to devices. This approach facilitates the reusability across multiple network elements. For example, managing the IP prefix sets from a network level makes it easier to maintain by the security groups.

Network operators maintain sets of IP prefixes that are related to each other, e.g., deny-lists or accept-lists that are associated with those provided by a VPN customer. These lists are maintained and manipulated by security expert teams.

Note that ACLs are used locally in devices but are triggered by other tools such as DDoS mitigation [RFC9132] or BGP Flow Spec [RFC8955] [RFC8956]. Therefore, supporting means to easily map to the filtering rules conveyed in messages triggered by these tools is valuable from a network operation standpoint.

The document also defines IANA-maintained modules for ICMP types and IPv6 extension headers. The design of the modules adheres to the recommendations in [I-D.boucadair-netmod-rfc8407bis]. The templates to generate the modules is available at Appendix A.1, Appendix B.1, and Appendix C.1. Readers should refer to the IANA websites [REF_TBC], [REF_TBC2], and [REF_TBC3] to retrieve the latest version of the modules. The modules are provided in Appendix A.2, Appendix B.2, and Appendix C.2 for the users convenience, but these appendices will be removed from the final RFC.

2. 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.

The terminology for describing YANG modules is defined in [RFC7950]. The meaning of the symbols in the tree diagrams is defined in [RFC8340].

In addition to the terms defined in [RFC8519], this document makes use of the following term:

Defined set: :Refers to reusable description of one or multiple information elements (e.g., IP address, IP prefix, port number, or ICMP type).

3. Overall Structure of The Enhanced ACL Module

3.1. Tree Structure

Figure 1 shows the full enhanced ACL tree:

```
module: ietf-acl-enh
  +--rw defined-sets
  |  +--rw ipv4-prefix-sets
  |  |  +--rw prefix-set* [name]
  |  |     +--rw name           string
  |  |     +--rw description?   string
  |  |     +--rw prefix*        inet:ipv4-prefix
  |  +--rw ipv6-prefix-sets
  |  |  +--rw prefix-set* [name]
  |  |     +--rw name           string
  |  |     +--rw description?   string
  |  |     +--rw prefix*        inet:ipv6-prefix
  |  +--rw port-sets
  |  |  +--rw port-set* [name]
  |  |     +--rw name    string
  |  |     +--rw port* [id]
  |  |        +--rw id                                string
  |  |        +--rw (port)?
  |  |           +--:(port-range-or-operator)
  |  |              +--rw port-range-or-operator
  |  |                 +--rw (port-range-or-operator)?
  |  |                    +--:(range)
  |  |                    |  +--rw lower-port    inet:port-number
  |  |                    |  +--rw upper-port    inet:port-number
  |  |                    +--:(operator)
  |  |                       +--rw operator?     operator
  |  |                       +--rw port          inet:port-number
  |  +--rw protocol-sets
  |  |  +--rw protocol-set* [name]
  |  |     +--rw name        string
  |  |     +--rw protocol*   union
  |  +--rw icmpv4-type-sets
  |  |  +--rw icmpv4-type-set* [name]
  |  |     +--rw name     string
  |  |     +--rw types* [type]
```

```
   |  |        +--rw type              iana-icmpv4-types:icmpv4-type
   |  |        +--rw code?             uint8
   |  |        +--rw rest-of-header?   binary
   |  +--rw icmpv6-type-sets
   |     +--rw icmpv6-type-set* [name]
   |        +--rw name     string
   |        +--rw types* [type]
   |           +--rw type              iana-icmpv6-types:icmpv6-type
   |           +--rw code?             uint8
   |           +--rw rest-of-header?   binary
   +--rw aliases
      +--rw alias* [name]
         +--rw name          string
         +--rw vlan*         uint16
         +--rw prefix*       inet:ip-prefix
         +--rw port-range* [lower-port]
         |  +--rw lower-port    inet:port-number
         |  +--rw upper-port?   inet:port-number
         +--rw protocol*     uint8
         +--rw fqdn*         inet:domain-name
         +--rw uri*          inet:uri

   augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
     +--rw (payload)?
     |  +--:(prefix-pattern)
     |     +--rw prefix-pattern {match-on-payload}?
     |        +--rw offset?       identityref
     |        +--rw offset-end?   uint64
     |        +--rw operator?     operator
     |        +--rw prefix?       binary
     +--rw (alias)?
     |  +--:(alias-name)
     |     +--rw alias-name*       alias-ref
     +--rw (mpls)?
        +--:(mpls-values)
           +--rw mpls-values {match-on-mpls}?
              +--rw traffic-class?       uint8
              +--rw label-position?      identityref
              +--rw upper-label-range?   rt-types:mpls-label
              +--rw lower-label-range?   rt-types:mpls-label
              +--rw label-block-name?    string
              +--rw ttl-value?           uint8
   augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l2:
     +--rw vlan-filter {match-on-vlan-filter}?
     |  +--rw frame-type?         string
     |  +--rw (vlan-type)?
     |     +--:(range)
     |     |  +--rw lower-vlan    uint16
```

```
    |      |  +--rw upper-vlan    uint16
    |      +--:(operator)
    |         +--rw operator?     packet-fields:operator
    |         +--rw vlan*         uint16
    +--rw isid-filter {match-on-isid-filter}?
       +--rw (isid-type)?
          +--:(range)
          |  +--rw lower-isid    uint16
          |  +--rw upper-isid    uint16
          +--:(operator)
             +--rw operator?     packet-fields:operator
             +--rw isid*         uint16
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3
          /acl:ipv4/acl:ipv4:
    +--rw ipv4-fragment
    |  +--rw operator?   operator
    |  +--rw type?       fragment-type
    +--rw source-ipv4-prefix-list?        ipv4-prefix-set-ref
    +--rw destination-ipv4-prefix-list?   ipv4-prefix-set-ref
    +--rw next-header-set?                protocol-set-ref
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3
          /acl:ipv6/acl:ipv6:
    +--rw ipv6-fragment
    |  +--rw operator?   operator
    |  +--rw type?       fragment-type
    +--rw source-ipv6-prefix-list?        ipv6-prefix-set-ref
    +--rw destination-ipv6-prefix-list?   ipv6-prefix-set-ref
    +--rw protocol-set?                   protocol-set-ref
    +--rw extension-header?
            iana-ipv6-ext-types:ipv6-extension-header-type
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4/acl:tcp:
    +--rw flags-bitmask
    |  +--rw (mode)?
    |     +--:(explicit)
    |     |  +--rw operator?             operator
    |     |  +--rw explicit-tcp-flag*   identityref
    |     +--:(builtin)
    |        +--rw bitmask?             uint16
    +--rw source-tcp-port-set?        port-set-ref
    +--rw destination-tcp-port-set?   port-set-ref
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4/acl:udp:
    +--rw source-udp-port-set?        port-set-ref
    +--rw destination-udp-port-set?   port-set-ref
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4/acl:icmp:
    +--rw icmpv4-set?   icmpv4-type-set-ref
    +--rw icmpv6-set?   icmpv6-type-set-ref
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:actions:
    +--rw log-action
```

```
    |  +--rw log-type?   identityref
    |  +--rw log-id?     string
    +--rw counter-action
    |  +--rw counter-type?   identityref
    |  +--rw counter-name*   string
    +--rw rate-limit?       decimal64
```

Figure 1: Enhanced ACL tree

3.2. Defined Sets

The augmented ACL structure includes several containers to manage reusable sets of elements that can be matched in an ACL entry. Each set is uniquely identified by a name and can be called from the relevant entry. The following sets are defined:

* IPv4 prefix set: It contains a list of IPv4 prefixes. A match will be considered if the IP address (source or destination, depending on the ACL entry) is contained in any of the prefixes.

* IPv6 prefix set: It contains a list of IPv6 prefixes. A match will be considered if the IP address (source or destination, depending on the ACL entry) is contained in any of the prefixes.

* Port sets: It contains a list of port numbers to be used in TCP/UDP entries. The port numbers can be individual port numbers, a range of ports, and an operation.

* Protocol sets: It contains a list of protocol values. Each protocol can be identified either by a number (e.g., 17) or a name (e.g., UDP).

* ICMP sets: It contains a list of ICMPv4 or ICMPv6 types, each of them identified by a type value, optionally the code and the rest of the header.

* Aliases: An alias is defined by a combination of various parameters (e.g., IP prefix, protocol, port number, or VLAN). Sets of aliases can be defined and referred to in match criteria.

3.3. IPv6 Extension Headers

The module can be used to manage ACLs that require matching against IPv6 extension headers. To that aim, a new IANA-maintained module is defined in in this document.

3.4. TCP Flags Handling

The augmented ACL structure includes a new leaf 'flags-bitmask' to better handle flags.

Clients that support both 'flags-bitmask' and 'flags' matching fields MUST NOT set these fields in the same request.

Figure 2 shows an example of a request to install a filter to discard incoming TCP messages having all flags unset.

```
{
   "ietf-access-control-list:acls":{
      "acl":[
         {
            "name":"tcp-flags-example",
            "aces":{
               "ace":[
                  {
                     "name":"null-attack",
                     "matches":{
                        "tcp":{
                           "acl-enh:flags-bitmask":{
                              "operator":"not any",
                              "bitmask":4095
                           }
                        }
                     },
                     "actions":{
                        "forwarding":"drop"
                     }
                  }
               ]
            }
         }
      ]
   }
}
```

Figure 2: Example of an ACL to Deny TCP Null Attack Messages (Request Body)

3.5. Fragments Handling

The augmented ACL structure includes a new leaf 'fragment' to better handle fragments.

Clients that support both 'fragment' and 'flags' matching fields MUST NOT set these fields in the same request.

Figure 3 shows the content of a POST request to allow the traffic destined to 198.51.100.0/24 and UDP port number 53, but to drop all fragmented packets. The following ACEs are defined (in this order):

* "drop-all-fragments" ACE: discards all fragments.

* "allow-dns-packets" ACE: accepts DNS packets destined to 198.51.100.0/24.

```
{
   "ietf-access-control-list:acls":{
      "acl":[
         {
            "name":"dns-fragments",
            "type":"ipv4-acl-type",
            "aces":{
               "ace":[
                  {
                     "name":"drop-all-fragments",
                     "matches":{
                        "ipv4":{
                           "acl-enh:ipv4-fragment":{
                              "operator":"match",
                              "type":"isf"
                           }
                        }
                     },
                     "actions":{
                        "forwarding":"drop"
                     }
                  },
                  {
                     "name":"allow-dns-packets",
                     "matches":{
                        "ipv4":{
                           "destination-ipv4-network":"198.51.100.0/24"
                        },
                        "udp":{
                           "destination-port":{
                              "operator":"eq",
                              "port":53
                           }
                        },
                        "actions":{
                           "forwarding":"accept"
                        }
                     }
                  }
               ]
            }
         }
      ]
   }
}
```

Figure 3: Example Illustrating Candidate Filtering of IPv4 Fragmented Packets (Message Body)

Figure 4 shows an example of the body of a POST request to allow the traffic destined to 2001:db8::/32 and UDP port number 53, but to drop all fragmented packets. The following ACEs are defined (in this order):

* "drop-all-fragments" ACE: discards all fragments (including atomic fragments). That is, IPv6 packets that include a Fragment header (44) are dropped.

* "allow-dns-packets" ACE: accepts DNS packets destined to 2001:db8::/32.

```
{
   "ietf-access-control-list:acls":{
      "acl":[
         {
            "name":"dns-fragments",
            "type":"ipv6-acl-type",
            "aces":{
               "ace":[
                  {
                     "name":"drop-all-fragments",
                     "matches":{
                        "ipv6":{
                           "acl-enh:ipv6-fragment":{
                              "operator":"match",
                              "type":"isf"
                           }
                        }
                     },
                     "actions":{
                        "forwarding":"drop"
                     }
                  },
                  {
                     "name":"allow-dns-packets",
                     "matches":{
                        "ipv6":{
                           "destination-ipv6-network":"2001:db8::/32"
                        },
                        "udp":{
                           "destination-port":{
                              "operator":"eq",
                              "port":53
                           }
                        }
                     },
                     "actions":{
                        "forwarding":"accept"
                     }
                  }
               ]
            }
         }
      ]
   }
}
```

Figure 4: An Example Illustrating Filtering of IPv6 Fragmented Packets (Message Body)

3.6. Payload-based Filtering

Some transport protocols use existing protocols (e.g., TCP or UDP) as substrate. The match criteria for such protocols may rely upon the 'protocol' under '13', TCP/UDP match criteria, part of the TCP/UDP payload, or a combination thereof. A new feature, called "match-on-payload", is defined in the document.

3.7. Match MPLS Headers

The ACL model can be used to create rules to match MPLS fields on a packet. The MPLS headers defined in [RFC3032] and [RFC5462] contains the following fields:

* Traffic Class: 3 bits 'EXP' renamed to 'Traffic Class Field."

* Label Value: A 20-bit field that carries the actual value of the MPLS Label.

* TTL: An eight-bit field that is used to encode a time-to-live value.

The structure of the MPLS ACL subtree is shown in Figure 5:

```
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    ...
    +--rw (mpls)?
       +--:(mpls-values)
          +--rw mpls-values {match-on-mpls}?
             +--rw traffic-class?       uint8
             +--rw label-position       identityref
             +--rw upper-label-range?   uint32
             +--rw lower-label-range?   uint32
             +--rw label-block-name     string
             +--rw ttl-value?           uint8
```

Figure 5: MPLS Header Match Subtree

3.8. VLAN Filter

Being able to filter all packets that are bridged within a VLAN or that are routed into or out of a bridge domain is part of the VPN control requirements derived of the EVPN definition done in [RFC7209]. So, all packets that are bridged within a VLAN or that are routed into or out of a VLAN can be captured, forwarded, translated or discarded based on the network policy applied.

Figure 6 shows an ACL example to illustrate how to apply a VLAN range filter.

```
{
   "ietf-acces-control-list:acls":{
      "acl":[
         {
            "name":"VLAN_FILTER",
            "aces":{
               "ace":[
                  {
                     "name":"1",
                     "matches":{
                        "ietf-acl-enh:vlan-filter":{
                           "lower-vlan":10,
                           "upper-vlan":20
                        }
                     },
                     "actions":{
                        "forwarding":"ietf-acces-control-list:accept"
                     }
                  }
               ]
            }
         }
      ]
   }
}
```

Figure 6: Example of VLAN Filter (Message Body)

3.9. ISID Filter

Provider backbone bridging (PBB) was originally defined as Virtual Bridged Local Area Networks [IEEE802.1ah] standard. However, instead of multiplexing VLANs, PBB duplicates the MAC layer of the customer frame and separates it from the provider domain, by encapsulating it in a 24 bit instance service identifier (I-SID). This provides for more transparency between the customer network and the provider network.

The I-component forms the customer or access facing interface or routing instance. The I-component is responsible for mapping customer Ethernet traffic to the appropriate I-SID. In the network is mandatory to configure the default service identifier.

Being able to filter by I-component Service identifier is a feature of the EVNP-PBB configuration.

Figure 7 shows an ACL example to illustrate the ISID range filtering.

```
{
   "ietf-acces-control-list:acls":{
      "acl":[
         {
            "name":"test",
            "aces":{
               "ace":[
                  {
                     "name":"1",
                     "matches":{
                        "ietf-acl-enh:isid-filter":{
                           "lower-isid":100,
                           "upper-isid":200
                        }
                     },
                     "actions":{
                        "forwarding":"ietf-acces-control-list:accept"
                     }
                  }
               ]
            }
         }
      ]
   }
}
```

Figure 7: Example ISID Filter (Message Body)

3.10. Additional Actions

In order to support rate-limiting (see Appendix D.6), a new action called "rate-limit" is defined. Figure 8 shows an ACL example to rate-limit incoming SYNs during a SYN flood attack.

```
{
   "ietf-access-control-list:acls":{
      "acl":[
         {
            "name":"tcp-flags-example-with-rate-limit",
            "aces":{
               "ace":[
                  {
                     "name":"rate-limit-syn",
                     "matches":{
                        "tcp":{
                           "acl-enh:flags-bitmask":{
                              "operator":"match",
                              "bitmask":2
                           }
                        }
                     },
                     "actions":{
                        "forwarding":"accept",
                        "acl-enh:rate-limit":"20.00"
                     }
                  }
               ]
            }
         }
      ]
   }
}
```

Figure 8: An Example of Rate-Limit Incoming TCP SYNs (Message Body).

Also, the model supports new actions to complement existing ones: Log ('log-action') and write a counter ('counter-action'). The current version of the module supports only local actions.

4. Enhanced ACL YANG Module

This model imports types from [RFC6991], [RFC8519], and [RFC8294].

```
<CODE BEGINS> file "ietf-acl-enh@2022-10-24.yang"
module ietf-acl-enh {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acl-enh";
  prefix acl-enh;

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

```
        "RFC 6991: Common YANG Data Types";
    }
    import ietf-access-control-list {
      prefix acl;
      reference
        "RFC 8519: YANG Data Model for Network Access
                   Control Lists (ACLs), Section 4.1";
    }
    import ietf-packet-fields {
      prefix packet-fields;
      reference
        "RFC 8519: YANG Data Model for Network Access
                   Control Lists (ACLs), Section 4.2";
    }
    import ietf-routing-types {
      prefix rt-types;
      reference
        "RFC 8294: Common YANG Data Types for the Routing Area";
    }
    import iana-icmpv4-types {
      prefix iana-icmpv4-types;
      reference
        "TBC";
    }
    import iana-icmpv6-types {
      prefix iana-icmpv6-types;
      reference
        "TBC";
    }
    import iana-ipv6-ext-types {
      prefix iana-ipv6-ext-types;
      reference
        "TBC";
    }

    organization
      "IETF NETMOD Working Group";
    contact
      "WG Web:   https://datatracker.ietf.org/wg/netmod/
       WG List:  mailto:netmod@ietf.org

       Author:   Mohamed Boucadair
                 mailto:mohamed.boucadair@orange.com
       Author:   Samier Barguil
                 mailto:samier.barguilgiraldo.ext@telefonica.com
       Author:   Oscar Gonzalez de Dios
                 mailto:oscar.gonzalezdedios@telefonica.com";
    description
```

```
       "This module contains YANG definitions for enhanced ACLs.

        Copyright (c) 2023 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 Revised BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

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

     revision 2022-10-24 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: Extensions to the Access Control Lists (ACLs)
                    YANG Model";
     }

     feature match-on-payload {
       description
         "Match based on a pattern is supported.";
     }

     feature match-on-vlan-filter {
       description
         "Match based on a VLAN range of vlan list is supported.";
     }

     feature match-on-isid-filter {
       description
         "Match based on a ISID range of vlan list is supported.";
     }

     feature match-on-alias {
       description
         "Match based on aliases.";
     }

     feature match-on-mpls {
       description
         "Match based on MPLS headers.";
     }
```

```
  identity offset-type {
    description
      "Base identity for payload offset type.";
  }

  identity layer2 {
    base offset-type;
    description
      "The offset starts at the beginning of the Data Link layer
       header.";
  }

  identity layer3 {
    base offset-type;
    description
      "The offset starts at the beginning of the IP header.";
  }

  identity layer4 {
    base offset-type;
    description
      "The offset start right after the IP header. This can be
       typically the beginning of transport header (e.g., TCP
       or UDP).";
  }

  identity payload {
    base offset-type;
    description
      "The offset start right after the end of the transport
       payload. For example, this represents the beginning of the
       TCP data right after any TCP options or the beginning of
       the UDP payload right after the UDP header.";
  }

  identity tcp-flag {
    description
      "Base Identity for the TCP Flags.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity ack {
    base tcp-flag;
    description
      "Acknowledgment TCP flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
```

```
  }

  identity syn {
    base tcp-flag;
    description
      "Synchronize sequence numbers.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity fin {
    base tcp-flag;
    description
      "No more data from the sender.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity urg {
    base tcp-flag;
    description
      "Urgent pointer TCP flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity psh {
    base tcp-flag;
    description
      "The Push function flag is similar to the URG flag and tells
       the receiver to process these packets as they are received
       instead of buffering them.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity rst {
    base tcp-flag;
    description
      "Reset TCP flag bit.";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity ece {
    base tcp-flag;
    description
      "ECN-Echo TCP flag bit.";
```

```
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity cwr {
    base tcp-flag;
    description
      "Congestion Window Reduced flag bit";
    reference
      "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
  }

  identity mpls-acl-type {
    base acl:acl-base;
    description
      "An ACL that matches on fields from the MPLS header.";
  }

  identity label-position {
    description
      "Base identity for deriving MPLS label position.";
  }

  identity top {
    base label-position;
    description
      "Top of the label stack.";
  }

  identity bottom {
    base label-position;
    description
      "Bottom of the label stack.";
  }

  identity log-types {
    description
      "Base identity for deriving the Log actions.";
  }

  identity local-log {
    base log-types;
    description
      "A local log is used to record the ACL results.";
  }

  identity counter-type {
    description
```

```
        "Base identity for deriving the Counter actions.";
    }

    identity counter-name {
      base counter-type;
      description
        "Identity for counter name to be updated based on
          the ACL match actions.";

    }

    typedef operator {
      type bits {
        bit not {
          position 0;
          description
            "If set, logical negation of operation.";
        }
        bit match {
          position 1;
          description
            "Match bit. This is a bitwise match operation defined as
             '(data & value) == value'.";
        }
        bit any {
          position 2;
          description
            "Any bit. This is a match on any of the bits in  bitmask.
             It evaluates to 'true' if any of the bits in the value mask
             are set in the data,  i.e., '(data & value) != 0'.";
        }
      }
      description
        "Specifies how to apply the defined bitmask.
         'any' and 'match' bits must not be set simultaneously.";
    }

    typedef fragment-type {
      type bits {
        bit df {
          position 0;
          description
            "Don't fragment bit for IPv4.
             Must be set to 0 when it appears in an IPv6 filter.";
        }
        bit isf {
          position 1;
          description
```

```
            "Is a fragment.";
        }
        bit ff {
          position 2;
          description
            "First fragment.";
        }
        bit lf {
          position 3;
          description
            "Last fragment.";
        }
      }
      description
        "Different fragment types to match against.";
    }

    typedef ipv4-prefix-set-ref {
      type leafref {
        path "/acl-enh:defined-sets/acl-enh:ipv4-prefix-sets"
           + "/acl-enh:prefix-set/acl-enh:name";
      }
      description
        "Defines a reference to an IPv4 prefix set.";
    }

    typedef ipv6-prefix-set-ref {
      type leafref {
        path "/acl-enh:defined-sets/acl-enh:ipv6-prefix-sets"
           + "/acl-enh:prefix-set/acl-enh:name";
      }
      description
        "Defines a reference to an IPv6 prefix set.";
    }

    typedef port-set-ref {
      type leafref {
        path "/acl-enh:defined-sets/acl-enh:port-sets"
           + "/acl-enh:port-set/acl-enh:name";
      }
      description
        "Defines a reference to a port set.";
    }

    typedef protocol-set-ref {
      type leafref {
        path "/acl-enh:defined-sets/acl-enh:protocol-sets"
           + "/acl-enh:protocol-set/acl-enh:name";
```

```
      }
      description
        "Defines a reference to a protocol set.";
    }

    typedef icmpv4-type-set-ref {
      type leafref {
        path "/acl-enh:defined-sets/acl-enh:icmpv4-type-sets"
           + "/acl-enh:icmpv4-type-set/acl-enh:name";
      }
      description
        "Defines a reference to an ICMPv4 type set.";
    }

    typedef icmpv6-type-set-ref {
      type leafref {
        path "/acl-enh:defined-sets/acl-enh:icmpv6-type-sets"
           + "/acl-enh:icmpv6-type-set/acl-enh:name";
      }
      description
        "Defines a reference to an ICMPv6 type set.";
    }

    typedef alias-ref {
      type leafref {
        path "/acl-enh:aliases/acl-enh:alias/acl-enh:name";
      }
      description
        "Defines a reference to an alias.";
    }

    grouping tcp-flags {
      description
        "Operations on TCP flags.";
      choice mode {
        description
          "Choice of how flags are indicated.";
        case explicit {
          leaf operator {
            type operator;
            default "match";
            description
              "How to interpret the TCP flags.";
          }
          leaf-list explicit-tcp-flag {
            type identityref {
              base tcp-flag;
            }
```

```
            description
              "An explicit list of the TCP flags that are to be
               matched.";
          }
        }
        case builtin {
          leaf bitmask {
            type uint16;
            description
              "The bitmask matches the last 4 bits of byte 12 and 13 of
               the TCP header.  For clarity, the 4 bits of byte 12
               corresponding to the TCP data offset field are not
               included in any matching.";
            reference
             "RFC 9293: Transmission Control Protocol (TCP),
                        Section 3.1";
          }
        }
      }
    }

    grouping fragment-fields {
      description
        "Operations on fragment types.";
      leaf operator {
        type operator;
        default "match";
        description
          "How to interpret the fragment type.";
      }
      leaf type {
        type fragment-type;
        description
          "What fragment type to look for.";
      }
    }

  grouping mpls-match-parameters-config {
    description
      "Parameters for the configuration of MPLS match rules.";

    leaf traffic-class {
      type uint8 {
        range "0..7";
      }
      description
        "The value of the MPLS traffic class (TC) bits,
         formerly known as the EXP bits.";
```

```
    }

    leaf label-position {
      type identityref {
        base label-position;
      }
      description
        "Position of the label";
    }

    leaf upper-label-range {
      type rt-types:mpls-label;
      description
        "Match MPLS label value on the MPLS header.
         The usage of this field indicated the upper
         range value in the top of the stack.
         This label value does not include the
         encodings of Traffic Class and TTL.";
       reference
         "RFC 3032: MPLS Label Stack Encoding";
    }

    leaf lower-label-range {
      type rt-types:mpls-label;
      description
        "Match MPLS label value on the MPLS header.
         The usage of this field indicated the lower
         range value in the top of the stack.
         This label value does not include the
         encodings of Traffic Class and TTL.";
       reference
         "RFC 3032: MPLS Label Stack Encoding";
    }

    leaf label-block-name {
      type string;
      description
      "Reference to a label block predefiend in the
       implementation.";
    }

    leaf ttl-value {
      type uint8;
      description
        "Time-to-live MPLS packet value match.";
      reference
        "RFC 3032: MPLS Label Stack Encoding";
      }
```

```
    }

    grouping payload {
      description
        "Operations on payload match.";
      leaf offset {
        type identityref {
          base offset-type;
        }
        description
          "Indicates the payload offset. This will indicate the position
           of the data in packet to use for the match.";
      }
      leaf offset-end {
        type uint64;
        units "bytes";
        description
          "Indicates the number of bytes, starting from the offset to
           cover when performing the prefix match.";
      }
      leaf operator {
        type operator;
        default "match";
        description
          "How to interpret the prefix match.";
      }
      leaf prefix {
        type binary;
        description
          "The binary pattern to match against.";
      }
    }

    grouping alias {
      description
        "Specifies an alias.";
      leaf-list vlan {
        type uint16;
          description
            "VLAN of the alias.";
      }
      leaf-list prefix {
        type inet:ip-prefix;
        description
          "IPv4 or IPv6 prefix of the alias.";
      }
      list port-range {
        key "lower-port";
```

```
        description
          "Port range.  When only lower-port is
           present, it represents a single port number.";
        leaf lower-port {
          type inet:port-number;
          mandatory true;
          description
            "Lower port number of the port range.";
        }
        leaf upper-port {
          type inet:port-number;
          must '. >= ../lower-port' {
            error-message
              "The upper-port number must be greater than
               or equal to the lower-port number.";
          }
          description
            "Upper port number of the port range.";
        }
      }
      leaf-list protocol {
        type uint8;
        description
          "Identifies the target protocol number.
           For example, 6 for TCP or 17 for UDP.";
      }
      leaf-list fqdn {
        type inet:domain-name;
        description
          "FQDN identifying the target.";
      }
      leaf-list uri {
        type inet:uri;
        description
          "URI identifying the target.";
      }
    }

    grouping icmpv4-header-fields {
      description
        "Collection of ICMPv4 header fields that can be
         used to set up a match filter.";
      leaf type {
        type iana-icmpv4-types:icmpv4-type;
        description
          "Also known as control messages.";
        reference
          "RFC 792: Internet Control Message Protocol.";
```

```
      }
      leaf code {
        type uint8;
        description
          "ICMP subtype.";
        reference
          "RFC 792: Internet Control Message Protocol.";
      }
      leaf rest-of-header {
        type binary;
        description
          "Unbounded in length, the contents vary based on the
           ICMP type and code.";
        reference
          "RFC 792: Internet Control Message Protocol";
      }
    }

    grouping icmpv6-header-fields {
      description
        "Collection of ICMPv6 header fields that can be
         used to set up a match filter.";
      leaf type {
        type iana-icmpv6-types:icmpv6-type;
        description
          "Also known as control messages.";
        reference
          "RFC 4443: Internet Control Message Protocol (ICMPv6)
                     for Internet Protocol Version 6 (IPv6)
                     Specification.";
      }
      leaf code {
        type uint8;
        description
          "ICMP code.";
        reference
          "RFC 4443: Internet Control Message Protocol (ICMPv6)
                     for Internet Protocol Version 6 (IPv6)
                     Specification.";
      }
      leaf rest-of-header {
        type binary;
        description
          "Unbounded in length, the contents vary based on the
           ICMP type and code. Also referred to as 'Message Body'
           in ICMPv6.";
        reference
          "RFC 4443: Internet Control Message Protocol (ICMPv6)
```

```
                    for Internet Protocol Version 6 (IPv6)
                    Specification.";
      }
    }

    grouping acl-complementary-actions {
      description
        "Collection of complementary ACL actions.";


      container log-action {
        description
          "Container for defining log actions.";

        leaf log-type {
          type identityref {
            base acl-enh:log-types;
          }
          description
            "The type of log action to be performed.";
        }
        leaf log-id {
          when "../log-type = 'local-log'" {
            description
              "Name of the log file updated when type is 'local-log'.";
          }
          type string;
          description
            "The name of the counter action.";
        }
      }

      container counter-action {
        description
          "Container for defining counter actions.";

        leaf counter-type {
          type identityref {
            base acl-enh:counter-type;
          }
          description
            "The type of counter action to be performed.";
        }
        leaf-list counter-name {
          when "../counter-type = 'counter-name'" {
            description
              "Name for the counter or variable to update when counter-type
              is 'counter-name'.";
```

```
          }
          type string;
          description
            "List of possible variables or counter names to
            update based on match critieria.";
        }
      }
    }

    augment "/acl:acls/acl:acl/acl:aces/acl:ace"
          + "/acl:matches" {
      description
        "Adds a match type based on the payload.";
      choice payload {
        description
          "Matches based upon a prefix pattern.";
        container prefix-pattern {
          if-feature "match-on-payload";
          description
            "Indicates the rule to perform the payload-based match.";
          uses payload;
        }
      }
      choice alias {
        description
          "Matches based upon aliases.";
        leaf-list alias-name {
          type alias-ref;
          description
            "Indicates one or more aliases.";
        }
      }
      choice mpls {
        container mpls-values {
          if-feature "match-on-mpls";
          uses mpls-match-parameters-config;
          description
            "Provides the rule set that matches MPLS headers.";
        }
        description
          "Matches against MPLS headers, for example, label values";
      }
    }

    augment "/acl:acls/acl:acl/acl:aces"
          + "/acl:ace/acl:matches/acl:l2" {
      description
        "Adds a match type based on MAC VLAN and ISID filters.";
```

```
      container vlan-filter {
        if-feature "match-on-vlan-filter";
        description
          "Indicates how to handle MAC VLANs.";
        leaf frame-type {
          type string;
          description
            "Entering the frame type allows the
             filter to match a specific type of frame format";
        }
        choice vlan-type {
          description
            "VLAN definition from range or operator.";
          case range {
            leaf lower-vlan {
              type uint16;
              must '. <= ../upper-vlan' {
                error-message
                  "The lower-vlan must be less than or equal to
                   the upper-vlan.";
              }
              mandatory true;
              description
                "Lower boundary for a vlan.";
            }
            leaf upper-vlan {
              type uint16;
              mandatory true;
              description
                "Upper boundary for a vlan.";
            }
          }
          case operator {
            leaf operator {
              type packet-fields:operator;
              default "eq";
              description
                "Operator to be applied on the vlan below.";
            }
            leaf-list vlan {
              type uint16;
              description
                "VLAN number along with the operator on which to
                 match.";
            }
          }
        }
      }
```

```
      container isid-filter {
        if-feature "match-on-isid-filter";
        description
          "Indicates how to handle ISID filters.
           The I-component is responsible for mapping customer
           Ethernet traffic to the appropriate ISID.";
        choice isid-type {
          description
            "ISID definition from range or operator.";
          case range {
            leaf lower-isid {
              type uint16;
              must '. <= ../upper-isid' {
                error-message
                  "The lower-isid must be less than or equal to
                   the upper-isid.";
              }
              mandatory true;
              description
                "Lower boundary for a ISID.";
            }
            leaf upper-isid {
              type uint16;
              mandatory true;
              description
                "Upper boundary for a ISID.";
            }
          }
          case operator {
            leaf operator {
              type packet-fields:operator;
              default "eq";
              description
                "Operator to be applied on the ISID below.";
            }
            leaf-list isid {
              type uint16;
              description
                "ISID number along with the operator on which to
                 match.";
            }
          }
        }
      }

      augment "/acl:acls/acl:acl/acl:aces"
            + "/acl:ace/acl:matches/acl:l3/acl:ipv4/acl:ipv4" {
```

```
      description
        "Handle non-initial and initial fragments for IPv4 packets.";
      container ipv4-fragment {
        description
          "Indicates how to handle IPv4 fragments.";
        uses fragment-fields;
      }
      leaf source-ipv4-prefix-list {
        type ipv4-prefix-set-ref;
        description
          "A reference to an IPv4 prefix list to match the source
           address.";
      }
      leaf destination-ipv4-prefix-list {
        type ipv4-prefix-set-ref;
        description
          "A reference to a prefix list to match the destination
           address.";
      }
      leaf next-header-set {
        type protocol-set-ref;
        description
          "A reference to a protocol set to match the next-header
           field.";
      }
    }

    augment "/acl:acls/acl:acl/acl:aces"
          + "/acl:ace/acl:matches/acl:l3/acl:ipv6/acl:ipv6" {
      description
        "Handles non-initial and initial fragments for IPv6 packets.";
      container ipv6-fragment {
        description
          "Indicates how to handle IPv6 fragments.";
        uses fragment-fields;
      }
      leaf source-ipv6-prefix-list {
        type ipv6-prefix-set-ref;
        description
          "A reference to a prefix list to match the source address.";
      }
      leaf destination-ipv6-prefix-list {
        type ipv6-prefix-set-ref;
        description
          "A reference to a prefix list to match the destination
           address.";
      }
      leaf protocol-set {
```

```
        type protocol-set-ref;
        description
          "A reference to a protocol set to match the protocol field.";
      }
      leaf extension-header {
         type iana-ipv6-ext-types:ipv6-extension-header-type;
         description
           "IPv6 extension header value.";
      }
    }

    augment "/acl:acls/acl:acl/acl:aces"
          + "/acl:ace/acl:matches/acl:l4/acl:tcp" {
      description
        "Handles TCP flags and port sets.";
      container flags-bitmask {
        description
          "Indicates how to handle TCP flags.";
        uses tcp-flags;
      }
      leaf source-tcp-port-set {
        type port-set-ref;
        description
          "A reference to a port set to match the source port.";
      }
      leaf destination-tcp-port-set {
        type port-set-ref;
        description
          "A reference to a port set to match the destination port.";
      }
    }

    augment "/acl:acls/acl:acl/acl:aces"
          + "/acl:ace/acl:matches/acl:l4/acl:udp" {
      description
        "Handle UDP port sets.";
      leaf source-udp-port-set {
        type port-set-ref;
        description
          "A reference to a port set to match the source port.";
      }
      leaf destination-udp-port-set {
        type port-set-ref;
        description
          "A reference to a port set to match the destination port.";
      }
    }
```

```
  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:matches/acl:l4/acl:icmp" {
    description
      "Handle ICMP type sets.";
    leaf icmpv4-set {
      type icmpv4-type-set-ref;
      description
        "A reference to an ICMPv4 type set to match the ICMPv4 type
         field.";
    }
    leaf icmpv6-set {
      type icmpv6-type-set-ref;
      description
        "A reference to an ICMPv6 type set to match the ICMPv6 type
         field.";
    }
  }

  augment "/acl:acls/acl:acl/acl:aces"
        + "/acl:ace/acl:actions" {
    description
      "Complementary actions including Rate-limit action.";

    uses acl-complementary-actions;

    leaf rate-limit {
      when "../acl:forwarding = 'acl:accept'" {
        description
          "Rate-limit valid only when accept action is used.";
      }
      type decimal64 {
        fraction-digits 2;
      }
      units "bytes per second";
      description
        "Indicates a rate-limit for the matched traffic.";
    }
  }


  container defined-sets {
    description
      "Predefined sets of attributes used in policy match
       statements.";
    container ipv4-prefix-sets {
      description
        "Data definitions for a list of IPv4 or IPv6
         prefixes which are matched as part of a policy.";
```

```
        list prefix-set {
          key "name";
          description
            "List of the defined prefix sets.";
          leaf name {
            type string;
            description
              "Name of the prefix set -- this is used as a label to
               reference the set in match conditions.";
          }
          leaf description {
            type string;
            description
              "Defined Set description.";
          }
          leaf-list prefix {
            type inet:ipv4-prefix;
            description
              "List of IPv4 prefixes to be used in match
               conditions.";
          }
        }
      }
      container ipv6-prefix-sets {
        description
          "Data definitions for a list of IPv6 prefixes which are
           matched as part of a policy.";
        list prefix-set {
          key "name";
          description
            "List of the defined prefix sets.";
          leaf name {
            type string;
            description
              "Name of the prefix set -- this is used as a label to
               reference the set in match conditions.";
          }
          leaf description {
            type string;
            description
              "A textual description of the prefix list.";
          }
          leaf-list prefix {
            type inet:ipv6-prefix;
            description
              "List of IPv6 prefixes to be used in match conditions.";
          }
        }
```

```
      }
      container port-sets {
        description
          "Data definitions for a list of ports which can
           be matched in policies.";
        list port-set {
          key "name";
          description
            "List of port set definitions.";
          leaf name {
            type string;
            description
              "Name of the port set -- this is used as a label to
               reference the set in match conditions.";
          }
          list port {
            key "id";
            description
              "Port numbers along with the operator on which to
               match.";
            leaf id {
              type string;
              description
                "Identifier of the list of port numbers.";
            }
            choice port {
              description
                "Choice of specifying the port number or referring to a
                 group of port numbers.";
              container port-range-or-operator {
                description
                  "Indicates a set of ports.";
                uses packet-fields:port-range-or-operator;
              }
            }
          }
        }
      }
      container protocol-sets {
        description
          "Data definitions for a list of protocols which can be matched
           in policies.";
        list protocol-set {
          key "name";
          description
            "List of protocol set definitions.";
          leaf name {
            type string;
```

```
            description
              "Name of the protocols set -- this is used as a label to
               reference the set in match conditions.";
          }
          leaf-list protocol {
            type union {
              type uint8;
              type string;
            }
            description
              "Value of the protocol set.";
            //Check if we can reuse an IANA-maintained module
          }
        }
      }
      container icmpv4-type-sets {
        description
          "Data definitions for a list of ICMPv4 types which can be
           matched in policies.";
        list icmpv4-type-set {
          key "name";
          description
            "List of ICMP type set definitions.";
          leaf name {
            type string;
            description
              "Name of the ICMPv4 type set -- this is used as a label to
               reference the set in match conditions.";
          }
          list types {
            key "type";
            description
              "Includes a list of ICMPv4 types.";
            uses icmpv4-header-fields;
          }
        }
      }
    container icmpv6-type-sets {
        description
          "Data definitions for a list of ICMPv6 types which can be
           matched in policies.";
        list icmpv6-type-set {
          key "name";
          description
            "List of ICMP type set definitions.";
          leaf name {
            type string;
            description
```

```
                "Name of the ICMPv6 type set -- this is used as a label to
                 reference the set in match conditions.";
            }
            list types {
              key "type";
              description
                "Includes a list of ICMPv6 types.";
              uses icmpv6-header-fields;
            }
          }
        }
    }
      container aliases {
        description
          "Top-levl container for aliases.";
        list alias {
          key "name";
          description
            "List of aliases.";
          leaf name {
            type string;
            description
              "The name of the alias.";
          }
          uses alias;
        }
      }
    }
    <CODE ENDS>
```

5. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocol 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 [RFC8446].

The Network Configuration Access Control Model (NACM) [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.

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 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:

* TBC

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:

* TBC

6. IANA Considerations

6.1. URI Registrations

This document requests IANA to register the following URIs in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

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

URI: urn:ietf:params:xml:ns:yang:iana-icmpv4-types
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-icmpv6-types
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
```

6.2. YANG Module Name Registrations

This document requests IANA to register the following YANG modules in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

```
      name: ietf-acl-enh
      namespace: urn:ietf:params:xml:ns:yang:ietf-acl-enh
      maintained by IANA: N
      prefix: acl-enh
      reference: RFC XXXX

      name: iana-icmpv4-types
      namespace: urn:ietf:params:xml:ns:yang:iana-icmpv4-types
      maintained by IANA: Y
      prefix: iana-icmpv4-types
      reference: RFC XXXX

      name: iana-icmpv6-types
      namespace: urn:ietf:params:xml:ns:yang:iana-icmpv6-types
      maintained by IANA: Y
      prefix: iana-icmpv6-types
      reference: RFC XXXX

      name: iana-ipv6-ext-types
      namespace: urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types
      maintained by IANA: Y
      prefix: iana-ipv6-ext-types
      reference: RFC XXXX
```

6.3. Considerations for IANA-Maintained Modules

6.3.1. ICMPv4 Types IANA Module

IANA is requested to create and post the initial version of the "iana-icmpv4-types" YANG module by applying the XSLT stylesheet from Appendix A.1 to the XML version of [IANA-ICMPv4].

This document defines the initial version of the IANA-maintained "iana-icmpv4-types" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add this note to the registry [IANA-YANG-PARAMETERS]:

> New values must not be directly added to the "iana-icmpv4-types" YANG module. They must instead be added to the "ICMP Type Numbers" registry [IANA-ICMPv4].

When a value is added to the "ICMP Type Numbers" registry, a new "enum" statement must be added to the "iana-icmpv4-types" YANG module. The "enum" statement, and sub-statements thereof, should be defined:

"enum": Replicates a name from the registry.

"value": Contains the decimal value of the IANA-assigned value.

"status": Is included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

"description": Replicates the description from the registry.

"reference": Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-icmpv4-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to "ICMP Type Numbers" [IANA-ICMPv4]:

When this registry is modified, the YANG module "iana-icmpv4-types" [IANA_ICMPv4_YANG_URL] must be updated as defined in RFCXXXX.

IANA is requested to updated the "Reference" in the "ICMP Type Numbers" registry as follows:

OLD: [RFC2780]

NEW: [RFC2780][This_Document]

6.3.2. ICMPv6 Types IANA Module

IANA is requested to create and post the initial version of the "iana-icmpv6-types" YANG module by applying the XSLT stylesheet from Appendix B.1 to the XML version of [IANA-ICMPv4].

This document defines the initial version of the IANA-maintained "iana-icmpv6-types" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add this note to the registry [IANA-YANG-PARAMETERS]:

New values must not be directly added to the "iana-icmpv6-types" YANG module. They must instead be added to the "ICMPv6 "type" Numbers" registry [IANA-ICMPv6].

When a value is added to the "ICMPv6 "type" Numbers" registry, a new "enum" statement must be added to the "iana-icmpv6-types" YANG module. The "enum" statement, and sub-statements thereof, should be defined:

"enum": Replicates a name from the registry.

"value": Contains the decimal value of the IANA-assigned value.

"status": Is included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

"description": Replicates the description from the registry.

"reference": Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-icmpv6-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to "ICMPv6 "type" Numbers" [IANA-ICMPv6]:

When this registry is modified, the YANG module "iana-icmpv6-types" [IANA_ICMPv6_YANG_URL] must be updated as defined in RFCXXXX.

IANA is requested to updated the "Reference" in the "ICMPv6 "type" Numbers" registry as follows:

OLD: [RFC4443]

NEW: [RFC4443][This_Document]

6.3.3. IPv6 Extension Header Types IANA Module

IANA is requested to create and post the initial version of the "iana-ipv6-ext-types" YANG module by applying the XSLT stylesheet from Appendix C.1 to the XML version of [IANA-IPv6].

This document defines the initial version of the IANA-maintained "iana-ipv6-ext-types" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add this note to the registry [IANA-YANG-PARAMETERS]:

> New values must not be directly added to the "iana-ipv6-ext-types" YANG module. They must instead be added to the "IPv6 Extension Header Types" registry [IANA-ICMPv6].

When a value is added to the "IPv6 Extension Header Types" registry, a new "enum" statement must be added to the "iana-ipv6-ext-types" YANG module. The "enum" statement, and sub-statements thereof, should be defined:

"enum": Replicates a name from the registry.

"value": Contains the decimal value of the IANA-assigned value.

"status": Is included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

"description": Replicates the description from the registry.

"reference": Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-ipv6-ext-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to the "IPv6 Extension Header Types" registry [IANA-IPv6]:

When this registry is modified, the YANG module "iana-ipv6-ext-types" [IANA_IPV6_YANG_URL] must be updated as defined in RFCXXXX.

IANA is requested to updated the "Reference" in the "IPv6 Extension Header Types" registry as follows:

OLD: [RFC2780][RFC5237][RFC7045]

NEW: [RFC2780][RFC5237][RFC7045][This_Document]

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, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/rfc/rfc3032>.

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

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <https://www.rfc-editor.org/rfc/rfc5462>.

[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/rfc/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/rfc/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/rfc/rfc6242>.

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

[RFC7209] Sajassi, A., Aggarwal, R., Uttaro, J., Bitar, N., Henderickx, W., and A. Isaac, "Requirements for Ethernet VPN (EVPN)", RFC 7209, DOI 10.17487/RFC7209, May 2014, <https://www.rfc-editor.org/rfc/rfc7209>.

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

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/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/rfc/rfc8174>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/rfc/rfc8294>.

[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/rfc/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/rfc/rfc8519>.

[RFC8956] Loibl, C., Ed., Raszuk, R., Ed., and S. Hares, Ed., "Dissemination of Flow Specification Rules for IPv6", RFC 8956, DOI 10.17487/RFC8956, December 2020, <https://www.rfc-editor.org/rfc/rfc8956>.

7.2. Informative References

[I-D.boucadair-netmod-rfc8407bis]
Boucadair, M. and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-boucadair-netmod-rfc8407bis-02, 26 July 2023, <https://datatracker.ietf.org/doc/html/draft-boucadair-netmod-rfc8407bis-02>.

[IANA-ICMPv4]
"ICMP Type Numbers", n.d.,
<https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml>.

[IANA-ICMPv6]
"ICMPv6 type Numbers", n.d.,
<https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml>.

[IANA-IPv6]
"IPv6 Extension Header Types", n.d.,
<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>.

[IANA-YANG-PARAMETERS]
"YANG Parameters", n.d.,
<https://www.iana.org/assignments/yang-parameters>.

[RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, DOI 10.17487/RFC2780, March 2000, <https://www.rfc-editor.org/rfc/rfc2780>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/rfc/rfc4443>.

[RFC5237] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the Protocol Field", BCP 37, RFC 5237, DOI 10.17487/RFC5237, February 2008, <https://www.rfc-editor.org/rfc/rfc5237>.

[RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <https://www.rfc-editor.org/rfc/rfc7045>.

[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/rfc/rfc8340>.

[RFC8955] Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", RFC 8955, DOI 10.17487/RFC8955, December 2020, <https://www.rfc-editor.org/rfc/rfc8955>.

[RFC9132] Boucadair, M., Ed., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 9132, DOI 10.17487/RFC9132, September 2021, <https://www.rfc-editor.org/rfc/rfc9132>.

Appendix A. ICMPv4 Types

A.1. XSLT Template to Generate The ICMPv4 Types IANA-Maintained Module

```
<CODE BEGINS>
<?xml version="1.0" encoding="utf-8"?>
<stylesheet
    xmlns="http://www.w3.org/1999/XSL/Transform"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xmlns:iana="http://www.iana.org/assignments"
    xmlns:yin="urn:ietf:params:xml:ns:yang:yin:1"
    version="1.0">
  <import href="../../../xslt/iana-yinx.xsl"/>
  <output method="xml" encoding="utf-8"/>
  <strip-space elements="*"/>

  <template match="iana:registry[@id='icmp-parameters-types']">
    <element name="yin:typedef">
      <attribute name="name">icmpv4-type-name</attribute>
      <element name="yin:type">
        <attribute name="name">enumeration</attribute>
        <apply-templates
            select="iana:record[not(iana:description = 'Unassigned' or
                    starts-with(iana:description, 'Reserved') or
                    starts-with(iana:description, 'RFC3692')) or
                    contains(iana:description, 'experimental')]"/>
      </element>
      <element name="yin:description">
        <element name="yin:text">
          This enumeration type defines mnemonic names and
          corresponding numeric values of ICMPv4 types.
        </element>
      </element>
      <element name="yin:reference">
        <element name="yin:text">
          RFC 2708: IANA Allocation Guidelines For Values In
                    the Internet Protocol and Related Headers
        </element>
      </element>
    </element>
    <element name="yin:typedef">
      <attribute name="name">icmpv4-type</attribute>
```

```
        <element name="yin:type">
          <attribute name="name">union</attribute>
          <element name="yin:type">
            <attribute name="name">uint8</attribute>
          </element>
          <element name="yin:type">
            <attribute name="name">icmpv4-type-name</attribute>
          </element>
        </element>
        <element name="yin:description">
          <element name="yin:text">
            This type allows reference to an ICMPv4 type using either
            the assigned mnemonic name or numeric value.
          </element>
        </element>
      </element>
    </template>

    <template match="iana:record">
      <call-template name="enum">
        <with-param name="id">
          <choose>
            <when test="contains(iana:description, '(Deprecated)')">
              <value-of select="translate(normalize-space(substring-before(iana:
description,
                    '(Deprecated)')),' ','')"/>
            </when>
            <otherwise>
              <value-of select="translate(normalize-space(iana:description),' ',
'')"/>
            </otherwise>
          </choose>
        </with-param>
        <with-param name="deprecated"
                    select="contains(iana:description, '(Deprecated)')"/>
      </call-template>
    </template>

  </stylesheet>
  <CODE ENDS>
```

A.2. Initial Version of the The ICMPv4 Types IANA-Maintained Module

```
  <CODE BEGINS> file "iana-icmpv4-types@2020-09-25.yang"
  module iana-icmpv4-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-icmpv4-types";
    prefix iana-icmpv4-types;

    organization
```

```
       "Internet Assigned Numbers Authority (IANA)";

     contact
       "Internet Assigned Numbers Authority

        ICANN
        12025 Waterfront Drive, Suite 300
        Los Angeles, CA 90094


        Tel: +1 424 254 5300

        <mailto:iana@iana.org>";

     description
       "This YANG module translates IANA registry 'ICMP Type Numbers' to
        YANG derived types.

        Copyright (c) 2020 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 Revised 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).

        This version of this YANG module was generated from the
        corresponding IANA registry using an XSLT stylesheet from the
        'iana-yang' project (https://github.com/llhotka/iana-yang).";

     reference
       "Internet Control Message Protocol (ICMP) Parameters
        (https://www.iana.org/assignments/icmp-parameters/)";

     revision 2020-09-25 {
       description
         "Current revision as of the revision date specified in the XML
          representation of the registry page.";
       reference
         "https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xml";
     }

     /* Typedefs */

     typedef icmpv4-type-name {
       type enumeration {
```

```
        enum EchoReply {
          value 0;
          description
            "Echo Reply";
          reference
            "RFC 792";
        }
        enum DestinationUnreachable {
          value 3;
          description
            "Destination Unreachable";
          reference
            "RFC 792";
        }
        enum SourceQuench {
          value 4;
          status deprecated;
          description
            "Source Quench (Deprecated)";
          reference
            "- RFC 792
             - RFC 6633";
        }
        enum Redirect {
          value 5;
          description
            "Redirect";
          reference
            "RFC 792";
        }
        enum AlternateHostAddress {
          value 6;
          status deprecated;
          description
            "Alternate Host Address (Deprecated)";
          reference
            "RFC 6918";
        }
        enum Echo {
          value 8;
          description
            "Echo";
          reference
            "RFC 792";
        }
        enum RouterAdvertisement {
          value 9;
          description
```

```
            "Router Advertisement";
          reference
            "RFC 1256";
        }
        enum RouterSolicitation {
          value 10;
          description
            "Router Solicitation";
          reference
            "RFC 1256";
        }
        enum TimeExceeded {
          value 11;
          description
            "Time Exceeded";
          reference
            "RFC 792";
        }
        enum ParameterProblem {
          value 12;
          description
            "Parameter Problem";
          reference
            "RFC 792";
        }
        enum Timestamp {
          value 13;
          description
            "Timestamp";
          reference
            "RFC 792";
        }
        enum TimestampReply {
          value 14;
          description
            "Timestamp Reply";
          reference
            "RFC 792";
        }
        enum InformationRequest {
          value 15;
          status deprecated;
          description
            "Information Request (Deprecated)";
          reference
            "- RFC 792
             - RFC 6918";
        }
```

```
        enum InformationReply {
          value 16;
          status deprecated;
          description
            "Information Reply (Deprecated)";
          reference
            "- RFC 792
             - RFC 6918";
        }
        enum AddressMaskRequest {
          value 17;
          status deprecated;
          description
            "Address Mask Request (Deprecated)";
          reference
            "- RFC 950
             - RFC 6918";
        }
        enum AddressMaskReply {
          value 18;
          status deprecated;
          description
            "Address Mask Reply (Deprecated)";
          reference
            "- RFC 950
             - RFC 6918";
        }
        enum Traceroute {
          value 30;
          status deprecated;
          description
            "Traceroute (Deprecated)";
          reference
            "- RFC 1393
             - RFC 6918";
        }
        enum DatagramConversionError {
          value 31;
          status deprecated;
          description
            "Datagram Conversion Error (Deprecated)";
          reference
            "- RFC 1475
             - RFC 6918";
        }
        enum MobileHostRedirect {
          value 32;
          status deprecated;
```

```
          description
            "Mobile Host Redirect (Deprecated)";
          reference
            "- David Johnson <>
             - RFC 6918";
        }
        enum IPv6Where-Are-You {
          value 33;
          status deprecated;
          description
            "IPv6 Where-Are-You (Deprecated)";
          reference
            "- Bill Simpson <mailto:Bill.Simpson&um.cc.umich.edu>
             - RFC 6918";
        }
        enum IPv6I-Am-Here {
          value 34;
          status deprecated;
          description
            "IPv6 I-Am-Here (Deprecated)";
          reference
            "- Bill Simpson <mailto:Bill.Simpson&um.cc.umich.edu>
             - RFC 6918";
        }
        enum MobileRegistrationRequest {
          value 35;
          status deprecated;
          description
            "Mobile Registration Request (Deprecated)";
          reference
            "- Bill Simpson <mailto:Bill.Simpson&um.cc.umich.edu>
             - RFC 6918";
        }
        enum MobileRegistrationReply {
          value 36;
          status deprecated;
          description
            "Mobile Registration Reply (Deprecated)";
          reference
            "- Bill Simpson <mailto:Bill.Simpson&um.cc.umich.edu>
             - RFC 6918";
        }
        enum DomainNameRequest {
          value 37;
          status deprecated;
          description
            "Domain Name Request (Deprecated)";
          reference
```

```
            "- RFC 1788
             - RFC 6918";
          }
          enum DomainNameReply {
            value 38;
            status deprecated;
            description
              "Domain Name Reply (Deprecated)";
            reference
              "- RFC 1788
               - RFC 6918";
          }
          enum SKIP {
            value 39;
            status deprecated;
            description
              "SKIP (Deprecated)";
            reference
              "- Tom Markson <mailto:markson&osmosys.incog.com>
               - RFC 6918";
          }
          enum Photuris {
            value 40;
            description
              "Photuris";
            reference
              "RFC 2521";
          }
          enum ICMPmessagesutilizedbyexperimentalmobilityprotocolssuchasSeamoby {
            value 41;
            description
              "ICMP messages utilized by experimental mobility protocols
               such as Seamoby";
            reference
              "RFC 4065";
          }
          enum ExtendedEchoRequest {
            value 42;
            description
              "Extended Echo Request";
            reference
              "RFC 8335";
          }
          enum ExtendedEchoReply {
            value 43;
            description
              "Extended Echo Reply";
            reference
```

```
              "RFC 8335";
          }
        }
        description
          "This enumeration type defines mnemonic names and corresponding
           numeric values of ICMPv4 types.";
        reference
          "RFC 2708: IANA Allocation Guidelines For Values In the
           Internet Protocol and Related Headers";
      }

      typedef icmpv4-type {
        type union {
          type uint8;
          type icmpv4-type-name;
        }
        description
          "This type allows reference to an ICMPv4 type using either the
           assigned mnemonic name or numeric value.";
      }
    }
    <CODE ENDS>
```

Appendix B. ICMPv6 Types

B.1. XSLT Template to Generate The ICMPv6 Types IANA-Maintained Module

```
    <CODE BEGINS>
    <?xml version="1.0" encoding="utf-8"?>
    <stylesheet
        xmlns="http://www.w3.org/1999/XSL/Transform"
        xmlns:html="http://www.w3.org/1999/xhtml"
        xmlns:iana="http://www.iana.org/assignments"
        xmlns:yin="urn:ietf:params:xml:ns:yang:yin:1"
        version="1.0">
      <import href="../../../xslt/iana-yinx.xsl"/>
      <output method="xml" encoding="utf-8"/>
      <strip-space elements="*"/>

      <template match="iana:registry[@id='icmpv6-parameters-2']">
        <element name="yin:typedef">
          <attribute name="name">icmpv6-type-name</attribute>
          <element name="yin:type">
            <attribute name="name">enumeration</attribute>
            <apply-templates
                select="iana:record[not(iana:name = 'Unassigned' or
                        starts-with(iana:name, 'Reserved') or
                        starts-with(iana:name, 'Private'))]"/>
```

```
          </element>
          <element name="yin:description">
            <element name="yin:text">
              This enumeration type defines mnemonic names and
              corresponding numeric values of ICMPv6 types.
            </element>
          </element>
          <element name="yin:reference">
            <element name="yin:text">
              RFC 2708: IANA Allocation Guidelines For Values In
                        the Internet Protocol and Related Headers
            </element>
          </element>
        </element>
        <element name="yin:typedef">
          <attribute name="name">icmpv6-type</attribute>
          <element name="yin:type">
            <attribute name="name">union</attribute>
            <element name="yin:type">
              <attribute name="name">uint8</attribute>
            </element>
            <element name="yin:type">
              <attribute name="name">icmpv6-type-name</attribute>
            </element>
          </element>
          <element name="yin:description">
            <element name="yin:text">
              This type allows reference to an ICMPv6 type using either
              the assigned mnemonic name or numeric value.
            </element>
          </element>
        </element>
      </template>

      <template match="iana:record">
        <call-template name="enum">
          <with-param name="id">
            <choose>
              <when test="contains(iana:name, '(Deprecated)')">
                <value-of select="translate(normalize-space(substring-before(iana:
name,
                      '(Deprecated)')),' ','')"/>
              </when>
              <otherwise>
                <value-of select="translate(normalize-space(iana:name),' ','')"/>
              </otherwise>
            </choose>
          </with-param>
          <with-param name="description">
```

```
          <value-of select="concat(iana:name, '.')"/>
        </with-param>
        <with-param name="deprecated"
                    select="contains(iana:name, '(Deprecated)')"/>
      </call-template>
    </template>

  </stylesheet>
  <CODE ENDS>
```

B.2. Initial Version of the The ICMPv4 Types IANA-Maintained Module

```
  <CODE BEGINS> file "iana-icmpv6-types@2020-09-25.yang"
  module iana-icmpv6-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-icmpv6-types";
    prefix iana-icmpv6-types;

    organization
      "Internet Assigned Numbers Authority (IANA)";

    contact
      "Internet Assigned Numbers Authority

       ICANN
       12025 Waterfront Drive, Suite 300
       Los Angeles, CA 90094


       Tel: +1 424 254 5300

       <mailto:iana@iana.org>";

    description
      "This YANG module translates IANA registry 'ICMPv6 \"type\"
       Numbers' to YANG derived types.

       Copyright (c) 2023 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 Revised 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).

       This version of this YANG module was generated from the
```

```
        corresponding IANA registry using an XSLT stylesheet from the
        'iana-yang' project (https://github.com/llhotka/iana-yang).";

      reference
        "Internet Control Message Protocol version 6 (ICMPv6) Parameters
         (https://www.iana.org/assignments/icmpv6-parameters/)";

      revision 2023-04-28 {
        description
          "Current revision as of the revision date specified in the XML
           representation of the registry page.";
        reference
          "https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xm
l";
      }

      /* Typedefs */

      typedef icmpv6-type-name {
        type enumeration {
          enum DestinationUnreachable {
            value 1;
            description
              "Destination Unreachable.";
            reference
              "RFC 4443";
          }
          enum PacketTooBig {
            value 2;
            description
              "Packet Too Big.";
            reference
              "RFC 4443";
          }
          enum TimeExceeded {
            value 3;
            description
              "Time Exceeded.";
            reference
              "RFC 4443";
          }
          enum ParameterProblem {
            value 4;
            description
              "Parameter Problem.";
            reference
              "RFC 4443";
          }
          enum EchoRequest {
```

```
          value 128;
          description
            "Echo Request.";
          reference
            "RFC 4443";
        }
        enum EchoReply {
          value 129;
          description
            "Echo Reply.";
          reference
            "RFC 4443";
        }
        enum MulticastListenerQuery {
          value 130;
          description
            "Multicast Listener Query.";
          reference
            "RFC 2710";
        }
        enum MulticastListenerReport {
          value 131;
          description
            "Multicast Listener Report.";
          reference
            "RFC 2710";
        }
        enum MulticastListenerDone {
          value 132;
          description
            "Multicast Listener Done.";
          reference
            "RFC 2710";
        }
        enum RouterSolicitation {
          value 133;
          description
            "Router Solicitation.";
          reference
            "RFC 4861";
        }
        enum RouterAdvertisement {
          value 134;
          description
            "Router Advertisement.";
          reference
            "RFC 4861";
        }
```

```
        enum NeighborSolicitation {
          value 135;
          description
            "Neighbor Solicitation.";
          reference
            "RFC 4861";
        }
        enum NeighborAdvertisement {
          value 136;
          description
            "Neighbor Advertisement.";
          reference
            "RFC 4861";
        }
        enum RedirectMessage {
          value 137;
          description
            "Redirect Message.";
          reference
            "RFC 4861";
        }
        enum RouterRenumbering {
          value 138;
          description
            "Router Renumbering.";
          reference
            "RFC 2894";
        }
        enum ICMPNodeInformationQuery {
          value 139;
          description
            "ICMP Node Information Query.";
          reference
            "RFC 4620";
        }
        enum ICMPNodeInformationResponse {
          value 140;
          description
            "ICMP Node Information Response.";
          reference
            "RFC 4620";
        }
        enum InverseNeighborDiscoverySolicitationMessage {
          value 141;
          description
            "Inverse Neighbor Discovery Solicitation Message.";
          reference
            "RFC 3122";
```

```
        }
        enum InverseNeighborDiscoveryAdvertisementMessage {
          value 142;
          description
            "Inverse Neighbor Discovery Advertisement Message.";
          reference
            "RFC 3122";
        }
        enum Version2MulticastListenerReport {
          value 143;
          description
            "Version 2 Multicast Listener Report.";
          reference
            "RFC 3810";
        }
        enum HomeAgentAddressDiscoveryRequestMessage {
          value 144;
          description
            "Home Agent Address Discovery Request Message.";
          reference
            "RFC 6275";
        }
        enum HomeAgentAddressDiscoveryReplyMessage {
          value 145;
          description
            "Home Agent Address Discovery Reply Message.";
          reference
            "RFC 6275";
        }
        enum MobilePrefixSolicitation {
          value 146;
          description
            "Mobile Prefix Solicitation.";
          reference
            "RFC 6275";
        }
        enum MobilePrefixAdvertisement {
          value 147;
          description
            "Mobile Prefix Advertisement.";
          reference
            "RFC 6275";
        }
        enum CertificationPathSolicitationMessage {
          value 148;
          description
            "Certification Path Solicitation Message.";
          reference
```

```
            "RFC 3971";
          }
          enum CertificationPathAdvertisementMessage {
            value 149;
            description
              "Certification Path Advertisement Message.";
            reference
              "RFC 3971";
          }
          enum ICMPmessagesutilizedbyexperimentalmobilityprotocolssuchasSeamoby {
            value 150;
            description
              "ICMP messages utilized by experimental mobility protocols
               such as Seamoby.";
            reference
              "RFC 4065";
          }
          enum MulticastRouterAdvertisement {
            value 151;
            description
              "Multicast Router Advertisement.";
            reference
              "RFC 4286";
          }
          enum MulticastRouterSolicitation {
            value 152;
            description
              "Multicast Router Solicitation.";
            reference
              "RFC 4286";
          }
          enum MulticastRouterTermination {
            value 153;
            description
              "Multicast Router Termination.";
            reference
              "RFC 4286";
          }
          enum FMIPv6Messages {
            value 154;
            description
              "FMIPv6 Messages.";
            reference
              "RFC 5568";
          }
          enum RPLControlMessage {
            value 155;
            description
```

```
            "RPL Control Message.";
          reference
            "RFC 6550";
        }
        enum ILNPv6LocatorUpdateMessage {
          value 156;
          description
            "ILNPv6 Locator Update Message.";
          reference
            "RFC 6743";
        }
        enum DuplicateAddressRequest {
          value 157;
          description
            "Duplicate Address Request.";
          reference
            "RFC 6775";
        }
        enum DuplicateAddressConfirmation {
          value 158;
          description
            "Duplicate Address Confirmation.";
          reference
            "RFC 6775";
        }
        enum MPLControlMessage {
          value 159;
          description
            "MPL Control Message.";
          reference
            "RFC 7731";
        }
        enum ExtendedEchoRequest {
          value 160;
          description
            "Extended Echo Request.";
          reference
            "RFC 8335";
        }
        enum ExtendedEchoReply {
          value 161;
          description
            "Extended Echo Reply.";
          reference
            "RFC 8335";
        }
      }
      description
```

```
        "This enumeration type defines mnemonic names and corresponding
         numeric values of ICMPv6 types.";
      reference
        "RFC 2708: IANA Allocation Guidelines For Values In the
         Internet Protocol and Related Headers";
    }

    typedef icmpv6-type {
      type union {
        type uint8;
        type icmpv6-type-name;
      }
      description
        "This type allows reference to an ICMPv6 type using either the
         assigned mnemonic name or numeric value.";
    }
  }
  <CODE ENDS>
```

Appendix C. IPv6 Extension Header Types

C.1. XSLT Template to Generate The IPv6 Extension Header Types IANA-Maintained Module

```
  <CODE BEGINS>
  <?xml version="1.0" encoding="utf-8"?>
  <stylesheet
      xmlns="http://www.w3.org/1999/XSL/Transform"
      xmlns:html="http://www.w3.org/1999/xhtml"
      xmlns:iana="http://www.iana.org/assignments"
      xmlns:yin="urn:ietf:params:xml:ns:yang:yin:1"
      version="1.0">
    <import href="../../../xslt/iana-yinx.xsl"/>
    <output method="xml" encoding="utf-8"/>
    <strip-space elements="*"/>

    <template match="iana:registry[@id='extension-header']">
      <element name="yin:typedef">
        <attribute name="name">ipv6-extension-header-type-name</attribute>
        <element name="yin:type">
          <attribute name="name">enumeration</attribute>
          <apply-templates
              select="iana:record[not(iana:description = 'Unassigned' or
                      starts-with(iana:description, 'Reserved') or
                      starts-with(iana:description, 'Use for experimentation and
testing')) or
                      contains(iana:description, 'experimental')]"/>
        </element>
        <element name="yin:description">
```

```
          <element name="yin:text">
            This enumeration type defines mnemonic names and
            corresponding numeric values of IPv6 Extension header types.
          </element>
        </element>
        <element name="yin:reference">
          <element name="yin:text">
            RFC 2708: IANA Allocation Guidelines For Values In
                      the Internet Protocol and Related Headers
          </element>
        </element>
      </element>
      <element name="yin:typedef">
        <attribute name="name">ipv6-extension-header-type</attribute>
        <element name="yin:type">
          <attribute name="name">union</attribute>
          <element name="yin:type">
            <attribute name="name">uint8</attribute>
          </element>
          <element name="yin:type">
            <attribute name="name">ipv6-extension-header-type-name</attribute>
          </element>
        </element>
        <element name="yin:description">
          <element name="yin:text">
            This type allows reference to an IPv6 Extension header type using ei
ther
            the assigned mnemonic name or the numeric protocol number value.
          </element>
        </element>
      </element>
    </template>

    <template match="iana:record">
      <call-template name="enum">
        <with-param name="id">
          <choose>
            <when test="contains(iana:description, '(Deprecated)')">
              <value-of select="translate(normalize-space(substring-before(iana:
description,
                    '(Deprecated)')),' ','')"/>
            </when>
            <otherwise>
              <value-of select="translate(normalize-space(iana:description),' ',
'')"/>
            </otherwise>
          </choose>
        </with-param>
        <with-param name="deprecated"
                    select="contains(iana:description, '(Deprecated)')"/>
      </call-template>
```

```
    </template>

  </stylesheet>
  <CODE ENDS>
```

C.2. Initial Version of the The ICMPv4 Types IANA-Maintained Module

```
  <CODE BEGINS> file "iana-ipv6-ext-types@2023-09-29.yang"
  module iana-ipv6-ext-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types";
    prefix iana-ipv6-ext-types;

    organization
      "Internet Assigned Numbers Authority (IANA)";

    contact
      "Internet Assigned Numbers Authority

       ICANN
       12025 Waterfront Drive, Suite 300
       Los Angeles, CA 90094


       Tel: +1 424 254 5300

       <mailto:iana@iana.org>";

    description
      "This YANG module translates IANA registry 'IPv6 Extension Header
       Types' to YANG derived types.

       Copyright (c) 2023 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 Revised 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).

       This version of this YANG module was generated from the
       corresponding IANA registry using an XSLT stylesheet from the
       'iana-yang' project (https://github.com/llhotka/iana-yang).";

    reference
      "Internet Protocol Version 6 (IPv6) Parameters
```

```
        (https://www.iana.org/assignments/ipv6-parameters/)";

    revision 2023-09-29 {
      description
        "Current revision as of the revision date specified in the XML
         representation of the registry page.";
      reference
        "https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xml";
    }

    /* Typedefs */

    typedef ipv6-extension-header-type-name {
      type enumeration {
        enum IPv6Hop-by-HopOption {
          value 0;
          description
            "IPv6 Hop-by-Hop Option";
          reference
            "RFC 8200";
        }
        enum RoutingHeaderforIPv6 {
          value 43;
          description
            "Routing Header for IPv6";
          reference
            "- RFC 8200
             - RFC 5095";
        }
        enum FragmentHeaderforIPv6 {
          value 44;
          description
            "Fragment Header for IPv6";
          reference
            "RFC 8200";
        }
        enum EncapsulatingSecurityPayload {
          value 50;
          description
            "Encapsulating Security Payload";
          reference
            "RFC 4303";
        }
        enum AuthenticationHeader {
          value 51;
          description
            "Authentication Header";
          reference
```

```
            "RFC 4302";
        }
        enum DestinationOptionsforIPv6 {
          value 60;
          description
            "Destination Options for IPv6";
          reference
            "RFC 8200";
        }
        enum MobilityHeader {
          value 135;
          description
            "Mobility Header";
          reference
            "RFC 6275";
        }
        enum HostIdentityProtocol {
          value 139;
          description
            "Host Identity Protocol";
          reference
            "RFC 7401";
        }
        enum Shim6Protocol {
          value 140;
          description
            "Shim6 Protocol";
          reference
            "RFC 5533";
        }
      }
      description
        "This enumeration type defines mnemonic names and corresponding
         numeric values of IPv6 Extension header types.";
      reference
        "RFC 2708: IANA Allocation Guidelines For Values In the
         Internet Protocol and Related Headers";
    }

    typedef ipv6-extension-header-type {
      type union {
        type uint8;
        type ipv6-extension-header-type-name;
      }
      description
        "This type allows reference to an IPv6 Extension header type
         using either the assigned mnemonic name or the numeric
         protocol number value.";
```

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

Appendix D. Problem Statement & Gap Analysis

D.1. Suboptimal Configuration: Lack of Support for Lists of Prefixes

IP prefix-related data nodes, e.g., "destination-ipv4-network" or "destination-ipv6-network", do not support handling a list of IP prefixes, which may then lead to having to support large numbers of ACL entries in a configuration file.

The same issue is encountered when ACLs have to be in place to mitigate DDoS attacks that involve a set of sources (e.g., [RFC9132]). The situation is even worse when both a list of sources and destination prefixes are involved in the filtering.

Figure 9 shows an example of the required ACL configuration for filtering traffic from two prefixes.

```
{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "first-prefix",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "my-test-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network":
                    "2001:db8:6401:1::/64",
                  "source-ipv6-network":
                    "2001:db8:1234::/96",
                  "protocol": 17,
                  "flow-label": 10000
                },
                "udp": {
                  "source-port": {
                    "operator": "lte",
                    "port": 80
                  },
                  "destination-port": {
                    "operator": "neq",
                    "port": 1010
```

```
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        },
        {
          "name": "second-prefix",
          "type": "ipv6-acl-type",
          "aces": {
            "ace": [
              {
                "name": "my-test-ace",
                "matches": {
                  "ipv6": {
                    "destination-ipv6-network":
                      "2001:db8:6401:c::/64",
                    "source-ipv6-network":
                      "2001:db8:1234::/96",
                    "protocol": 17,
                    "flow-label": 10000
                  },
                  "udp": {
                    "source-port": {
                      "operator": "lte",
                      "port": 80
                    },
                    "destination-port": {
                      "operator": "neq",
                      "port": 1010
                    }
                  }
                },
                "actions": {
                  "forwarding": "accept"
                }
              }
            ]
          }
        }
      ]
    }
  }
```

Figure 9: Example Illustrating Sub-optimal Use of the ACL Model with a Prefix List (Message Body)

Such a configuration is suboptimal for both:

* Network controllers that need to manipulate large files. All or a subset for this configuration will need to be passed to the underlying network devices.

* Devices may receive such a configuration and thus will need to maintain it locally.

D.2. Manageability: Impossibility to Use Aliases or Defined Sets

The same approach as the one discussed for IP prefixes can be generalized by introducing the concept of "aliases" or "defined sets".

The defined sets are reusable definitions across several ACLs. Each category is modelled in YANG as a list of parameters related to the class it represents. The following sets can be considered:

* Prefix sets: Used to create lists of IPv4 or IPv6 prefixes.

* Protocol sets: Used to create a list of protocols.

* Port number sets: Used to create lists of TCP or UDP port values (or any other transport protocol that makes uses of port numbers). The identity of the protocols is identified by the protocol set, if present. Otherwise, a set applies to any protocol.

* ICMP sets: Uses to create lists of ICMP-based filters. This applies only when the protocol is set to ICMP or ICMPv6.

Aliases may also be considered to manage resources that are identified by a combination of various parameters (e.g., prefix, protocol, port number, FQDN, or VLAN IDs). Note that some aliases can be provided by decomposing them into separate sets.

D.3. Bind ACLs to Devices, Not Only Interfaces

In the context of network management, an ACL may be enforced in many network locations. As such, the ACL module should allow for binding an ACL to multiple devices, not only (abstract) interfaces.

The ACL name must, thus, be unique at the scale of the network, but the same name may be used in many devices when enforcing node-specific ACLs.

D.4. Partial or Lack of IPv4/IPv6 Fragment Handling

[RFC8519] does not support fragment handling for IPv6 but offers a partial support for IPv4 through the use of 'flags'. Nevertheless, the use of 'flags' is problematic since it does not allow a bitmask to be defined. For example, setting other bits not covered by the 'flags' filtering clause in a packet will allow that packet to get through (because it won't match the ACE).

Defining a new IPv4/IPv6 matching field called 'fragment' is thus required to efficiently handle fragment-related filtering rules.

D.5. Suboptimal TCP Flags Handling

[RFC8519] supports including flags in the TCP match fields, however that structure does not support matching operations as those supported in BGP Flow Spec. Defining this field to be defined as a flag bitmask together with a set of operations is meant to efficiently handle TCP flags filtering rules.

D.6. Rate-Limit Action

[RFC8519] specifies that forwarding actions can be 'accept' (i.e., accept matching traffic), 'drop' (i.e., drop matching traffic without sending any ICMP error message), or 'reject' (i.e., drop matching traffic and send an ICMP error message to the source). However, there are situations where the matching traffic can be accepted, but with a rate-limit policy. This capability is not supported by [RFC8519].

D.7. Payload-based Filtering

Some transport protocols use existing protocols (e.g., TCP or UDP) as substrate. The match criteria for such protocols may rely upon the 'protocol' under 'l3', TCP/UDP match criteria, part of the TCP/UDP payload, or a combination thereof. [RFC8519] does not support matching based on the payload.

Likewise, the current version of the ACL model does not support filtering of encapsulated traffic.

D.8. Reuse the ACLs Content Across Several Devices

Having a global network view of the ACLs is highly valuable for service providers. An ACL could be defined and applied based on the network topology hierarchy. So, an ACL can be defined at the network level and, then, that same ACL can be used (or referenced to) in several devices (including termination points) within the same network.

This network/device ACLs differentiation introduces several new requirements, e.g.:

* An ACL name can be used at both network and device levels.

* An ACL content updated at the network level should imply a transaction that updates the relevant content in all the nodes using this ACL.

* ACLs defined at the device level have a local meaning for the specific node.

* A device can be associated with a router, a VRF, a logical system, or a virtual node. ACLs can be applied in physical and logical infrastructure.

D.9. Match MPLS Headers

The ACLs could be used to create rules to match MPLS fields on a packet. [RFC8519] does not support such function.

Appendix E. Acknowledgements

Many thanks to Jon Shallow and Miguel Cros for the review and comments to the document, including prior to publishing the document.

Thanks to Qiufang Ma, Victor Lopez, and Joe Clarke for the comments and suggestions.

The IANA-maintained models were generated using an XSLT stylesheet from the 'iana-yang' project (https://github.com/llhotka/iana-yang).

This work is partially supported by the European Commission under Horizon 2020 Secured autonomic traffic management for a Tera of SDN flows (Teraflow) project (grant agreement number 101015857).

Authors' Addresses

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil
Telefonica
Email: samier.barguilgiraldo.ext@telefonica.com

Mohamed Boucadair
Orange
Email: mohamed.boucadair@orange.com

Qin Wu
Huawei
Email: bill.wu@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 20 April 2024

R.G. Wilton
Cisco Systems
S. Mansfield
Ericsson

18 October 2023

Common Interface Extension YANG Data Models
draft-ietf-netmod-intf-ext-yang-12

Abstract

This document defines two YANG modules that augment the Interfaces data model defined in the "YANG Data Model for Interface Management" with additional configuration and operational data nodes to support common lower layer interface properties, such as interface MTU.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

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 20 April 2024.

Copyright Notice

Table of Contents

1. Introduction

This document defines two NMDA compatible [RFC8342] YANG 1.1 [RFC7950] modules for the management of network interfaces. It defines various augmentations to the generic interfaces data model [RFC8343] to support configuration of lower layer interface properties that are common across many types of network interface.

One of the aims of this document is to provide a standard definition for these configuration items regardless of the underlying interface type. For example, a definition for configuring or reading the MAC address associated with an interface is provided that can be used for any interface type that uses Ethernet framing.

Several of the augmentations defined here are not backed by any formal standard specification. Instead, they are for features that are commonly implemented in equivalent ways by multiple independent network equipment vendors. The aim of this document is to define common paths and leaves for the configuration of these equivalent features in a uniform way, making it easier for users of the YANG model to access these features in a vendor independent way. Where necessary, a description of the expected behavior is also provided with the aim of ensuring vendors implementations are consistent with the specified behavior.

Given that the modules contain a collection of discrete features with the common theme that they generically apply to interfaces, it is plausible that not all implementers of the YANG module will decide to support all features. Hence, separate feature keywords are defined for each logically discrete feature to allow implementers the flexibility to choose which specific parts of the model they support.

The augmentations are split into two separate YANG modules that each focus on a particular area of functionality. The two YANG modules defined in this document are:

ietf-if-extensions.yang - Defines extensions to the IETF interface data model to support common configuration data nodes.

ietf-if-ethernet-like.yang - Defines a module for any configuration and operational data nodes that are common across interfaces that use Ethernet framing.

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.

1.2. Tree Diagrams

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

2. Interface Extensions Module

The Interfaces Extensions YANG module provides some basic extensions to the IETF interfaces YANG module.

The module provides:

* A link flap suppression feature used to provide control over short-lived link state flaps.

* An interface link state dampening feature that is used to provide control over longer lived link state flaps.

* An encapsulation container and extensible choice statement for use by any interface types that allow for configurable L2 encapsulations.

* A loopback configuration leaf that is primarily aimed at loopback at the physical layer.

* MTU configuration leaves applicable to all packet/frame based interfaces.

* A forwarding mode leaf to indicate the OSI layer at which the interface handles traffic.

* A generic "sub-interface" identity that an interface identity definition can derive from if it defines a sub-interface.

* A parent interface leaf useable for all types of sub-interface that are children of parent interfaces.

The "ietf-if-extensions" YANG module has the following structure:

```
module: ietf-if-extensions
  augment /if:interfaces/if:interface:
    +--rw link-flap-suppression {link-flap-suppression}?
    |  +--rw down?                   uint32
    |  +--rw up?                     uint32
    |  +--ro carrier-transitions?    yang:counter64
    |  +--ro timer-running?          enumeration
    +--rw dampening! {dampening}?
    |  +--rw half-life?            uint32
    |  +--rw reuse?                uint32
    |  +--rw suppress?             uint32
    |  +--rw max-suppress-time?    uint32
    |  +--ro penalty?              uint32
    |  +--ro suppressed?           boolean
    |  +--ro time-remaining?       uint32
    +--rw encapsulation
    |  +--rw (encaps-type)?
    +--rw loopback?           identityref {loopback}?
    +--rw max-frame-size?     uint32 {max-frame-size}?
    +--ro forwarding-mode?    identityref
  augment /if:interfaces/if:interface:
    +--rw parent-interface if:interface-ref {sub-interfaces}?
  augment /if:interfaces/if:interface/if:statistics:
    +--ro in-discard-unknown-encaps?   yang:counter64
            {sub-interfaces}?
```

2.1. Link Flap Suppression

The link flap suppression feature augments the IETF interfaces data model with configuration for a simple algorithm that is used, generally on physical interfaces, to suppress short transient changes in the interface link state. It can be used in conjunction with the dampening feature described in Section 2.2 to provide effective control of unstable links and unwanted state transitions.

The principle of the link flap suppression feature is to use a short per interface timer to ensure that any interface link state transition that occurs and reverts back within the specified time interval is entirely suppressed without providing any signalling to any upper layer protocols that the state transition has occurred. E.g. in the case that the link state transition is suppressed then there is no change of the /if:interfaces/if:interface/oper-status or /if:interfaces/if:interfaces/last-change leaves for the interface that the feature is operating on. One obvious side effect of using this feature that is that any state transition will always be delayed by the specified time interval.

The configuration allows for separate timer values to be used in the suppression of down->up->down link transitions vs up->down->up link transitions.

The link flap suppression down timer leaf specifies the amount of time that an interface that is currently in link up state must be continuously down before the down state change is reported to higher level protocols. Use of this timer can cause traffic to be black holed for the configured value and delay reconvergence after link failures, therefore its use is normally restricted to cases where it is necessary to allow enough time for another protection mechanism (such as an optical layer automatic protection system) to take effect.

The link flap suppression up timer leaf specifies the amount of time that an interface that is currently in link down state must be continuously up before the down->up link state transition is reported to higher level protocols. This timer is generally useful as a debounce mechanism to ensure that a link is relatively stable before being brought into service. It can also be used effectively to limit the frequency at which link state transition events may occur. The default value for this leaf is determined by the underlying network device.

2.2. Dampening

The dampening feature introduces a configurable exponential decay mechanism to suppress the effects of excessive interface link state flapping. This feature allows the network operator to configure a device to automatically identify and selectively dampen a local interface which is flapping. Dampening an interface keeps the interface operationally down until the interface stops flapping and becomes stable. Configuring the dampening feature can improve convergence times and stability throughout the network by isolating failures so that disturbances are not propagated, which reduces the utilization of system processing resources by other devices in the network and improves overall network stability.

The basic algorithm uses a counter that is increased by 1000 units every time the underlying interface link state changes from up to down. If the counter increases above the suppress threshold then the interface is kept down (and out of service) until either the maximum suppression time is reached, or the counter has reduced below the reuse threshold. The half-life period determines that rate at which the counter is periodically reduced by half.

2.2.1. Suppress Threshold

The suppress threshold is the value of the accumulated penalty that triggers the device to dampen a flapping interface. The flapping interface is identified by the device and assigned a penalty for each up to down link state change, but the interface is not automatically dampened. The device tracks the penalties that a flapping interface accumulates. When the accumulated penalty reaches or exceeds the suppress threshold, the interface is placed in a suppressed state.

2.2.2. Half-Life Period

The half-life period determines how fast the accumulated penalties can decay exponentially. The accumulated penalty decays at a rate that causes its value to be reduced by half after each half-life period.

2.2.3. Reuse Threshold

If, after one or more half-life periods, the accumulated penalty decreases below the reuse threshold and the underlying interface link state is up then the interface is taken out of suppressed state and is allowed to go up.

2.2.4. Maximum Suppress Time

The maximum suppress time represents the maximum amount of time an interface can remain dampened when a new penalty is assigned to an interface. The default of the maximum suppress timer is four times the half-life period. The maximum value of the accumulated penalty is calculated using the maximum suppress time, reuse threshold and half-life period.

2.3. Encapsulation

The encapsulation container holds a choice node that is to be augmented with datalink layer specific encapsulations, such as HDLC, PPP, or sub-interface 802.1Q tag match encapsulations. The use of a choice statement ensures that an interface can only have a single datalink layer protocol configured.

The different encapsulations themselves are defined in separate YANG modules defined in other documents that augment the encapsulation choice statement. For example the Ethernet specific basic 'dot1q-vlan' encapsulation is defined in ietf-if-l3-vlan.yang and the 'flexible' encapsulation is defined in ietf-flexible-encapsulation.yang, both modules from [I-D.ietf-netmod-sub-intf-vlan-model].

2.4. Loopback

The loopback configuration leaf allows any physical interface to be configured to be in one of the possible following physical loopback modes, i.e. internal loopback, line loopback, or use of an external loopback connector. The use of YANG identities allows for the model to be extended with other modes of loopback if required.

The following loopback modes are defined:

* Internal loopback - All egress traffic on the interface is internally looped back within the interface to be received on the ingress path.

* Line loopback - All ingress traffic received on the interface is internally looped back within the interface to the egress path.

* Loopback Connector - The interface has a physical loopback connector attached that loops all egress traffic back into the interface's ingress path, with equivalent semantics to internal loopback.

2.5. Maximum frame size

A maximum frame size configuration leaf (max-frame-size) is provided to specify the maximum size of a layer 2 frame that may be transmitted or received on an interface. The value includes the overhead of any layer 2 header, the maximum length of the payload, and any frame check sequence (FCS) bytes. If configured, the max-frame-size leaf on an interface also restricts the max-frame-size of any child sub-interfaces, and the available MTU for protocols.

2.6. Sub-interface

The sub-interface feature specifies the minimal leaves required to define a child interface that is parented to another interface.

A sub-interface is a logical interface that handles a subset of the traffic on the parent interface. Separate configuration leaves are used to classify the subset of ingress traffic received on the parent interface to be processed in the context of a given sub-interface. All egress traffic processed on a sub-interface is given to the parent interface for transmission. Otherwise, a sub-interface is like any other interface in /if:interfaces and supports the standard interface features and configuration.

For some vendor specific interface naming conventions the name of the child interface is sufficient to determine the parent interface, which implies that the child interface can never be reparented to a different parent interface after it has been created without deleting the existing sub-interface and recreating a new sub-interface. Even in this case it is useful to have a well-defined leaf to cleanly identify the parent interface.

The model also allows for arbitrarily named sub-interfaces by having an explicit parent-interface leaf define the child -> parent relationship. In this naming scenario it is also possible for implementations to allow for logical interfaces to be reparented to new parent interfaces without needing the sub-interface to be destroyed and recreated.

2.7. Forwarding Mode

The forwarding mode leaf provides additional information as to what mode or layer an interface is logically operating and forwarding traffic at. The implication of this leaf is that for traffic forwarded at a given layer that any headers for lower layers are stripped off before the packet is forwarded at the given layer. Conversely, on egress any lower layer headers must be added to the packet before it is transmitted out of the interface.

The following forwarding modes are defined:

* Physical - Traffic is being forwarded at the physical layer. This includes DWDM or OTN based switching.

* Data-link - Layer 2 based forwarding, such as Ethernet/VLAN based switching, or L2VPN services.

* Network - Network layer based forwarding, such as IP, MPLS, or L3VPNs.

3. Interfaces Ethernet-Like Module

The Interfaces Ethernet-Like Module is a small module that contains all configuration and operational data that is common across interface types that use Ethernet framing as their datalink layer encapsulation.

This module currently contains leaves for the configuration and reporting of the operational MAC address and the burnt-in MAC address (BIA) associated with any interface using Ethernet framing.

The "ietf-if-ethernet-like" YANG module has the following structure:

```
module: ietf-if-ethernet-like
  augment /if:interfaces/if:interface:
    +--rw ethernet-like
       +--rw mac-address?       yang:mac-address
       |       {configurable-mac-address}?
       +--ro bia-mac-address?   yang:mac-address
  augment /if:interfaces/if:interface/if:statistics:
    +--ro in-drop-unknown-dest-mac-pkts?   yang:counter64
    +--ro in-discard-overflows?            yang:counter64
```

4. Interface Extensions YANG Module

This YANG module augments the interface container defined in [RFC8343]. It also contains references to [RFC6991] and [RFC7224].

```
<CODE BEGINS> file "ietf-if-extensions@2023-01-26.yang"
module ietf-if-extensions {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-if-extensions";

  prefix if-ext;

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

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }

  import iana-if-type {
    prefix ianaift;
    reference "RFC 7224: IANA Interface Type YANG Module";
  }

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

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

     Editor:   Robert Wilton
```

```
                <mailto:rwilton@cisco.com>";

    description
      "This module contains common definitions for extending the IETF
       interface YANG model (RFC 8343) with common configurable layer 2
       properties.

       Copyright (c) 2023 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 Revised 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).

       This version of this YANG module is part of RFC XXXX
       (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
       for full legal notices.

       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 (RFC 2119) (RFC 8174) when, and only when,
       they appear in all capitals, as shown here.";

    revision 2023-01-26 {
      description
        "Initial revision.";

      reference
        "RFC XXXX, Common Interface Extension YANG Data Models";
    }

    feature link-flap-suppression {
      description
        "This feature indicates that configurable interface link
         delay is supported, which is a feature is used to limit the
         propagation of very short interface link state flaps.";
      reference "RFC XXXX, Section 2.1 Link Flap Suppression";
    }

    feature dampening {
      description
        "This feature indicates that the device supports interface
         dampening, which is a feature that is used to limit the
         propagation of interface link state flaps over longer
```

```
         periods.";
      reference "RFC XXXX, Section 2.2 Dampening";
    }

    feature loopback {
      description
        "This feature indicates that configurable interface loopback is
         supported.";
      reference "RFC XXXX, Section 2.4 Loopback";
    }

    feature max-frame-size {
      description
        "This feature indicates that the device supports configuring or
         reporting the maximum frame size on interfaces.";
      reference "RFC XXXX, Section 2.5 Maximum Frame Size";
    }

    feature sub-interfaces {
      description
        "This feature indicates that the device supports the
         instantiation of sub-interfaces.  Sub-interfaces are defined
         as logical child interfaces that allow features and forwarding
         decisions to be applied to a subset of the traffic processed
         on the specified parent interface.";
      reference "RFC XXXX, Section 2.6 Sub-interface";
    }

    /*
     * Define common identities to help allow interface types to be
     * assigned properties.
     */
    identity sub-interface {
      description
        "Base type for generic sub-interfaces.

         New or custom interface types can derive from this type to
         inherit generic sub-interface configuration.";
      reference "RFC XXXX, Section 2.6 Sub-interface";
    }

    identity ethSubInterface{
      base ianaift:l2vlan;
      base sub-interface;

      description
        "This identity represents the child sub-interface of any
         interface types that uses Ethernet framing (with or without
```

```
          802.1Q tagging).";
      }

      identity loopback {
        description "Base identity for interface loopback options";
        reference "RFC XXXX, Section 2.4";
      }
      identity internal {
        base loopback;
        description
          "All egress traffic on the interface is internally looped back
           within the interface to be received on the ingress path.";
        reference "RFC XXXX, Section 2.4";
      }
      identity line {
        base loopback;
        description
          "All ingress traffic received on the interface is internally
           looped back within the interface to the egress path.";
        reference "RFC XXXX, Section 2.4";
      }
      identity connector {
        base loopback;
        description
          "The interface has a physical loopback connector attached that
           loops all egress traffic back into the interface's ingress
           path, with equivalent semantics to loopback internal.";
        reference "RFC XXXX, Section 2.4";
      }


      identity forwarding-mode {
        description "Base identity for forwarding-mode options.";
        reference "RFC XXXX, Section 2.7";
      }
      identity physical {
        base forwarding-mode;
        description
          "Physical layer forwarding.  This includes DWDM or OTN based
           optical switching.";
        reference "RFC XXXX, Section 2.7";
      }
      identity data-link {
        base forwarding-mode;
        description
          "Layer 2 based forwarding, such as Ethernet/VLAN based
           switching, or L2VPN services.";
        reference "RFC XXXX, Section 2.7";
```

```
  }
  identity network {
    base forwarding-mode;
    description
      "Network layer based forwarding, such as IP, MPLS, or L3VPNs.";
    reference "RFC XXXX, Section 2.7";
  }


  /*
   * Augments the IETF interfaces model with leaves to configure
   * and monitor link-flap-suppression on an interface.
   */
  augment "/if:interfaces/if:interface" {
    description
      "Augments the IETF interface model with optional common
       interface level commands that are not formally covered by any
       specific standard.";

    /*
     * Defines standard YANG for the Link Flap Suppression feature.
     */
    container link-flap-suppression {
      if-feature "link-flap-suppression";
      description
        "Holds link flap related feature configuration.";
      leaf down {
        type uint32;
        units milliseconds;
        description
          "Delays the propagation of a 'loss of carrier signal' event
           that would cause the interface state to go down, i.e. the
           command allows short link flaps to be suppressed. The
           configured value indicates the minimum time interval (in
           milliseconds) that the link signal must be continuously
           down before the interface state is brought down. If not
           configured, the behavior on loss of link signal is
           vendor/interface specific, but with the general
           expectation that there should be little or no delay.";
      }
      leaf up {
        type uint32;
        units milliseconds;
        description
          "Defines the minimum time interval (in milliseconds) that
           the link signal must be continuously present and error
           free before the interface state is allowed to transition
           from down to up.  If not configured, the behavior is
```

```
            vendor/interface specific, but with the general
            expectation that sufficient default delay should be used
            to ensure that the interface is stable when enabled before
            being reported as being up.  Configured values that are
            too low for the hardware capabilties may be rejected.";
        }
        leaf carrier-transitions {
          type yang:counter64;
          units transitions;
          config false;
          description
            "Defines the number of times the underlying link state
             has changed to, or from, state up.  This counter should be
             incremented even if the high layer interface state changes
             are being suppressed by a running link flap suppression
             timer.";
        }
        leaf timer-running {
          type enumeration {
            enum none {
              description
                "No link flap suppression timer is running.";
            }
            enum up {
              description
                "link-flap-suppression up timer is running.  The
                 underlying link state is up, but interface state is
                 not reported as up.";
            }
            enum down {
              description
                "link-flap-suppression down timer is running.
                 Interface state is reported as up, but the underlying
                 link state is actually down.";
            }
          }
          config false;
          description
            "Reports whether a link flap suppression timer is actively
             running, in which case the interface state does not match
             the underlying link state.";
        }

        reference "RFC XXXX, Section 2.1 Link Flap Suppression";
      }

      /*
       * Augments the IETF interfaces model with a container to hold
```

```
        * generic interface dampening
        */
       container dampening {
         if-feature "dampening";
         presence
           "Enable interface link flap dampening with default settings
            (that are vendor/device specific).";
         description
           "Interface dampening limits the propagation of interface link
            state flaps over longer periods.";
         reference "RFC XXXX, Section 2.2 Dampening";

         leaf half-life {
           type uint32;
           units seconds;
           description
             "The time (in seconds) after which a penalty would be half
              its original value.  Once the interface has been assigned
              a penalty, the penalty is decreased at a decay rate
              equivalent to the half-life.  For some devices, the
              allowed values may be restricted to particular multiples
              of seconds.  The default value is vendor/device
              specific.";
           reference "RFC XXXX, Section 2.3.2 Half-Life Period";
         }

         leaf reuse {
           type uint32;
           description
             "Penalty value below which a stable interface is
              unsuppressed (i.e. brought up) (no units).  The default
              value is vendor/device specific.  The penalty value for a
              link up->down state change is 1000 units.";
           reference "RFC XXXX, Section 2.2.3 Reuse Threshold";
         }

         leaf suppress {
           type uint32;
           description
             "Limit at which an interface is suppressed (i.e. held down)
              when its penalty exceeds that limit (no units).  The value
              must be greater than the reuse threshold.  The default
              value is vendor/device specific.  The penalty value for a
              link up->down state change is 1000 units.";
           reference "RFC XXXX, Section 2.2.1 Suppress Threshold";
         }

         leaf max-suppress-time {
```

```
          type uint32;
          units seconds;
          description
            "Maximum time (in seconds) that an interface can be
             suppressed before being unsuppressed if no further link
             up->down state change penalties have been applied.  This
             value effectively acts as a ceiling that the penalty value
             cannot exceed.  The default value is vendor/device
             specific.";
          reference "RFC XXXX, Section 2.2.4 Maximum Suppress Time";
        }

        leaf penalty {
          type uint32;
          config false;
          description
            "The current penalty value for this interface.  When the
             penalty value exceeds the 'suppress' leaf then the
             interface is suppressed (i.e. held down).";
          reference "RFC XXXX, Section 2.2 Dampening";
        }

        leaf suppressed {
          type boolean;
          config false;
          description
            "Represents whether the interface is suppressed (i.e. held
             down) because the 'penalty' leaf value exceeds the
             'suppress' leaf.";
          reference "RFC XXXX, Section 2.2 Dampening";
        }

        leaf time-remaining {
          when '../suppressed = "true"' {
            description
              "Only suppressed interfaces have a time remaining.";
          }
          type uint32;
          units seconds;
          config false;
          description
            "For a suppressed interface, this leaf represents how long
             (in seconds) that the interface will remain suppressed
             before it is allowed to go back up again.";
          reference "RFC XXXX, Section 2.2 Dampening";
        }
      }
```

```
      /*
       * Various types of interfaces support a configurable layer 2
       * encapsulation, any that are supported by YANG should be
       * listed here.
       *
       * Different encapsulations can hook into the common encaps-type
       * choice statement.
       */
      container encapsulation {
        when
          "derived-from-or-self(../if:type,
                                'ianaift:ethernetCsmacd') or
           derived-from-or-self(../if:type,
                                'ianaift:ieee8023adLag') or
           derived-from-or-self(../if:type, 'ianaift:pos') or
           derived-from-or-self(../if:type,
                                'ianaift:atmSubInterface') or
           derived-from-or-self(../if:type, 'ianaift:l2vlan') or
           derived-from-or-self(../if:type, 'ethSubInterface')" {

          description
            "All interface types that can have a configurable L2
             encapsulation.";
        }

        description
          "Holds the OSI layer 2 encapsulation associated with an
           interface.";
        choice encaps-type {
          description
            "Extensible choice of layer 2 encapsulations";
          reference "RFC XXXX, Section 2.3 Encapsulation";
        }
      }

       /*
        * Various types of interfaces support loopback configuration,
        * any that are supported by YANG should be listed here.
        */
      leaf loopback {
        when "derived-from-or-self(../if:type,
                                   'ianaift:ethernetCsmacd') or
              derived-from-or-self(../if:type, 'ianaift:sonet') or
              derived-from-or-self(../if:type, 'ianaift:atm') or
              derived-from-or-self(../if:type, 'ianaift:otnOtu')" {
          description
            "All interface types that support loopback configuration.";
        }
```

```
        if-feature "loopback";
        type identityref {
          base loopback;
        }
        description "Enables traffic loopback.";
        reference "RFC XXXX, Section 2.4 Loopback";
      }

      /*
       * Allows the maximum frame size to be configured or reported.
       */
      leaf max-frame-size {
        if-feature "max-frame-size";
        type uint32 {
          range "64 .. max";
        }
        description
          "The maximum size of layer 2 frames that may be transmitted
           or received on the interface (including any frame header,
           maximum frame payload size, and frame checksum sequence).

           If configured, the max-frame-size also limits the maximum
           frame size of any child sub-interfaces.  The MTU available
           to higher layer protocols is restricted to the maximum frame
           payload size, and MAY be further restricted by explicit
           layer 3 or protocol specific MTU configuration.";

        reference "RFC XXXX, Section 2.5 Maximum Frame Size";
      }

      /*
       * Augments the IETF interfaces model with a leaf that indicates
       * which mode, or layer, is being used to forward the traffic.
       */
      leaf forwarding-mode {
        type identityref {
          base forwarding-mode;
        }
        config false;

        description
          "The forwarding mode that the interface is operating in.";
        reference "RFC XXXX, Section 2.7 Forwarding Mode";
      }
    }

    /*
     * Add generic support for sub-interfaces.
```

```
     *
     * This should be extended to cover all interface types that are
     * child interfaces of other interfaces.
     */
    augment "/if:interfaces/if:interface" {
      when "derived-from(if:type, 'sub-interface') or
            derived-from-or-self(if:type, 'ianaift:l2vlan') or
            derived-from-or-self(if:type, 'ianaift:atmSubInterface') or
            derived-from-or-self(if:type, 'ianaift:frameRelay')"  {
        description
          "Any ianaift:types that explicitly represent sub-interfaces
           or any types that derive from the sub-interface identity.";
      }
      if-feature "sub-interfaces";

      description
        "Adds a parent interface field to interfaces that model
         sub-interfaces.";
      leaf parent-interface {

        type if:interface-ref;

        mandatory true;
        description
          "This is the reference to the parent interface of this
           sub-interface.";
        reference "RFC XXXX, Section 2.6 Sub-interface";
      }
    }

    /*
     * Add discard counter for unknown sub-interface encapsulation
     */
    augment "/if:interfaces/if:interface/if:statistics" {
      when "derived-from-or-self(../if:type,
                                 'ianaift:ethernetCsmacd') or
            derived-from-or-self(../if:type,
                                 'ianaift:ieee8023adLag') or
            derived-from-or-self(../if:type, 'ianaift:ifPwType')" {
        description
          "Applies to interfaces that can demultiplex ingress frames to
           sub-interfaces.";
      }
      if-feature "sub-interfaces";

      description
        "Augment the interface model statistics with a sub-interface
         demux discard counter.";
```

```
      leaf in-discard-unknown-encaps {
        type yang:counter64;
        units frames;
        description
          "A count of the number of frames that were well formed, but
           otherwise discarded because their encapsulation does not
           classify the frame to the interface or any child
           sub-interface.  E.g., a frame might be discarded because the
           it has an unknown VLAN Id, or does not have a VLAN Id when
           one is expected.

           For consistency, frames counted against this counter are
           also counted against the IETF interfaces statistics.  In
           particular, they are included in in-octets and in-discards,
           but are not included in in-unicast-pkts, in-multicast-pkts
           or in-broadcast-pkts, because they are not delivered to a
           higher layer.

           Discontinuities in the values of this counter can occur at
           re-initialization of the management system, and at other
           times as indicated by the value of the 'discontinuity-time'
           leaf defined in the ietf-interfaces YANG module
           (RFC 8343).";
      }
    }
  }
  <CODE ENDS>
```

5. Interfaces Ethernet-Like YANG Module

This YANG module augments the interface container defined in RFC 8343 [RFC8343] for Ethernet-like interfaces. This includes Ethernet interfaces, 802.3 LAG (802.1AX) interfaces, Switch Virtual interfaces, and Pseudo-Wire Head-End interfaces. It also contains references to [RFC6991], [RFC7224], and [IEEE_802.3.2_2019].

```
<CODE BEGINS> file "ietf-if-ethernet-like@2023-01-26.yang"
module ietf-if-ethernet-like {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like";

  prefix ethlike;

  import ietf-interfaces {
    prefix if;
    reference
```

```
        "RFC 8343: A YANG Data Model For Interface Management";
    }

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

    import iana-if-type {
      prefix ianaift;
      reference "RFC 7224: IANA Interface Type YANG Module";
    }

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

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

       Editor:   Robert Wilton
                 <mailto:rwilton@cisco.com>";

    description
      "This module contains YANG definitions for configuration for
       'Ethernet-like' interfaces.  It is applicable to all interface
       types that use Ethernet framing and expose an Ethernet MAC
       layer, and includes such interfaces as physical Ethernet
       interfaces, Ethernet LAG interfaces and VLAN sub-interfaces.

       Additional interface configuration and counters for physical
       Ethernet interfaces are defined in
       ieee802-ethernet-interface.yang, as part of IEEE Std
       802.3.2-2019.

       Copyright (c) 2022 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 Revised 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).

       This version of this YANG module is part of RFC XXXX
       (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
       for full legal notices.";
```

```
     revision 2023-01-26 {
       description "Initial revision.";

       reference
         "RFC XXXX, Common Interface Extension YANG Data Models";
     }

     feature configurable-mac-address {
       description
         "This feature indicates that MAC addresses on Ethernet-like
          interfaces can be configured.";
       reference
         "RFC XXXX, Section 3, Interfaces Ethernet-Like Module";
     }


     /*
      * Configuration parameters for Ethernet-like interfaces.
      */
     augment "/if:interfaces/if:interface" {
       when "derived-from-or-self(if:type, 'ianaift:ethernetCsmacd') or
             derived-from-or-self(if:type, 'ianaift:ieee8023adLag') or
             derived-from-or-self(if:type, 'ianaift:ifPwType')" {
         description "Applies to all Ethernet-like interfaces";
       }
       description
         "Augment the interface model with parameters for all
          Ethernet-like interfaces.";

       container ethernet-like {
         description
           "Contains parameters for interfaces that use Ethernet framing
            and expose an Ethernet MAC layer.";

         leaf mac-address {
           if-feature "configurable-mac-address";
           type yang:mac-address;
           description
             "The MAC address of the interface.  The operational value
              matches the /if:interfaces/if:interface/if:phys-address
              leaf defined in ietf-interface.yang.";
         }

         leaf bia-mac-address {
           type yang:mac-address;
           config false;
           description
             "The 'burnt-in' MAC address.  I.e the default MAC address
```

```
            assigned to the interface if no MAC address has been
            explicitly configured on it.";
        }
      }
    }


    /*
     * Configuration parameters for Ethernet-like interfaces.
     */
    augment "/if:interfaces/if:interface/if:statistics" {
      when "derived-from-or-self(../if:type,
                                 'ianaift:ethernetCsmacd') or
            derived-from-or-self(../if:type,
                                 'ianaift:ieee8023adLag') or
            derived-from-or-self(../if:type, 'ianaift:ifPwType')" {
        description "Applies to all Ethernet-like interfaces";
      }
      description
        "Augment the interface model statistics with additional
         counters related to Ethernet-like interfaces.";

      leaf in-discard-unknown-dest-mac-pkts {
        type yang:counter64;
        units frames;
        description
          "A count of the number of frames that were well formed, but
           otherwise discarded because the destination MAC address did
           not pass any ingress destination MAC address filter.

           For consistency, frames counted against this counter are
           also counted against the IETF interfaces statistics.  In
           particular, they are included in in-octets and in-discards,
           but are not included in in-unicast-pkts, in-multicast-pkts
           or in-broadcast-pkts, because they are not delivered to a
           higher layer.

           Discontinuities in the values of this counter can occur at
           re-initialization of the management system, and at other
           times as indicated by the value of the 'discontinuity-time'
           leaf defined in the ietf-interfaces YANG module
           (RFC 8343).";
      }

      leaf in-discard-overflows {
        type yang:counter64;
        units frames;
        description
```

```
            "A count of the number of frames discarded because of
             overflows.";
        }
      }
    }
    <CODE ENDS>
```

6. Examples

The following sections give some examples of how different parts of the YANG modules could be used. Examples are not given for the more trivial configuration, or for sub-interfaces, for which examples are contained in [I-D.ietf-netmod-sub-intf-vlan-model].

6.1. Carrier delay configuration

The following example shows how the operational state datastore could look like for an Ethernet interface without any link flap suppression configuration. The down leaf value of 0 indicates that link down events as always propagated to high layers immediately, but an up leaf value of 50 indicates that the interface must be up and stable for at least 50 msecs before the interface is reported as being up to the high layers.

```
<?xml version="1.0" encoding="utf-8"?>
<interfaces
 xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <if-ext:link-flap-suppression>
      <if-ext:down>0</if-ext:down>
      <if-ext:up>50</if-ext:up>
    </if-ext:link-flap-suppression>
  </interface>
</interfaces>
```

The following example shows explicit link flap suppression delay up and down values have been configured. A 50 msec down leaf value has been used to potentially allow optical protection to recover the link before the higher layer protocol state is flapped. A 1 second (1000 milliseconds) up leaf value has been used to ensure that the link is always reasonably stable before allowing traffic to be carried over it. This also has the benefit of greatly reducing the rate at which higher layer protocol state flaps could occur.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
    xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <if-ext:link-flap-suppression>
        <if-ext:down>50</if-ext:down>
        <if-ext:up>1000</if-ext:up>
      </if-ext:link-flap-suppression>
    </interface>
  </interfaces>
</config>
```

6.2. Dampening configuration

The following example shows what the operational state datastore may look like for an interface configured with interface dampening. The 'suppressed' leaf indicates that the interface is currently suppressed (i.e. down) because the 'penalty' is greater than the 'suppress' leaf threshold. The 'time-remaining' leaf indicates that the interface will remain suppressed for another 103 seconds before the 'penalty' is below the 'reuse' leaf value and the interface is allowed to go back up again.

```
<?xml version="1.0" encoding="utf-8"?>
<interfaces
 xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <oper-status>down</oper-status>
    <dampening
     xmlns="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
      <half-life>60</half-life>
      <reuse>750</reuse>
      <suppress>2000</suppress>
      <max-suppress-time>240</max-suppress-time>
      <penalty>2480</penalty>
      <suppressed>true</suppressed>
      <time-remaining>103</time-remaining>
    </dampening>
  </interface>
</interfaces>
```

6.3. MAC address configuration

The following example shows how the operational state datastore could look like for an Ethernet interface without an explicit MAC address configured. The mac-address leaf always reports the actual operational MAC address that is in use. The bia-mac-address leaf always reports the default MAC address assigned to the hardware.

```
<?xml version="1.0" encoding="utf-8"?>
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <phys-address>00:00:5E:00:53:30</phys-address>
      <ethernet-like
        xmlns="urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like">
        <mac-address>00:00:5E:00:53:30</mac-address>
        <bia-mac-address>00:00:5E:00:53:30</bia-mac-address>
      </ethernet-like>
    </interface>
  </interfaces>
```

The following example shows the intended configuration for interface eth0 with an explicit MAC address configured.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
    xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <ethernet-like
        xmlns="urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like">
        <mac-address>00:00:5E:00:53:35</mac-address>
      </ethernet-like>
    </interface>
  </interfaces>
</config>
```

After the MAC address configuration has been successfully applied, the operational state datastore reporting the interface MAC address properties would contain the following, with the mac-address leaf updated to match the configured value, but the bia-mac-address leaf retaining the same value - which should never change.

```
<?xml version="1.0" encoding="utf-8"?>
<interfaces
 xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
      <phys-address>00:00:5E:00:53:35</phys-address>
    <ethernet-like
      xmlns="urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like">
      <mac-address>00:00:5E:00:53:35</mac-address>
      <bia-mac-address>00:00:5E:00:53:30</bia-mac-address>
    </ethernet-like>
  </interface>
</interfaces>
```

7. Acknowledgements

The authors wish to thank Eric Gray, Ing-Wher Chen, Jon Culver, Juergen Schoenwaelder, Ladislav Lhotka, Lou Berger, Mahesh Jethanandani, Martin Bjorklund, Michael Zitao, Neil Ketley, Qin Wu, William Lupton, Xufeng Liu, Andy Bierman, and Vladimir Vassilev for their helpful comments contributing to this document.

8. IANA Considerations

8.1. YANG Module Registrations

The following YANG modules are requested to be registered in the IANA "YANG Module Names" [RFC6020] registry:

The ietf-if-extensions module:

Name: ietf-if-extensions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-extensions

Prefix: if-ext

Reference: RFCXXXX

The ietf-if-ethernet-like module:

Name: ietf-if-ethernet-like

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like

Prefix: ethlike

Reference: RFCXXXX

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

URI: urn:ietf:params:xml:ns:yang:ietf-if-extensions

Registrant Contact: The IESG.

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

URI: urn:ietf:params:xml:ns:yang:ietf-if-ethernet-like

Registrant Contact: The IESG.

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

9. 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 8341 [RFC8341] 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:

9.1. ietf-if-extensions.yang

The ietf-if-extensions YANG module contains various configuration leaves that affect the behavior of interfaces. Modifying these leaves can cause an interface to go down, or become unreliable, or to drop traffic forwarded over it. More specific details of the possible failure modes are given below.

The following leaf could cause the interface to go down and stop processing any ingress or egress traffic on the interface. It could also cause broadcast traffic storms.

* /if:interfaces/if:interface/loopback

The following leaves could cause instabilities at the interface link layer, and cause unwanted higher layer routing path changes if the leaves are modified, although they would generally only affect a device that had some underlying link stability issues:

* /if:interfaces/if:interface/link-flap-suppression/down

* /if:interfaces/if:interface/link-flap-suppression/up

* /if:interfaces/if:interface/dampening/half-life

* /if:interfaces/if:interface/dampening/reuse

* /if:interfaces/if:interface/dampening/suppress

* /if:interfaces/if:interface/dampening/max-suppress-time

The following leaves could cause traffic loss on the interface because the received or transmitted frames do not comply with the frame matching criteria on the interface and hence would be dropped:

* /if:interfaces/if:interface/encapsulation

* /if:interfaces/if:interface/max-frame-size

* /if:interfaces/if:interface/forwarding-mode

Changing the parent-interface leaf could cause all traffic on the affected interface to be dropped. The affected leaf is:

* /if:interfaces/if:interface/parent-interface

9.2. ietf-if-ethernet-like.yang

Generally, the configuration nodes in the ietf-if-ethernet-like YANG module are concerned with configuration that is common across all types of Ethernet-like interfaces. The module currently only contains a node for configuring the operational MAC address to use on an interface. Adding/modifying/deleting this leaf has the potential risk of causing protocol instability, excessive protocol traffic, and general traffic loss, particularly if the configuration change caused a duplicate MAC address to be present on the local network. The following leaf is affected:

* interfaces/interface/ethernet-like/mac-address

10. References

10.1. 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, <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>.

[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>.

[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>.

[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>.

[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>.

10.2. Informative References

[I-D.ietf-netmod-sub-intf-vlan-model]
Wilton, R. and S. Mansfield, "Sub-interface VLAN YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-sub-intf-vlan-model-08, 26 January 2023, <https://www.ietf.org/archive/id/draft-ietf-netmod-sub-intf-vlan-model-08.txt>.

[IEEE_802.3.2_2019]
IEEE, "IEEE Standard for Ethernet - YANG Data Model Definitions", IEEE 802-3, DOI 10.1109/IEEESTD.2019.8737019, 14 June 2019, <https://ieeexplore.ieee.org/document/8737019>.

[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>.

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

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <https://www.rfc-editor.org/info/rfc7224>.

[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>.

[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>.

Authors' Addresses

Robert Wilton
Cisco Systems
Email: rwilton@cisco.com

Scott Mansfield
Ericsson
Email: scott.mansfield@ericsson.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 20 April 2024

R.G. Wilton, Ed.
Cisco Systems
S. Mansfield, Ed.
Ericsson

18 October 2023

Sub-interface VLAN YANG Data Models
draft-ietf-netmod-sub-intf-vlan-model-09

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 configuration of Layer 3 and Layer 2 sub-interfaces (e.g. L2VPN attachment circuits) that can interoperate with IETF based forwarding protocols; such as IP and L3VPN services; or L2VPN services like VPWS, VPLS, and EVPN. The sub-interfaces also interoperate with VLAN tagged traffic orignating from an IEEE 802.1Q compliant bridge.

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.

The YANG data models in this document conforms to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.

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 20 April 2024.

Table of Contents

1. Introduction

This document defines two YANG [RFC7950] 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 [RFC8343]. 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, for example, IPv6 [RFC8200], Ethernet Pseudo Wires [RFC4448] and VPLS [RFC4761] [RFC4762], when configured via appropriate YANG data models [RFC8344] [I-D.ietf-bess-l2vpn-yang], to interoperate 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 in the 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:

ietf-if-vlan-encapsulation.yang - Defines the model for basic classification of VLAN tagged traffic, normally to L3 packet forwarding services

ietf-if-flexible-encapsulation.yang - Defines the model for flexible classification of Ethernet/VLAN traffic, normally to L2 frame forwarding services

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.

The term 'sub-interface' is defined in section 2.6 of Interface Extensions YANG [I-D.ietf-netmod-intf-ext-yang].

1.2. Tree Diagrams

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

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 compliant 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.

3. Interface VLAN Encapsulation Model

The Interface VLAN encapsulation model provides appropriate leaves for termination of an 802.1Q VLAN tagged segment to a sub-interface (or interface) based L3 service, such as IP. It allows for termination of traffic with one or two 802.1Q VLAN tags.

The L3 service must be configured via a separate YANG data model, e.g., [RFC8344]. A short example of configuring 802.1Q VLAN sub-interfaces with IP using YANG is provided in Section 7.1.

The "ietf-if-vlan-encapsulation" YANG module has the following structure:

```
module: ietf-if-vlan-encapsulation
  augment /if:interfaces/if:interface/if-ext:encapsulation
            /if-ext:encaps-type:
    +--:(dot1q-vlan)
       +--rw dot1q-vlan
          +--rw outer-tag
          |  +--rw tag-type dot1q-tag-type
          |  +--rw vlan-id     vlanid
          +--rw second-tag!
             +--rw tag-type dot1q-tag-type
             +--rw vlan-id     vlanid
```

4. Interface Flexible Encapsulation Model

The Interface Flexible Encapsulation model is designed to allow for the flexible provisioning of layer 2 services. It provides the capability to classify and demultiplex 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 frame header before the frame is handed off to the appropriate forwarding code for further handling. The forwarding instance, e.g., L2VPN, VPLS, etc., is configured using a separate YANG configuration model defined elsewhere, e.g., [I-D.ietf-bess-l2vpn-yang].

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 model also allows a flexible encapsulation and rewrite to be configured directly on an Ethernet or LAG interface without configuring separate child sub-interfaces. Ingress frames that do not match the encapsulation are dropped. Egress frames MUST conform to the encapsulation.

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.

A short example of configuring 802.1Q VLAN sub-interfaces with L2VPN using YANG is provided in Section 7.2.

The "ietf-if-flexible-encapsulation" YANG module has the following structure:

```
module: ietf-if-flexible-encapsulation
  augment /if:interfaces/if:interface/if-ext:encapsulation
            /if-ext:encaps-type:
    +--:(flexible)
       +--rw flexible
          +--rw match
          |  +--rw (match-type)
          |     +--:(default)
          |     |  +--rw default?                empty
          |     +--:(untagged)
          |     |  +--rw untagged?               empty
          |     +--:(dot1q-priority-tagged)
          |     |  +--rw dot1q-priority-tagged
          |     |     +--rw tag-type dot1q-types:dot1q-tag-type
          |     +--:(dot1q-vlan-tagged)
          |        +--rw dot1q-vlan-tagged
          |           +--rw outer-tag
          |           |  +--rw tag-type dot1q-tag-type
          |           |  +--rw vlan-id     union
          |           +--rw second-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 dot1q-tag-rewrite {dot1q-tag-rewrites}?
          |     |        +--rw pop-tags?    uint8
          |     |        +--rw push-tags!
          |     |           +--rw outer-tag
          |     |           |  +--rw tag-type dot1q-tag-type
          |     |           |  +--rw vlan-id     vlanid
          |     |           +--rw second-tag!
```

```
|      |              +--rw tag-type dot1q-tag-type
|      |              +--rw vlan-id     vlanid
|      +--:(asymmetrical) {asymmetric-rewrites}?
|         +--rw ingress
|         |  +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|         |     +--rw pop-tags?    uint8
|         |     +--rw push-tags!
|         |        +--rw outer-tag
|         |        |  +--rw tag-type dot1q-tag-type
|         |        |  +--rw vlan-id     vlanid
|         |        +--rw second-tag!
|         |           +--rw tag-type dot1q-tag-type
|         |           +--rw vlan-id     vlanid
|         +--rw egress
|            +--rw dot1q-tag-rewrite {dot1q-tag-rewrites}?
|               +--rw pop-tags?    uint8
|               +--rw push-tags!
|                  +--rw outer-tag
|                  |  +--rw tag-type dot1q-tag-type
|                  |  +--rw vlan-id     vlanid
|                  +--rw second-tag!
|                     +--rw tag-type dot1q-tag-type
|                     +--rw vlan-id     vlanid
|
+--rw local-traffic-default-encaps!
   +--rw outer-tag
   |  +--rw tag-type dot1q-tag-type
   |  +--rw vlan-id     vlanid
   +--rw second-tag!
      +--rw tag-type dot1q-tag-type
      +--rw vlan-id     vlanid
```

5. VLAN Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in ietf-if-extensions.yang [I-D.ietf-netmod-intf-ext-yang]. It also contains references to [RFC8343], [RFC7224], and [IEEE_802.1Q_2022].

<CODE BEGINS> file "ietf-if-vlan-encapsulation@2023-01-26.yang"

```
module ietf-if-vlan-encapsulation {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation";
  prefix if-vlan;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
```

```
  }

  import iana-if-type {
    prefix ianaift;
    reference
      "RFC 7224: IANA Interface Type YANG Module";
  }

  import ieee802-dot1q-types {
    prefix dot1q-types;
    revision-date 2022-01-19;
    reference
      "IEEE Std 802.1Q-2022: IEEE Standard for Local and
       metropolitan area networks -- Bridges and Bridged Networks";
  }

  import ietf-if-extensions {
    prefix if-ext;
    reference
      "RFC XXXX: Common Interface Extension YANG Data Models";
  }

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

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

     Editor:   Robert Wilton
               <mailto:rwilton@cisco.com>";

  description
    "This YANG module models configuration to classify IEEE 802.1Q
     VLAN tagged Ethernet traffic by exactly matching the tag type
     and VLAN identifier of one or two 802.1Q VLAN tags in the frame.

     Copyright (c) 2023 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 Revised 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).

     This version of this YANG module is part of RFC XXXX
```

```
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
        for full legal notices.

        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 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

    revision 2023-01-26 {
      description
        "Latest draft revision";
      reference
        "RFC XXXX: Sub-interface VLAN YANG Data Models";
    }

    augment "/if:interfaces/if:interface/if-ext:encapsulation/"
          + "if-ext:encaps-type" {
      when "derived-from-or-self(../if:type,
                                 'ianaift:ethernetCsmacd') or
            derived-from-or-self(../if:type,
                                 'ianaift:ieee8023adLag') or
            derived-from-or-self(../if:type, 'ianaift:l2vlan') or
            derived-from-or-self(../if:type,
                                 'if-ext:ethSubInterface')" {
        description
          "Applies only to Ethernet-like interfaces and
           sub-interfaces.";
      }

      description
        "Augment the generic interface encapsulation with basic 802.1Q
         VLAN tag classifications";

      case dot1q-vlan {
        container dot1q-vlan {

          description
            "Classifies 802.1Q VLAN tagged Ethernet frames to a
             sub-interface (or interface) by exactly matching the
             number of tags, tag type(s) and VLAN identifier(s).

             Only frames matching the classification configured on a
             sub-interface/interface are processed on that
             sub-interface/interface.

             Frames that do not match any sub-interface are processed
             directly on the parent interface, if it is associated with
```

```
            a forwarding instance, otherwise they are dropped.";

          container outer-tag {
            must 'tag-type = "dot1q-types:s-vlan" or '
               + 'tag-type = "dot1q-types:c-vlan"' {

              error-message
                "Only C-VLAN and S-VLAN tags can be matched.";

              description
                "For IEEE 802.1Q interoperability, only C-VLAN and
                 S-VLAN tags are matched.";
            }

            description
              "Specifies the VLAN tag values to match against the
               outermost (first) 802.1Q VLAN tag in the frame.";

            uses dot1q-types:dot1q-tag-classifier-grouping;
          }

          container second-tag {
            must '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
               + 'tag-type = "dot1q-types:c-vlan"' {

              error-message
                "When matching two 802.1Q VLAN tags, the outermost
                 (first) tag in the frame MUST be specified and be of
                 S-VLAN type and the second tag in the frame must be of
                 C-VLAN tag type.";

              description
                "For IEEE 802.1Q interoperability, when matching two
                 802.1Q VLAN tags, it is REQUIRED that the outermost
                 tag exists and is an S-VLAN, and the second tag is a
                 C-VLAN.";
            }

            presence "Classify frames that have two 802.1Q VLAN tags.";

            description
              "Specifies the VLAN tag values to match against the
               second outermost 802.1Q VLAN tag in the frame.";

            uses dot1q-types:dot1q-tag-classifier-grouping;
          }
        }
      }
```

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

6. Flexible Encapsulation YANG Module

This YANG module augments the 'encapsulation' container defined in ietf-if-extensions.yang [I-D.ietf-netmod-intf-ext-yang]. This YANG module also augments the 'interface' list entry defined in [RFC8343]. It also contains references to [RFC7224], and [IEEE_802.1Q_2022].

```
<CODE BEGINS> file "ietf-if-flexible-encapsulation@2023-01-26.yang"
module ietf-if-flexible-encapsulation {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation";
  prefix if-flex;

  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model For Interface Management";
  }

  import iana-if-type {
    prefix ianaift;
    reference
      "RFC 7224: IANA Interface Type YANG Module";
  }

  import ieee802-dot1q-types {
    prefix dot1q-types;
    revision-date 2022-01-19;
    reference
      "IEEE Std 802.1Q-2022: IEEE Standard for Local and
       metropolitan area networks -- Bridges and Bridged Networks";
  }

  import ietf-if-extensions {
    prefix if-ext;
    reference
      "RFC XXXX: Common Interface Extension YANG Data Models";
  }

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

  contact
```

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

        Editor:   Robert Wilton
                  <mailto:rwilton@cisco.com>";

     description
       "This YANG module describes interface configuration for flexible
        classification and rewrites of IEEE 802.1Q VLAN tagged Ethernet
        traffic.

        Copyright (c) 2022 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 Revised 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).

        This version of this YANG module is part of RFC XXXX
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
        for full legal notices.

        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 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

     revision 2023-01-26 {
       description
         "Latest draft revision";
       reference
         "RFC XXXX: Sub-interface VLAN YANG Data Models";
     }

     feature flexible-rewrites {
       description
         "This feature indicates that the network element supports
           specifying flexible rewrite operations.";
     }

     feature asymmetric-rewrites {
       description
         "This feature indicates that the network element supports
          specifying different rewrite operations for the ingress
```

```
          rewrite operation and egress rewrite operation.";
      }

      feature dot1q-tag-rewrites {
        description
          "This feature indicates that the network element supports the
           flexible rewrite functionality specifying 802.1Q tag
           rewrites.";
      }

      grouping flexible-match {
        description
          "Represents a flexible frame classification:

           The rules for a flexible match are:
             1. Match-type: default, untagged, priority tag, or tag
                stack.
             2. Each tag in the stack of tags matches:
              a. tag type (802.1Q or 802.1ad) +
              b. tag value:
                i.   single tag
                ii.  set of tag ranges/values.
                iii. 'any' keyword";

        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.";
            }
          }

          case untagged {
            description
              "Match untagged Ethernet frames only";
```

```
          leaf untagged {
            type empty;

            description
              "Untagged match.  Matches all untagged traffic.";
          }
        }

        case dot1q-priority-tagged {
          description
            "Match 802.1Q priority tagged Ethernet frames only";

          container dot1q-priority-tagged {
            description
              "802.1Q priority tag match";

            leaf tag-type {
              type dot1q-types:dot1q-tag-type;
              mandatory true;

              description
                "The 802.1Q tag type of matched priority
                 tagged packets";
            }
          }
        }

        case dot1q-vlan-tagged {
          container dot1q-vlan-tagged {
            description
              "Matches VLAN tagged frames";

            container outer-tag {
              must 'tag-type = "dot1q-types:s-vlan" or '
                 + 'tag-type = "dot1q-types:c-vlan"' {

                error-message
                  "Only C-VLAN and S-VLAN tags can be matched.";

                description
                  "For IEEE 802.1Q interoperability, only C-VLAN and
                   S-VLAN tags can be matched.";
              }

              description
                "Classifies traffic using the outermost (first) VLAN
                 tag on the frame.";
```

```
            uses "dot1q-types:"
               + "dot1q-tag-ranges-or-any-classifier-grouping";
          }

          container second-tag {
            must
              '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
            + 'tag-type = "dot1q-types:c-vlan"' {

              error-message
                "When matching two tags, the outermost (first) tag
                 must be specified and of S-VLAN type and the second
                 outermost tag must be of C-VLAN tag type.";

              description
                "For IEEE 802.1Q interoperability, when matching two
                 tags, it is required that the outermost (first) tag
                 exists and is an S-VLAN, and the second outermost
                 tag is a C-VLAN.";
            }

            presence "Also classify on the second VLAN tag.";

            description
              "Classifies traffic using the second outermost VLAN tag
               on the frame.";

            uses "dot1q-types:"
               + "dot1q-tag-ranges-or-any-classifier-grouping";
          }

          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.";
          }
        }
      }
    }
  }

  grouping dot1q-tag-rewrite {
    description
      "Flexible rewrite grouping.  Can be either be expressed
```

```
          symmetrically, or independently in the ingress and/or egress
          directions.";

        leaf pop-tags {
          type uint8 {
            range "1..2";
          }

          description
            "The number of 802.1Q VLAN tags to pop, or translate if used
             in conjunction with push-tags.

             Popped tags are the outermost tags on the frame.";
        }

        container push-tags {
          presence "802.1Q tags are pushed or translated";

          description
            "The 802.1Q tags to push on the front of the frame, or
             translate if configured in conjunction with pop-tags.";

          container outer-tag {
            must 'tag-type = "dot1q-types:s-vlan" or '
               + 'tag-type = "dot1q-types:c-vlan"' {

              error-message "Only C-VLAN and S-VLAN tags can be pushed.";

              description
                "For IEEE 802.1Q interoperability, only C-VLAN and S-VLAN
                 tags can be pushed.";
            }

            description
              "The outermost (first) VLAN tag to push onto the frame.";

            uses dot1q-types:dot1q-tag-classifier-grouping;
          }

          container second-tag {
            must '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
               + 'tag-type = "dot1q-types:c-vlan"' {

              error-message
                "When pushing/rewriting two tags, the outermost tag must
                 be specified and of S-VLAN type and the second outermost
                 tag must be of C-VLAN tag type.";
```

```
            description
              "For IEEE 802.1Q interoperability, when pushing two tags,
               it is required that the outermost tag exists and is an
               S-VLAN, and the second outermost tag is a C-VLAN.";
          }

          presence
            "In addition to the first tag, also push/rewrite a second
             VLAN tag.";

          description
            "The second outermost VLAN tag to push onto the frame.";

          uses dot1q-types:dot1q-tag-classifier-grouping;
        }
      }
    }

    grouping flexible-rewrite {
      description
        "Grouping for flexible rewrites of fields in the L2 header.

         Restricted to flexible 802.1Q VLAN tag rewrites, but could be
         extended to cover rewrites of other fields in the L2 header in
         future.";

      container dot1q-tag-rewrite {
        if-feature "dot1q-tag-rewrites";

        description
          "802.1Q VLAN tag rewrite.

           Translate operations are expressed as a combination of tag
           push and pop operations.  E.g., translating the outer tag is
           expressed as popping a single tag, and pushing a single tag.
           802.1Q tags that are translated SHOULD preserve the PCP and
           DEI fields unless if a different QoS behavior has been
           specified.";
        uses dot1q-tag-rewrite;
      }
    }

    augment "/if:interfaces/if:interface/if-ext:encapsulation/"
          + "if-ext:encaps-type" {
      when "derived-from-or-self(../if:type,
                                 'ianaift:ethernetCsmacd') or
            derived-from-or-self(../if:type,
                                 'ianaift:ieee8023adLag') or
```

```
            derived-from-or-self(../if:type, 'ianaift:l2vlan') or
            derived-from-or-self(../if:type,
                                 'if-ext:ethSubInterface')" {

          description
            "Applies only to Ethernet-like interfaces and
             sub-interfaces.";
        }

        description
          "Augment the generic interface encapsulation with flexible
           match and rewrite for VLAN sub-interfaces.";

        case flexible {
          description
            "Flexible encapsulation and rewrite";

          container flexible {
            description
              "Flexible encapsulation allows for the matching of ranges
               and sets of 802.1Q VLAN Tags and performing rewrite
               operations on the VLAN tags.

               The structure is also designed to be extended to allow for
               matching/rewriting other fields within the L2 frame header
               if required.";

            container match {
              description
                "Flexibly classifies Ethernet frames to a sub-interface
                 (or interface) based on the L2 header fields.

                 Only frames matching the classification configured on a
                 sub-interface/interface are processed on that
                 sub-interface/interface.

                 Frames that do not match any sub-interface are processed
                 directly on the parent interface, if it is associated
                 with a forwarding instance, otherwise they are dropped.

                 If a frame could be classified to multiple
                 sub-interfaces then they get classified to the
                 sub-interface with the most specific match.  E.g.,
                 matching two VLAN tags in the frame is more specific
                 than matching the outermost VLAN tag, which is more
                 specific than the catch all 'default' match.";

              uses flexible-match;
```

```
          }

          container rewrite {
            if-feature "flexible-rewrites";

            description
              "L2 frame rewrite operations.

               Rewrites allows for modifications to the L2 frame header
               as it transits the interface/sub-interface.  Examples
               include adding a VLAN tag, removing a VLAN tag, or
               rewriting the VLAN Id carried in a VLAN tag.";

            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.

                     E.g., if a tag is pushed on ingress traffic, then
                     the reverse operation is a 'pop 1', that is
                     performed on traffic egressing the interface, so
                     a peer device sees a consistent L2 encapsulation
                     for both ingress and egress traffic.";
                }
              }

              case asymmetrical {
                if-feature "asymmetric-rewrites";

                description
                  "Asymmetrical rewrite.

                   Rewrite operations may be specified in only a single
                   direction, or different rewrite operations may be
                   specified in each direction.";

                container ingress {
                  uses flexible-rewrite;
```

```
                description
                  "A rewrite operation that only applies to ingress
                   traffic.

                   Ingress rewrite operations are performed before
                   the frame is subsequently processed by the
                   forwarding operation.";
                }

                container egress {
                  uses flexible-rewrite;

                  description
                    "A rewrite operation that only applies to egress
                     traffic.";
                }
              }
            }
          }

          container local-traffic-default-encaps {
            presence "A local traffic default encapsulation has been
                      specified.";

            description
              "Specifies the 802.1Q VLAN tags to use by default for
               locally sourced traffic from the interface.

               Used for encapsulations that match a range of VLANs (or
               'any'), where the source VLAN Ids are otherwise
               ambiguous.";

            container outer-tag {
              must 'tag-type = "dot1q-types:s-vlan" or '
                 + 'tag-type = "dot1q-types:c-vlan"' {

                error-message
                  "Only C-VLAN and S-VLAN tags can be matched.";

                description
                  "For IEEE 802.1Q interoperability, only C-VLAN and
                   S-VLAN tags can be matched.";
              }

              description
                "The outermost (first) VLAN tag for locally sourced
                 traffic.";
```

```
              uses dot1q-types:dot1q-tag-classifier-grouping;
            }

            container second-tag {
              must
                '../outer-tag/tag-type = "dot1q-types:s-vlan" and '
              + 'tag-type = "dot1q-types:c-vlan"' {

                error-message
                  "When specifying two tags, the outermost (first) tag
                   must be specified and of S-VLAN type and the second
                   outermost tag must be of C-VLAN tag type.";

                description
                  "For IEEE 802.1Q interoperability, when specifying
                   two tags, it is required that the outermost (first)
                   tag exists and is an S-VLAN, and the second
                   outermost tag is a C-VLAN.";
              }

              presence
                "Indicates existence of a second outermost VLAN tag.";

              description
                "The second outermost VLAN tag for locally sourced
                 traffic.";

              uses dot1q-types:dot1q-tag-classifier-grouping;
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

7. Examples

The following sections give examples of configuring a sub-interface supporting L3 forwarding, and a sub-interface being used in conjunction with the IETF L2VPN YANG model [I-D.ietf-bess-l2vpn-yang].

7.1. Layer 3 sub-interfaces with IPv6

This example illustrates two layer sub-interfaces, 'eth0.1' and 'eth0.2', both are child interfaces of the Ethernet interface 'eth0'.

'eth0.1' is configured to match traffic with two VLAN tags: an outer S-VLAN of 10 and an inner C-VLAN of 20.

'eth0.2' is configured to match traffic with a single S-VLAN tag, with VLAN Id 11.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
      <name>eth0.1</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
      <if-ext:encapsulation>
        <dot1q-vlan
         xmlns=
            "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
          <outer-tag>
            <tag-type>dot1q-types:s-vlan</tag-type>
            <vlan-id>10</vlan-id>
          </outer-tag>
          <second-tag>
            <tag-type>dot1q-types:c-vlan</tag-type>
            <vlan-id>20</vlan-id>
          </second-tag>
        </dot1q-vlan>
      </if-ext:encapsulation>
      <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
        <forwarding>true</forwarding>
        <address>
          <ip>2001:db8:10::1</ip>
          <prefix-length>48</prefix-length>
        </address>
      </ipv6>
    </interface>
    <interface>
      <name>eth0.2</name>
      <type>ianaift:l2vlan</type>
      <if-ext:parent-interface>eth0</if-ext:parent-interface>
```

```
        <if-ext:encapsulation>
          <dot1q-vlan
           xmlns=
              "urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation">
            <outer-tag>
              <tag-type>dot1q-types:s-vlan</tag-type>
              <vlan-id>11</vlan-id>
            </outer-tag>
          </dot1q-vlan>
        </if-ext:encapsulation>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <forwarding>true</forwarding>
          <address>
            <ip>2001:db8:11::1</ip>
            <prefix-length>48</prefix-length>
          </address>
        </ipv6>
      </interface>
    </interfaces>
  </config>
```

7.2. Layer 2 sub-interfaces with L2VPN

This example illustrates a layer 2 sub-interface 'eth0.3' configured to match traffic with a S-VLAN tag of 10, and C-VLAN tag of 21; and remov the outer tag (S-VLAN 10) before the traffic is passed off to the L2VPN service.

It also illustrates another sub-interface 'eth1.0' under a separate physical interface configured to match traffic with a C-VLAN of 50, with the tag removed before traffic is given to any service. Sub-interface 'eth1.0' is not currently bound to any service and hence traffic classified to that sub-interface is dropped.

```
<?xml version="1.0" encoding="utf-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  xmlns:dot1q-types="urn:ieee:std:802.1Q:yang:ieee802-dot1q-types"
  xmlns:if-ext="urn:ietf:params:xml:ns:yang:ietf-if-extensions">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
    </interface>
    <interface>
```

```
        <name>eth0.3</name>
        <type>ianaift:l2vlan</type>
        <if-ext:parent-interface>eth0</if-ext:parent-interface>
        <if-ext:encapsulation>
          <flexible xmlns=
          "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation">
            <match>
              <dot1q-vlan-tagged>
                <outer-tag>
                  <tag-type>dot1q-types:s-vlan</tag-type>
                  <vlan-id>10</vlan-id>
                </outer-tag>
                <second-tag>
                  <tag-type>dot1q-types:c-vlan</tag-type>
                  <vlan-id>21</vlan-id>
                </second-tag>
              </dot1q-vlan-tagged>
            </match>
            <rewrite>
              <symmetrical>
                <dot1q-tag-rewrite>
                  <pop-tags>1</pop-tags>
                </dot1q-tag-rewrite>
              </symmetrical>
            </rewrite>
          </flexible>
        </if-ext:encapsulation>
      </interface>
      <interface>
        <name>eth1</name>
        <type>ianaift:ethernetCsmacd</type>
      </interface>
      <interface>
        <name>eth1.0</name>
        <type>ianaift:l2vlan</type>
        <if-ext:parent-interface>eth0</if-ext:parent-interface>
        <if-ext:encapsulation>
          <flexible xmlns=
          "urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation">
            <match>
              <dot1q-vlan-tagged>
                <outer-tag>
                  <tag-type>dot1q-types:c-vlan</tag-type>
                  <vlan-id>50</vlan-id>
                </outer-tag>
              </dot1q-vlan-tagged>
            </match>
            <rewrite>
```

```
              <symmetrical>
                <dot1q-tag-rewrite>
                  <pop-tags>1</pop-tags>
                </dot1q-tag-rewrite>
              </symmetrical>
            </rewrite>
          </flexible>
        </if-ext:encapsulation>
      </interface>
    </interfaces>
    <network-instances
        xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
      <network-instance
       xmlns:l2vpn="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
        <name>p2p-l2-1</name>
        <description>Point to point L2 service</description>
        <l2vpn:type>l2vpn:vpws-instance-type</l2vpn:type>
        <l2vpn:signaling-type>
          l2vpn:ldp-signaling
        </l2vpn:signaling-type>
        <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
          <name>local</name>
          <ac>
            <name>eth0.3</name>
          </ac>
        </endpoint>
        <endpoint xmlns="urn:ietf:params:xml:ns:yang:ietf-l2vpn">
          <name>remote</name>
          <pw>
            <name>pw1</name>
          </pw>
        </endpoint>
        <vsi-root>
        <!-- Does not Validate -->
        </vsi-root>
      </network-instance>
    </network-instances>
    <pseudowires
        xmlns="urn:ietf:params:xml:ns:yang:ietf-pseudowires">
      <pseudowire>
        <name>pw1</name>
          <peer-ip>2001:db8::50></peer-ip>
          <pw-id>100</pw-id>
      </pseudowire>
    </pseudowires>
  </config>
```

8. Acknowledgements

The authors would particularly like to thank Benoit Claise, John Messenger, Glenn Parsons, and Dan Romascanu for their help progressing this draft.

The authors would also like to thank Martin Bjorklund, Alex Campbell, Don Fedyk, Eric Gray, Giles Heron, Marc Holness, Iftekhar Hussain, Neil Ketley, William Lupton, John Messenger, Glenn Parsons, Ludwig Pauwels, Joseph White, Vladimir Vassilev, and members of the IEEE 802.1 WG for their helpful reviews and feedback on this draft.

9. IANA Considerations

9.1. YANG Module Registrations

The following YANG modules are requested to be registered in the IANA "YANG Module Names" [RFC6020] registry:

The ietf-if-vlan-encapsulation module:

Name: ietf-if-vlan-encapsulation

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation

Prefix: if-vlan

Reference: RFCXXXX

The ietf-if-flexible-encapsulation module:

Name: ietf-if-flexible-encapsulation

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation

Prefix: if-flex

Reference: RFCXXXX

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

URI: urn:ietf:params:xml:ns:yang:ietf-if-vlan-encapsulation

Registrant Contact: The IESG.

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

URI: urn:ietf:params:xml:ns:yang:ietf-if-flexible-encapsulation

Registrant Contact: The IESG.

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

10. 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 [RFC8341] 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. ietf-if-vlan-encapsulation.yang

The nodes in the vlan encapsulation 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/dot1q-vlan, that are sensitive to this are:

* outer-tag/tag-type
* outer-tag/vlan-id
* second-tag/tag-type
* second-tag/vlan-id

10.2. ietf-if-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
* dot1q-priority-tagged
* dot1q-priority-tagged/tag-type
* dot1q-vlan-tagged/outer-tag/vlan-type
* dot1q-vlan-tagged/outer-tag/vlan-id
* dot1q-vlan-tagged/second-tag/vlan-type
* dot1q-vlan-tagged/second-tag/vlan-id

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/dot1q-tag-rewrite/pop-tags
* symmetrical/dot1q-tag-rewrite/push-tags/outer-tag/tag-type
* symmetrical/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id
* symmetrical/dot1q-tag-rewrite/push-tags/second-tag/tag-type
* symmetrical/dot1q-tag-rewrite/push-tags/second-tag/vlan-id
* asymmetrical/ingress/dot1q-tag-rewrite/pop-tags

* asymmetrical/ingress/dot1q-tag-rewrite/push-tags/outer-tag/tag-type

* asymmetrical/ingress/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id

* asymmetrical/ingress/dot1q-tag-rewrite/push-tags/second-tag/tag-type

* asymmetrical/ingress/dot1q-tag-rewrite/push-tags/second-tag/vlan-id

* asymmetrical/egress/dot1q-tag-rewrite/pop-tags

* asymmetrical/egress/dot1q-tag-rewrite/push-tags/outer-tag/tag-type

* asymmetrical/egress/dot1q-tag-rewrite/push-tags/outer-tag/vlan-id

* asymmetrical/egress/dot1q-tag-rewrite/push-tags/second-tag/tag-type

* asymmetrical/egress/dot1q-tag-rewrite/push-tags/second-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:

* outer-tag/vlan-type

* outer-tag/vlan-id

* second-tag/vlan-type

* second-tag/vlan-id

11. References

11.1. Normative References

[I-D.ietf-netmod-intf-ext-yang]
Wilton, R. and S. Mansfield, "Common Interface Extension YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-intf-ext-yang-11, 26 January 2023, <https://www.ietf.org/archive/id/draft-ietf-netmod-intf-ext-yang-11.txt>.

[IEEE_802.1Q_2022]
IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks", IEEE 802-1q-2022, DOI 10.1109/IEEESTD.2022.10004498, 30 December 2022, <https://ieeexplore.ieee.org/document/10004498>.

[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>.

[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>.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <https://www.rfc-editor.org/info/rfc7224>.

[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>.

[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>.

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

11.2. Informative References

[I-D.ietf-bess-l2vpn-yang]
Shah, H. C., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", Work in Progress, Internet-Draft, draft-ietf-bess-l2vpn-yang-10, 2 July 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-bess-l2vpn-yang-10>.

[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, <https://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, <https://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, <https://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, <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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[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>.

[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>.

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 has developed 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

Scott Mansfield (editor)
Ericsson
Email: scott.mansfield@ericsson.com

NETMOD
Internet-Draft
Updates: 8342, 6241, 8526, 8040 (if approved)
Intended status: Standards Track
Expires: 21 April 2024

Q. Ma, Ed.
Q. Wu
C. Feng
Huawei

19 October 2023

System-defined Configuration
draft-ietf-netmod-system-config-03

Abstract

This document describes how a management client and server handle YANG-modeled configuration data that is defined by the server itself. The system-defined configuration can be referenced (e.g. leafref) by configuration explicitly created by a client.

The Network Management Datastore Architecture (NMDA) defined in RFC 8342 is updated with a read-only conventional configuration datastore called "system" to hold system-defined configuration. As an alternative to clients explicitly copying referenced system-defined configuration into the target configuration datastore (e.g., <running>) so that the datastore is valid, a "resolve-system" parameter is defined to allow the server acting as a "system client" to copy referenced system-defined nodes automatically. This solution enables clients manipulating the target configuration datastore (e.g., <running>) to overlay (e.g., copy system configuration using the same key value as in <system>) and reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

This document updates RFC 8342, RFC 6241, RFC 8526 and RFC 8040.

This Internet-Draft will expire on 21 April 2024.

Table of Contents

1. Introduction

The Network Management Datastore Architecture (NMDA) [RFC8342] defines system configuration as the configuration that is supplied by the device itself and appears in <operational> when it is in use (Figure 2 in [RFC8342]).

However, there is a desire to enable a server to better structure and expose the system configuration. NETCONF/RESTCONF clients can benefit from a standard mechanism to retrieve what system configuration is available on a server.

Some servers allow the NETCONF/RESTCONF client to reference a system-defined node which isn't present in the target datastore (e.g., <running>). The absence of the system configuration in the datastore can render the datastore invalid from the perspective of a client or offline tools (e.g., missing leafref targets). This document describes several approaches to bring the datastore to a valid state and ensuring that all referential integrity constraints are satisfied.

Some servers allow the descendant nodes of system-defined configuration to be configured or modified. For example, the system configuration may contain an almost empty physical interface, while the client needs to be able to add, modify, or remove a number of descendant nodes. Some descendant nodes may not be modifiable (e.g., the interface "name" and "type" set by the system).

This document updates the Network Management Datastore Architecture (NMDA) defined in RFC 8342 with a read-only conventional configuration datastore called "system" to hold system-defined configuration. As an alternative to clients explicitly copying referenced system-defined configuration into the target configuration datastore (e.g., <running>) so that the datastore is valid, a "resolve-system" parameter is defined to allow the server acting as a "system client" to copy referenced system-defined nodes automatically. This solution enables clients manipulating the target configuration datastore (e.g., <running>) to overlay (e.g., copy system configuration using the same key value as in <system>) and reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

If a system-defined node is referenced, it refers to one of the following cases throughout this document:

* It is present in a leafref "path" statement and referred as the leafref value

* It is used as an "instance-identifier" type value

* It is present in an Xpath expression of "when" or "must" constraints

* It is defined to satisfy the "mandatory" constraints

* It is defined to exactly satisfy the "min-element" constraints

Conformance to this document requires the NMDA servers to implement the "ietf-system-datastore" YANG module (Section 6).

1.1. Terminology

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8342], [RFC8407], and [RFC8525] and uses terminologies from those documents.

The following terms are defined in this document:

System configuration: Configuration that is provided by the system itself. System configuration is present in the system configuration datastore (regardless of being applied by the device or referenced by other configuration nodes), and appears in the intended configuration datastore. System configuration that is considered active (according to the NMDA defined in RFC 8342) appears in <operational> with origin="system". It is a different and separate concept from factory default configuration defined in RFC 8808 (which represents a preset initial configuration that is used to initialize the configuration of a server).

System configuration datastore: A configuration datastore holding configuration provided by the system itself. This datastore is referred to as "<system>".

This document redefines the term "conventional configuration datastore" in Section 3 of [RFC8342] to add "system" to the list of conventional configuration datastores:

Conventional configuration datastore: One of the following set of configuration datastores: <running>, <startup>, <candidate>, <system>, and <intended>. These datastores share a common datastore schema, and protocol operations allow copying data between these datastores. The term "conventional" is chosen as a generic umbrella term for these datastores.

1.2. 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.

1.3. Updates to RFC 8342

This document updates RFC 8342 to define a configuration datastore called "system" to hold system configuration, it also redefines the term "conventional configuration datastore" from RFC 8342 to add "system" to the list of conventional configuration datastores. The contents of <system> are read-only to clients but may change dynamically itself. <system> aware client may retrieve all three types of system configuration defined in Section 2, reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes.

Configuration in <system> is merged with the current configuration of the device in <running> after the configuration transformations (e.g., template expansion, removal of inactive configuration defined in [RFC8342]) to create the contents of <intended>. As always, system configuration will appear in <operational> with origin="system" when it is in use.

The system datastore makes system configuration visible to clients in order for being referenced or configurable prior to present in <operational>.

1.4. Updates to RFC 6241 and RFC 8526

This document augments <edit-config> and <edit-data> RPC operations defined in [RFC6241] and [RFC8526] respectively, with a new additional input parameter "resolve-system". The <copy-config> RPC operation defined in [RFC6241] is also augmented to support "resolve-system" parameter.

The "resolve-system" parameter is optional and has no value. When it is provided and the server detects that there is a reference to a system-defined node during the validation, the server will automatically copy the referenced system configuration into the validated datastore to make the configuration valid without the client doing so explicitly. Legacy clients interacting with servers that support this parameter don’t see any changes in <edit-config>/<edit-data> and <copy-config> behaviors.

The server’s copy referenced nodes from <system> to the target datastore MUST be enforced at the end of the <edit-config>/<edit-data> or <copy-config> operations, regardless of which target datastore it is.

1.5. Updates to RFC 8040

This document extends Sections 4.8 and 9.1.1 of [RFC8040] to add a new query parameter "resolve-system" and corresponding query parameter capability URI.

1.5.1. Query Parameter

The "resolve-system" parameter controls whether to allow a server copy any referenced system-defined configuration automatically without the client doing so explicitly. This parameter is only allowed with no values carried. If this parameter has any unexpected value, then a "400 Bad Request" status-line is returned.

Name	Methods	Description
resolve-system	POST, PUT PATCH	resolve any references not resolved by the client and copy referenced system configuration into <running> automatically. This parameter can be given in any order.

Figure 1: RESTCONF "resolve-system" Query Parameter

1.5.2. Query Parameter URI

To enable a RESTCONF client to discover if the "resolve-system" query parameter is supported by the server, the following capability URI is defined, which is advertised by the server if supported, using the "ietf-restconf-monitoring" module defined in RFC 8040:

urn:ietf:params:restconf:capability:resolve-system:1.0

Comment: Should we define a similar capability identifier for NETCONF protocol?

2. Kinds of System Configuration

There are three types of system configurations defined in this document: immediately-active system configuration, conditionally-active system configuration, and inactive-until-referenced system configuration.

Active system configuration refers to configuration that is in use by a device. As per definition of the operational state datastore in [RFC8342], if system configuration is inactive, it should not appear

in <operational>. However, system configuration is present in <system> once it is generated, regardless of whether it is active or not.

2.1. Immediately-Active

Immediately-active system configurations are those generated in <system> and applied immediately when the device is powered on (e.g., a loopback interface), irrespective of physical resource present or not, a special functionality enabled or not.

2.2. Conditionally-Active

System configurations which are generated in <system> and applied based on specific conditions being met in a system, e.g., if a physical resource is present (e.g., insert interface card), the system will automatically detect it and load pre-provisioned configuration; when the physical resource is not present(remove interface card), the system configuration will be automatically cleared. Another example is when a special functionality is enabled, e.g., when a QoS feature is enabled, related QoS policies are automatically created by the system.

2.3. Inactive-Until-Referenced

There are some system configurations predefined (e.g., application ids, anti-x signatures, trust anchor certs, etc.) as a convenience for the clients, which must be referenced to be active. The clients can also define their own configurations for their unique requirements. Inactive-until-referenced system configurations are generated in <system> immediately when the device is powered on, but they are not active until being referenced.

3. The System Configuration Datastore (<system>)

NMDA servers compliant with this document MUST implement a system configuration datastore, and they SHOULD also implement <intended>.

Following guidelines for defining datastores in the appendix A of [RFC8342], this document introduces a new datastore resource named 'system' that represents the system configuration.

* Name: "system"

* YANG modules: all

* YANG nodes: all "config true" data nodes up to the root of the tree, generated by the system

* Management operations: The content of the datastore is set by the server in an implementation dependent manner. The content can not be changed by management operations via protocols such as NETCONF, RESTCONF, but may change itself by upgrades and/or when resource-conditions are met. The datastore can be read using the standard network management protocols such as NETCONF and RESCTCONF.

* Origin: This document does not define any new origin identity when it interacts with <intended> and flows into <operational>. The "system" origin Metadata Annotation [RFC7952] is used to indicate the origin of a data item is system.

* Protocols: YANG-driven management protocols, such as NETCONF and RESTCONF.

* Defining YANG module: "ietf-system-datastore".

The datastore's content is defined by the server and read-only to clients. Upon the content is created or changed, it will be merged into <intended>. Unlike <factory-default> [RFC8808], it MAY change dynamically, e.g., depending on factors like device upgrade or system-controlled resources change (e.g., HW available). The system configuration datastore doesn't persist across reboots; the contents of <system> will be lost upon reboot and recreated by the system with the same or changed contents. <factory-reset> RPC operation defined in [RFC8808] can reset it to its factory default configuration without including configuration generated due to the system update or client-enabled functionality.

The system datastore is defined as a conventional configuration datastore and shares a common datastore schema with other conventional datastores.

4. Static Characteristics of <system>

4.1. Read-only to Clients

The system datastore is a read-only configuration datastore (i.e., edits towards <system> directly MUST be denied), though the client may be allowed to override the value of a system-initialized data node (see Section 5.4).

4.2. May Change via Software Upgrades or Resource Changes

System configuration may change dynamically, e.g., depending on factors like device upgrade or if system-controlled resources (e.g., HW available) change. In some implementations, when a QoS feature is enabled, QoS-related policies are created by the system.

If the system configuration gets changed, YANG notifications (e.g., "push-change-update" notification) [RFC6470][RFC8639][RFC8641] can be used to notify the client. Any update of the contents in <system> will not cause the automatic update of <running>, even if some of the system configuration has already been copied into <running> explicitly or automatically before the update.

4.3. No Impact to <operational>

This work intends to have no impact to <operational>. System configuration appears in <operational> with "origin=system". This document enables a subset of those system generated nodes to be defined like configuration, i.e., made visible to clients in order for being referenced or configurable prior to present in <operational>. "Config false" nodes are out of scope, hence existing "config false" nodes are not impacted by this work.

5. Dynamic Behavior

5.1. Conceptual Model of Datastores

This document introduces a datastore named "system" which is used to hold all three types of system configurations defined in Section 2.

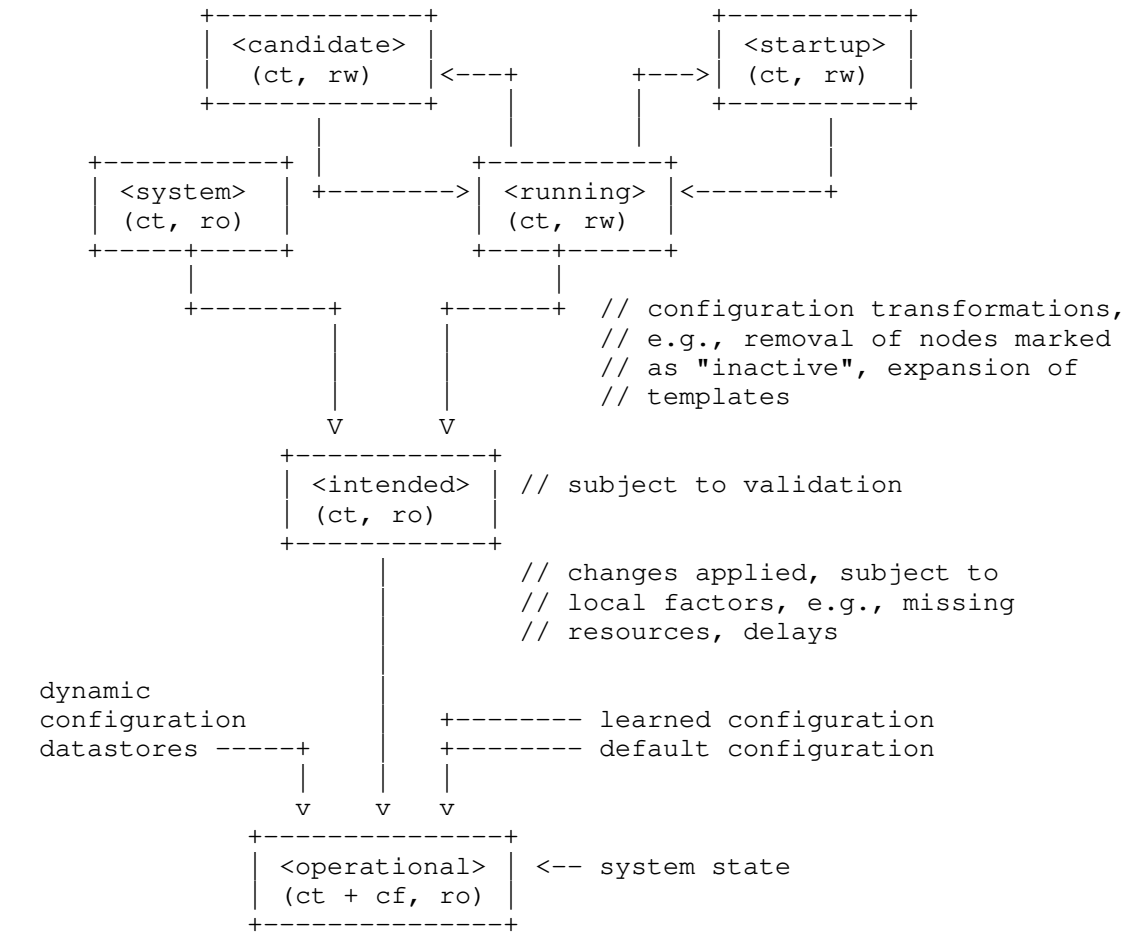
When the device is powered on, immediately-active system configuration will be generated in <system> and active immediately, but inactive-until-referenced system configuration only becomes active if it is referenced by client-defined configuration. While conditionally-active system configuration will only be created and active if the condition on system resources is met when the device is powered on or running.

All above three types of system configurations will appear in <system>. Clients MAY reference nodes defined in <system>, override values of configurations defined in <system>, and configure descendant nodes of system-defined nodes, by copying or writing intended configurations into the target configuration datastore (e.g., <running>).

Configuration in <system> is merged with the current configuration of the device in <running> after the configuration transformations (e.g., template expansion, removal of inactive configuration defined in [RFC8342]) to create the contents of <intended>, in which process, the data node appearing in <running> takes precedence over the same node in <system> if the server allows the node to be modifiable; additional nodes to a list entry or new list/leaf-list entries appearing in <running> extends the list entry or the whole list/leaf-list defined in <system> if the server allows the list/leaf-list to

be updated. In addition, the intended configuration datastore represents the configuration after all configuration transformation to <system> are performed (e.g., system-defined template expansion, removal of inactive system configuration). If a server implements <intended>, <system> MUST be merged into <intended>.

As a result, Figure 2 in Section 5 of RFC 8342 is updated with the below conceptual model of datastores which incorporates the system configuration datastore.



ct = config true; cf = config false
rw = read-write; ro = read-only
boxes denote named datastores

Figure 2: Architectural Model of Datastores

Servers MUST enforce that configuration references in <running> are resolved within <running> and ensure that <running> contains any referenced system configuration. Clients MUST either explicitly copy system-defined nodes into <running> or use the "resolve-system" parameter. The server MUST enforce that the referenced system nodes configured into <running> by the client is consistent with <system>. Note that <system> aware clients know how to discover what nodes exist in <system>. How clients unaware of the system datastore can find appropriate configurations is beyond the scope of this document.

No matter how the referenced system configurations are copied into <running>, the nodes copied into <running> would always be returned as the result of a read of <running>, regardless if the client is <system> aware.

Configuration defined in <system> is merged into <intended>. It is also present in <operational> if it is in use by the device, even if a client may delete the configuration which is copied from <system> into <running>. For example, system initializes a value for a particular leaf which is overridden by the client with a different value in <running>. The client may delete that node in <running>, in which case system-initialized value defined in <system> can be still in use and appear in <operational>.

Applied system configuration regardless of explicitly or automatically being copied into <running>, appears in <operational> with origin="system".

Comment: this might need further discussion: should the origin="system" be required for system configuration copied/pasted into <running>?

Any deletable system-provided configuration that is populated as part of <running> by the system at boot up, without being part of the contents of a <startup> datastore, must be defined in <factory-default> [RFC8808], which is used to initialize <running> when the device is first-time powered on or reset to its factory default condition.

5.2. Explicit Declaration of System Configuration

It is possible for a client to explicitly declare system configuration nodes in the target datastore (e.g., <running>) with the same values as in <system>, by configuring a node (list/leaf-list entry, leaf, etc.) in the target datastore (e.g., <running>) that matches the same node and value in <system>.

The explicit configuration of system-defined nodes in the target datastore (e.g., <running>) can be useful, for example, when the client doesn't want a "system client" to have a role or hasn't implemented the "resolve-system" parameter but need the datastore to be valid. The client can explicitly declare (i.e., configure in the datastore like <running>) the list entries (with at least the keys) for any system configuration list entries that are referenced elsewhere in <running>. The client does not necessarily need to declare all the contents of the list entry (i.e. the descendant nodes) , only the parts that are required to make the datastore appear valid.

5.3. Servers Auto-configuring Referenced System Configuration ("resolve-system" parameter)

This document defines a new parameter "resolve-system" to the input for the <edit-config>, <edit-data>, and <copy-config> operations. Clients that are aware of the "resolve-system" parameter MAY use this parameter to avoid the requirement to provide a referentially complete configuration in <running>.

If the "resolve-system" is present, and the server supports this capability, the server MUST copy relevant referenced system-defined nodes into the target datastore (e.g., <running>) without the client doing the copy/paste explicitly, to resolve any references not resolved by the client. The server acting as a "system client" like any other remote clients copies the referenced system-defined nodes when triggered by the "resolve-system" parameter.

The server may automatically configure the list entries (with at least the keys) in the target datastore (e.g., <running>) for any system configuration list entries that are referenced elsewhere by the clients. Similarly, not all the contents of the list entry (i.e., the descendant nodes) are necessarily copied by the server - only the parts that are required to make <running> valid.

There is no distinction between the configuration in the target datastore (e.g., <running>) which is automatically configured by the server and the one explicitly declared by the client, e.g., a read back of the datastore (i.e., <get>, <get-config> or <get-data> operation) returns automatically configured nodes. Note that even an auto-configured node is allowed to be deleted from the target datastore by the client, the operation request (e.g., <edit-config>) may not succeed due to incomplete referential integrity, it is also possible that the system automatically configures the deleted node again to make configuration valid, when a "resolve-system" parameter is carried. System configuration once copied into <running> will not be removed or updated automatically by the server even all references

to it are deleted or system configuration no longer appears in <system> due to factors like device upgrade or system-controlled resources (e.g., HW unavailable) change.

Comment: Should the server update configuration in <running> that is copied from <system> automatically (and manually?) during an upgrade? Jason: I think maybe servers that convert configuration during upgrade (a common approach) would want to convert/upgrade system config as well as any copied system config that exists in running.

If the "resolve-system" parameter is not given by the client, the server should not modify <running> in any way otherwise not specified by the client. Not using capitalized "SHOULD NOT" in the previous sentence is intentional. The intention is to bring awareness to the general need to not surprise clients with unexpected changes. It is desirable for clients to always opt into using mechanisms having server-side changes. This document enables a client to opt into this behavior using the "resolve-system" parameter. An example of this type of opt-in behavior can also be found in RFC 7317, which enables a client to opt into its behavior using a "0" prefix (see ianach:crypt-hash type defined in [RFC7317]).

Support for the "resolve-system" parameter is OPTIONAL. Non-NMDA servers MAY also implement this parameter without implementing the system configuration datastore, which would only eliminate the ability to expose the system configuration via protocol operations. If a server implements <system>, referenced system configuration is copied from <system> into the target datastore (e.g., <running>) when the "resolve-system" parameter is used; otherwise it is an implementation decision where to copy referenced system configuration into the target datastore (e.g., <running>).

Comments from Jason: Overall the resolve-system function may mean an expensive (time consuming) operation on the server side. Conceptually it may mean doing a validation on the running, and then when an error is hit, searching the 'system' datastore for something that could resolve that invalid aspect. Then running validation again and hitting the next error. It may require multiple passes (since some errors are dependent on the previous error being present or 'fixed').

5.4. Modifying (Overriding) System Configuration

In some cases, a server may allow some parts of system configuration to be modified. Modification of system configuration is achieved by the client writing configuration to <running> that overrides the system configuration. Configurations defined in <running> take precedence over system configuration nodes in <system> if the server allows the nodes to be modified.

For instance, descendant nodes in a system-defined list entry may be modifiable or not, even if some system configuration has been copied into <running> earlier. If a system node is non-modifiable, then writing a different value for that node MUST return an error. The immutability of system configuration is further defined in [I-D.ma-netmod-immutable-flag].

A server may also allow a client to add data nodes to a list entry in <system> by writing those additional nodes in <running>. Those additional data nodes may not exist in <system> (i.e., an *addition* rather than an override).

5.5. Examples

This section shows some examples of server-configuring of <running> automatically, declaring a system-defined node in <running> explicitly, modifying a system-instantiated leaf's value and configuring descendant nodes of a system-defined node. For each example, the corresponding XML snippets are provided.

5.5.1. Server Configuring of <running> Automatically

In this subsection, the following fictional module is used:

```
module example-application {
  yang-version 1.1;
  namespace "urn:example:application";
  prefix "app";

  import ietf-inet-types {
    prefix "inet";
  }
  container applications {
    list application {
      key "name";
      leaf name {
        type string;
      }
      leaf protocol {
        type enumeration {
          enum tcp;
          enum udp;
        }
      }
      leaf destination-port {
        type inet:port-number;
      }
    }
  }
}
```

The server may predefine some applications as a convenience for the clients. These predefined configurations are active only after being referenced by other configurations, which fall into the "inactive-until-referenced" system configuration as defined in Section 2. The system-instantiated application entries may be present in <system> as follows:

```
<applications xmlns="urn:example:application">
  <application>
    <name>ftp</name>
    <protocol>tcp</protocol>
    <destination-port>21</destination-port>
  </application>
  <application>
    <name>tftp</name>
    <protocol>udp</protocol>
    <destination-port>69</destination-port>
  </application>
  <application>
    <name>smtp</name>
    <protocol>tcp</protocol>
    <destination-port>25</destination-port>
  </application>
  ...
</applications>
```

The client may also define its customized applications. Suppose the configuration of applications is present in <running> as follows:

```
<applications xmlns="urn:example:application">
  <application>
    <name>my-app-1</name>
    <protocol>tcp</protocol>
    <destination-port>2345</destination-port>
  </application>
  <application>
    <name>my-app-2</name>
    <protocol>udp</protocol>
    <destination-port>69</destination-port>
  </application>
</applications>
```

A fictional ACL YANG module is used as follows, which defines a leafref for the leaf-list "application" data node to refer to an existing application name.

```
module example-acl {
  yang-version 1.1;
  namespace "urn:example:acl";
  prefix "acl";

  import example-application {
    prefix "app";
  }
  import ietf-inet-types {
    prefix "inet";
  }

  container acl {
    list acl_rule {
      key "name";
      leaf name {
        type string;
      }
      container matches {
        choice l3 {
          container ipv4 {
            leaf source_address {
              type inet:ipv4-prefix;
            }
            leaf dest_address {
              type inet:ipv4-prefix;
            }
          }
        }
        choice applications {
          leaf-list application {
            type leafref {
            path "/app:applications/app:application/app:name";
            }
          }
        }
      }
      leaf packet_action {
        type enumeration {
          enum forward;
          enum drop;
          enum redirect;
        }
      }
    }
  }
}
```

If a client configures an ACL rule referencing system predefined nodes which are not present in <running>, the client may issue an <edit-config> operation with the parameter "resolve-system" as follows:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <acl xmlns="urn:example:acl">
        <acl_rule>
          <name>allow_access_to_ftp_tftp</name>
          <matches>
            <ipv4>
              <source_address>198.51.100.0/24</source_address>
              <dest_address>192.0.2.0/24</dest_address>
            </ipv4>
            <application>ftp</application>
            <application>tftp</application>
            <application>my-app-1</application>
          </matches>
          <packet_action>forward</packet_action>
        </acl_rule>
      </acl>
    </config>
    <resolve-system/>
  </edit-config>
</rpc>
```

Then following gives the configuration of applications in <running> which is returned in the response to a follow-up <get-config> operation:

```
          <applications xmlns="urn:example:application">
            <application>
              <name>my-app-1</name>
              <protocol>tcp</protocol>
              <destination-port>2345</destination-port>
            </application>
            <application>
              <name>my-app-2</name>
              <protocol>udp</protocol>
              <destination-port>69</destination-port>
            </application>
            <application>
              <name>ftp</name>
            </application>
            <application>
              <name>tftp</name>
            </application>
          </applications>
```

Then the configuration of applications is present in <operational> as follows:

```
     <applications xmlns="urn:example:application"
                   xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                   or:origin="or:intended">
       <application>
         <name>my-app-1</name>
         <protocol>tcp</protocol>
         <destination-port>2345</destination-port>
       </application>
       <application>
         <name>my-app-2</name>
         <protocol>udp</protocol>
         <destination-port>69</destination-port>
       </application>
       <application or:origin="or:system">
         <name>ftp</name>
         <protocol>tcp</protocol>
         <destination-port>21</destination-port>
       </application>
       <application or:origin="or:system">
         <name>tftp</name>
         <protocol>udp</protocol>
         <destination-port>69</destination-port>
       </application>
     </applications>
```

Since the configuration of application "smtp" is not referenced by the client, and the server treats application "smtp" configuration as "inactive-until-referenced", it does not appear in <operational> but only in <system>.

5.5.2. Declaring a System-defined Node in <running> Explicitly

It's also possible for a client to explicitly declare the system-defined configurations that are referenced. For instance, in the above example, the client MAY also explicitly configure the following system defined applications "ftp" and "tftp" only with the list key "name" before referencing:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
     <target>
       <running/>
     </target>
     <config>
       <applications xmlns="urn:example:application">
         <application>
           <name>ftp</name>
         </application>
         <application>
           <name>tftp</name>
         </application>
       </applications>
     </config>
   </edit-config>
 </rpc>
```

Then the client issues an <edit-config> operation to configure an ACL rule referencing applications "ftp" and "tftp" without the parameter "resolve-system" as follows:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <acl xmlns="urn:example:acl">
        <acl_rule>
          <name>allow_access_to_ftp_tftp</name>
          <matches>
            <ipv4>
              <source_address>198.51.100.0/24</source_address>
              <dest_address>192.0.2.0/24</dest_address>
            </ipv4>
            <application>ftp</application>
            <application>tftp</application>
            <application>my-app-1</application>
          </matches>
          <packet_action>forward</packet_action>
        </acl_rule>
      </acl>
    </config>
  </edit-config>
</rpc>
```

Then following gives the configuration of applications in <running> which is returned in the response to a follow-up <get-config> operation, all the configuration of applications are explicitly configured by the client:

```
          <applications xmlns="urn:example:application">
            <application>
              <name>my-app-1</name>
              <protocol>tcp</protocol>
              <destination-port>2345</destination-port>
            </application>
            <application>
              <name>my-app-2</name>
              <protocol>udp</protocol>
              <destination-port>69</destination-port>
            </application>
            <application>
              <name>ftp</name>
            </application>
            <application>
              <name>tftp</name>
            </application>
          </applications>
```

Then the configuration of applications is present in <operational> as follows:

```
     <applications xmlns="urn:example:application"
                   xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                   or:origin="or:intended">
       <application>
         <name>my-app-1</name>
         <protocol>tcp</protocol>
         <destination-port>2345</destination-port>
       </application>
       <application>
         <name>my-app-2</name>
         <protocol>udp</protocol>
         <destination-port>69</destination-port>
       </application>
       <application>
         <name>ftp</name>
         <protocol or:origin="or:system">tcp</protocol>
         <destination-port or:origin="or:system">21</destination-port>
       </application>
       <application>
         <name>tftp</name>
         <protocol or:origin="or:system">udp</protocol>
         <destination-port or:origin="or:system">69</destination-port>
       </application>
     </applications>
```

Since the application names "ftp" and "tftp" are explicitly configured by the client, they take precedence over the values in <system>, the "origin" attribute will be set to "intended".

5.5.3. Modifying a System-instantiated Leaf's Value

In this subsection, we will use this fictional QoS data model:

```
module example-qos-policy {
  yang-version 1.1;
  namespace "urn:example:qos";
  prefix "qos";

  container qos-policies {
     list policy {
       key "name";
       leaf name {
       type string;
     }
       list queue {
         key "queue-id";
           leaf queue-id {
             type int32 {
               range "1..32";
             }
           }
           leaf maximum-burst-size {
             type int32 {
               range "0..100";
             }
           }
       }
     }
   }
```

Suppose a client creates a qos policy "my-policy" with 4 system instantiated queues(1~4). The configuration of qos-policies is present in <system> as follows:

```
<qos-policies xmlns="urn:example:qos">
  <name>my-policy</name>
  <queue>
    <queue-id>1</queue-id>
    <maximum-burst-size>50</maximum-burst-size>
  </queue>
  <queue>
    <queue-id>2</queue-id>
    <maximum-burst-size>60</maximum-burst-size>
  </queue>
  <queue>
    <queue-id>3</queue-id>
    <maximum-burst-size>70</maximum-burst-size>
  </queue>
  <queue>
    <queue-id>4</queue-id>
    <maximum-burst-size>80</maximum-burst-size>
  </queue>
</qos-policies>
```

A client modifies the value of maximum-burst-size to 55 in queue-id 1:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <qos-policies xmlns="urn:example:qos">
        <name>my-policy</name>
        <queue>
          <queue-id>1</queue-id>
          <maximum-burst-size>55</maximum-burst-size>
        </queue>
      </qos-policies>
    </config>
  </edit-config>
</rpc>
```

Then, the configuration of qos-policies is present in <operational> as follows:

```
<qos-policies  xmlns="urn:example:qos"
               xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
               or:origin="or:intended">
  <name>my-policy</name>
  <queue>
    <queue-id>1</queue-id>
    <maximum-burst-size>55</maximum-burst-size>
  </queue>
  <queue or:origin="or:system">
    <queue-id>2</queue-id>
    <maximum-burst-size>60</maximum-burst-size>
  </queue>
   <queue or:origin="or:system">
    <queue-id>3</queue-id>
    <maximum-burst-size>70</maximum-burst-size>
  </queue>
   <queue or:origin="or:system">
    <queue-id>4</queue-id>
    <maximum-burst-size>80</maximum-burst-size>
  </queue>
</qos-policies>
```

5.5.4. Configuring Descendant Nodes of a System-defined Node

This subsection also uses the fictional interface YANG module defined in Appendix C.3 of [RFC8342]. Suppose the system provides a loopback interface (named "lo0") with a default IPv4 address of "127.0.0.1" and a default IPv6 address of "::1".

The configuration of "lo0" interface is present in <system> as follows:

```
<interfaces>
  <interface>
    <name>lo0</name>
    <ip-address>127.0.0.1</ip-address>
    <ip-address>::1</ip-address>
  </interface>
</interfaces>
```

The configuration of "lo0" interface is present in <operational> as follows:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
            or:origin="or:system">
  <interface>
    <name>lo0</name>
    <ip-address>127.0.0.1</ip-address>
    <ip-address>::1</ip-address>
  </interface>
</interfaces>
```

Later on, the client further configures the description node of a "lo0" interface as follows:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interfaces>
        <interface>
          <name>lo0</name>
          <description>loopback</description>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

Then the configuration of interface "lo0" is present in <operational> as follows:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
            or:origin="or:intended">
  <interface>
    <name>lo0</name>
    <description>loopback</description>
    <ip-address or:origin="or:system">127.0.0.1</ip-address>
    <ip-address or:origin="or:system">::1</ip-address>
  </interface>
</interfaces>
```

6. The "ietf-system-datastore" Module

6.1. Data Model Overview

This YANG module defines a new YANG identity named "system" that uses the "ds:datastore" identity defined in [RFC8342]. A client can discover the system configuration datastore support on the server by reading the YANG library information from the operational state datastore. Note that no new origin identity is defined in this document, the "or:system" origin Metadata Annotation [RFC7952] is used to indicate the origin of a data item is system. Support for the "origin" annotation is identified with the feature "origin" defined in [RFC8526].

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-system-datastore" and "ietf-datastores" YANG modules:

```
Identities:
    +--- datastore
    |  +--- conventional
    |  |  +--- running
    |  |  +--- candidate
    |  |  +--- startup
    |  |  +--- system
    |  |  +--- intended
    |  +--- dynamic
    |  +--- operational
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

6.2. Example Usage

This section gives an example of data retrieval from <system>. The YANG module used are shown in Appendix C.2 of [RFC8342]. All the messages are presented in a protocol-independent manner. JSON is used only for its conciseness.

Suppose the following data is added to <running>:

```
{
    "bgp": {
        "local-as": "64501",
        "peer-as": "64502",
        "peer": {
            "name": "2001:db8::2:3"
        }
    }
}
```

REQUEST (a <get-data> or GET request sent from the NETCONF or RESTCONF client):

Datastore: <system>
Target:/bgp

An example of RESTCONF request:

```
GET /restconf/ds/system/bgp HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

RESPONSE ("local-port" leaf value is supplied by the system):

```
{
    "bgp": {
        "peer": {
            "name": "2001:db8::2:3",
            "local-port": "60794"
        }
    }
}
```

6.3. YANG Module

<CODE BEGINS> file "ietf-system-datastore@2023-10-19.yang"

```
module ietf-system-datastore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-system-datastore";
  prefix sysds;

  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture(NMDA)";
  }

  organization
    "IETF NETDOD (Network Modeling) Working Group";
  contact
    "WG Web:   https://datatracker.ietf.org/wg/netmod/
     WG List:  NETMOD WG list <mailto:netmod@ietf.org>

     Author: Qiufang Ma
             <mailto:maqiufang1@huawei.com>
     Author: Qin Wu
             <mailto:bill.wu@huawei.com>
```

```
        Author: Chong Feng
                <mailto:frank.fengchong@huawei.com>";
      description
        "This module defines a new YANG identity that uses the
         ds:datastore identity defined in [RFC8342].

         Copyright (c) 2022 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 Revised
         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).

         This version of this YANG module is part of RFC HHHH
         (https://www.rfc-editor.org/info/rfcHHHH); see the RFC
         itself for full legal notices.

         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 (RFC 2119)
         (RFC 8174) when, and only when, they appear in all
         capitals, as shown here.";

      revision 2023-10-19 {
        description
          "Initial version.";
        reference
          "RFC XXXX: System-defined Configuration";
      }

      identity system {
        base ds:conventional;
        description
          "This read-only datastore contains the configuration
           provided by the system itself.";
      }
    }

    <CODE ENDS>
```

7. The "ietf-netconf-resolve-system" Module

This YANG module is optional to implement.

7.1. Data Model Overview

This YANG module augments NETCONF <edit-config>, <edit-data> and <copy-config> operations with a new parameter "resolve-system" in the input parameters. If the "resolve-system" parameter is present, the server will copy the referenced system configuration into target datastore automatically. A NETCONF client can discover the "resolve-system" parameter support on the server by checking the YANG library information with "ietf-netconf-resolve-system" YANG module included from the operational state datastore.

The following tree diagram [RFC8340] illustrates the "ietf-netconf-resolve-system" module:

```
module: ietf-netconf-resolve-system
  augment /nc:edit-config/nc:input:
    +---w resolve-system?   empty
  augment /nc:copy-config/nc:input:
    +---w resolve-system?   empty
  augment /ncds:edit-data/ncds:input:
    +---w resolve-system?   empty
```

The following tree diagram [RFC8340] illustrates "edit-config", "copy-config" and "edit-data" rpcs defined in "ietf-netconf" and "ietf-netconf-nmda" respectively, augmented by "ietf-netconf-resolve-system" YANG module:

```
  rpcs:
    +---x edit-config
    |  +---w input
    |     +---w target
    |     |  +---w (config-target)
    |     |     +--:(candidate)
    |     |     |  +---w candidate?   empty {candidate}?
    |     |     +--:(running)
    |     |        +---w running?     empty {writable-running}?
    |     +---w default-operation?   enumeration
    |     +---w test-option?         enumeration {validate}?
    |     +---w error-option?        enumeration
    |     +---w (edit-content)
    |     |   +--:(config)
    |     |   |  +---w config?        <anyxml>
    |     |   +--:(url)
    |     |     +---w url?           inet:uri {url}?
    |     +---w resolve-system?      empty
    +---x copy-config
    |  +---w input
    |     +---w target
```

```
      |      |  +---w (config-target)
      |      |     +--:(candidate)
      |      |     |  +---w candidate?   empty {candidate}?
      |      |     +--:(running)
      |      |     |  +---w running?     empty {writable-running}?
      |      |     +--:(startup)
      |      |     |  +---w startup?     empty {startup}?
      |      |     +--:(url)
      |      |        +---w url?         inet:uri {url}?
      |      +---w source
      |      |  +---w (config-source)
      |      |     +--:(candidate)
      |      |     |  +---w candidate?   empty {candidate}?
      |      |     +--:(running)
      |      |     |  +---w running?     empty
      |      |     +--:(startup)
      |      |     |  +---w startup?     empty {startup}?
      |      |     +--:(url)
      |      |     |  +---w url?         inet:uri {url}?
      |      |     +--:(config)
      |      |        +---w config?      <anyxml>
      |      +---w resolve-system?       empty
      +---x edit-data
         +---w input
            +---w datastore            ds:datastore-ref
            +---w default-operation?   enumeration
            +---w (edit-content)
            |  +--:(config)
            |  |  +---w config?        <anydata>
            |  +--:(url)
            |     +---w url?           inet:uri {nc:url}?
            +---w resolve-system?      empty
```

7.2. Example Usage

This section gives an example of an <edit-config> request to reference system-defined data nodes which are not present in <running> with a "resolve-system" parameter. A retrieval of <running> to show the auto-copied referenced system configurations after the <edit-config> request is also given. The YANG module used is shown as follows, leafrefs refer to an existing name and address of an interface:

```
module example-interface-management {
  yang-version 1.1;
  namespace "urn:example:interfacemgmt";
  prefix "inm";

  container interfaces {
    list interface {
      key name;
      leaf name {
        type string;
      }
      leaf description {
        type string;
      }
      leaf mtu {
        type uint16;
      }
      leaf ip-address {
        type inet:ip-address;
      }
    }
  }
  container default-address {
    leaf ifname {
      type leafref {
        path "../../interfaces/interface/name";
      }
    }
    leaf address {
      type leafref {
        path "../../interfaces/interface[name = current()/../ifname]"
           + "/ip-address";
      }
    }
  }
}
```

Image that the system provides a loopback interface (named "lo0") with a predefined MTU value of "1500" and a predefined IP address of "127.0.0.1", <system> shows the following configuration of loopback interface:

```
   <interfaces xmlns="urn:example:interfacemgmt">
     <interface>
       <name>lo0</name>
       <mtu>1500</mtu>
       <ip-address>127.0.0.1</ip-address>
     </interface>
   </interfaces>
```

The client sends an <edit-config> operation to add the configuration of default-address with a "resolve-system" parameter:

```
  <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <edit-config>
      <target>
        <running/>
      </target>
      <config>
        <default-address xmlns="urn:example:interfacemgmt">
          <if-name>lo0</if-name>
          <address>127.0.0.1</address>
        </default-address>
      </config>
     <resolve-system/>
    </edit-config>
  </rpc>
```

Since the "resolve-system" parameter is provided, the server will resolve any leafrefs to system configurations and copy the referenced system-defined nodes into <running> automatically with the same value (i.e., the name and ip-address data nodes of lo0 interface) in <system> at the end of <edit-config> operation constraint enforcement. After the processing, a positive response is returned:

```
   <rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <ok/>
   </rpc-reply>
```

Then the client sends a <get-config> operation towards <running>:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <interfaces xmlns="urn:example:interfacemgmt"/>
    </filter>
  </get-config>
</rpc>
```

Given that the referenced interface "name" and "ip-address" of lo0 are configured by the server, the following response is returned:

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="urn:example:interfacemgmt">
      <interface>
        <name>lo0</name>
        <ip-address>127.0.0.1</ip-address>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

7.3. YANG Module

```
<CODE BEGINS> file "ietf-netconf-resolve-system@2023-10-19.yang"

module ietf-netconf-resolve-system {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system";
  prefix ncrs;

  import ietf-netconf {
    prefix nc;
    reference
      "RFC 6241: Network Configuration Protocol (NETCONF)";
  }
  import ietf-netconf-nmda {
    prefix ncds;
    reference
      "RFC 8526: NETCONF Extensions to Support the Network
       Management Datastore Architecture";
  }
```

```
  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author: Qiufang Ma
             <mailto:maqiufang1@huawei.com>
     Author: Qin Wu
             <mailto:bill.wu@huawei.com>
     Author: Chong Feng
             <mailto:frank.fengchong@huawei.com>";
  description
    "This module defines an extension to the NETCONF protocol
     that allows the NETCONF client to control whether the server
     is allowed to copy referenced system configuration
     automatically without the client doing so explicitly.

      Copyright (c) 2022 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 Revised
      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).

      This version of this YANG module is part of RFC HHHH
      (https://www.rfc-editor.org/info/rfcHHHH); see the RFC
      itself for full legal notices.

      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 (RFC 2119)
      (RFC 8174) when, and only when, they appear in all
      capitals, as shown here.";

  revision 2023-10-19 {
    description
      "Initial version.";
    reference
      "RFC XXXX: System-defined Configuration";
  }

  grouping resolve-system-grouping {
    description
```

```
        "Define the resolve-system parameter grouping.";
      leaf resolve-system {
        type empty;
        description
          "When present, the server is allowed to automatically
           configure referenced system configuration into the
           target configuration datastore.";
      }
    }

    augment "/nc:edit-config/nc:input" {
      description
        "Allows the server to automatically configure
         referenced system configuration to make configuration
         valid.";
      uses resolve-system-grouping;
    }

    augment "/nc:copy-config/nc:input" {
      description
        "Allows the server to automatically configure
         referenced system configuration to make configuration
         valid.";
      uses resolve-system-grouping;
    }

    augment "/ncds:edit-data/ncds:input" {
      description
        "Allows the server to automatically configure
         referenced system configuration to make configuration
         valid.";
      uses resolve-system-grouping;
    }
  }

  <CODE ENDS>
```

8. IANA Considerations

8.1. The "IETF XML" Registry

This document registers two XML namespace URNs in the 'IETF XML registry', following the format defined in [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-system-datastore
Registrant Contact: The IESG.
XML: N/A, the requested URIs are XML namespaces.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system
Registrant Contact: The IESG.
XML: N/A, the requested URIs are XML namespaces.

8.2. The "YANG Module Names" Registry

This document registers two module names in the 'YANG Module Names' registry, defined in [RFC6020] .

name: ietf-system-datastore
prefix: sys
namespace: urn:ietf:params:xml:ns:yang:ietf-system-datatstore
maintained by IANA: N
RFC: XXXX // RFC Ed.: replace XXXX and remove this comment

name: ietf-netconf-resolve-system
prefix: ncrs
namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-resolve-system
maintained by IANA: N
RFC: XXXX // RFC Ed.: replace XXXX and remove this comment

8.3. RESTCONF Capability URN Registry

This document registers a capability in the "RESTCONF Capability URNs" registry [RFC8040]:

Index	Capability Identifier
:resolve-system	urn:ietf:params:restconf:capability:resolve-system:1.0

9. Security Considerations

9.1. Regarding the "ietf-system-datastore" YANG Module

The YANG module defined in this document extends the base operations for NETCONF [RFC6241] and 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 [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

9.2. Regarding the "ietf-netconf-resolve-system" YANG Module

The YANG module defined in this document extends the base operations for NETCONF [RFC6241] and [RFC8526]. 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 [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

The security considerations for the base NETCONF protocol operations (see Section 9 of [RFC6241] apply to the new extended RPC operations defined in this document.

10. Contributors

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com

Chongfeng Xie
China Telecom
Beijing
China

Email: xiechf@chinatelecom.cn

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Acknowledgements

The authors would like to thank for following for discussions and providing input to this document (ordered by first name): Alex Clemm, Andy Bierman, Balazs Lengyel, Juergen Schoenwaelder, Martin Bjorklund, Mohamed Boucadair, Robert Wilton and Timothy Carey.

References

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, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <https://www.rfc-editor.org/info/rfc6470>.

[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>.

[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>.

[RFC8526] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", RFC 8526, DOI 10.17487/RFC8526, March 2019, <https://www.rfc-editor.org/info/rfc8526>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

Informative References

[I-D.ma-netmod-immutable-flag] Ma, Q., Wu, Q., Lengyel, B., and H. Li, "YANG Extension and Metadata Annotation for Immutable Flag", Work in Progress, Internet-Draft, draft-ma-netmod-immutable-flag-08, 9 July 2023, <https://datatracker.ietf.org/doc/html/draft-ma-netmod-immutable-flag-08>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://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, <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>.

[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>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

[RFC8808] Wu, Q., Lengyel, B., and Y. Niu, "A YANG Data Model for Factory Default Settings", RFC 8808, DOI 10.17487/RFC8808, August 2020, <https://www.rfc-editor.org/info/rfc8808>.

Appendix A. Key Use Cases

Following provides three use cases related to system-defined configuration lifecycle management. The simple interface data model defined in Appendix C.3 of [RFC8342] is used. For each use case, snippets of <running>, <system>, <intended> and <operational> are shown.

A.1. Device Powers On

<running>:

No configuration for "lo0" appears in <running>;

<system>:

```
     <interfaces>
       <interface>
         <name>lo0</name>
         <ip-address>127.0.0.1</ip-address>
         <ip-address>::1</ip-address>
       </interface>
     </interfaces>
```

<intended>:

```
        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>

   <operational>:

        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:system">
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
```

A.2. Client Commits Configuration

If a client creates an interface "et-0/0/0" but the interface does not physically exist at this point:

```
   <running>:

        <interfaces>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
        </interfaces>

   <system>:

        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>

   <intended>:
```

```
        <interfaces>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
          <interface>
        </interfaces>

   <operational>:

        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:intended">
          <interface or:origin="or:system">
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
        </interfaces>
```

A.3. Operator Installs Card into a Chassis

```
   <running>:

        <interfaces>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
          </interface>
        </interfaces>

   <system>:

        <interfaces>
          <interface>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <mtu>1500</mtu>
          </interface>
        </interfaces>
```

```
   <intended>:

        <interfaces>
            <name>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
          <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
            <mtu>1500</mtu>
          </interface>
          <interface>
        </interfaces>

   <operational>:

        <interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
                    or:origin="or:intended">
          <interface or:origin="or:system">
            <name or:origin>lo0</name>
            <ip-address>127.0.0.1</ip-address>
            <ip-address>::1</ip-address>
          </interface>
         <interface>
            <name>et-0/0/0</name>
            <description>Test interface</description>
            <mtu or:origin="or:system">1500</mtu>
          </interface>
          <interface>
        </interfaces>
```

Appendix B. Changes between Revisions

v02 - v03

* remove the merge mechanism related comments, as discussed in https://github.com/netconf-wg/netconf-next/issues/19

* Editorial changes

v01 - v02

* Define referenced system configuration

* better clarify "resolve-system" parameter

* update Figure 2 in NMDA RFC

* Editorial changes

v00 - v01

* Clarify why client's explicit copy is not preferred but cannot be avoided if resolve-system parameter is not defined

* Clarify active system configuration

* Update the timing when the server's auto copy should be enforced if a resolve-system parameter is used

* Editorial changes

Appendix C. Open Issues tracking

* Should the "with-origin" parameter be supported for <intended>?

Authors' Addresses

Qiufang Ma (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Feng Chong
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: frank.fengchong@huawei.com

Network Working Group
Internet-Draft
Updates: 6020, 7950, 8407, 8525 (if approved)
Intended status: Standards Track
Expires: 19 April 2024

R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman, Ed.
Equinix
B. Lengyel, Ed.
Ericsson
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia

17 October 2023

Updated YANG Module Revision Handling
draft-ietf-netmod-yang-module-versioning-10

Abstract

This document refines the RFC 7950 module update rules. It specifies a new YANG module update procedure that can document when non-backwards-compatible changes have occurred during the evolution of a YANG module. It extends the YANG import statement with a minimum revision suggestion to help document inter-module dependencies. It provides guidelines for managing the lifecycle of YANG modules and individual schema nodes. It provides a mechanism, via the revision label YANG extension, to specify a revision identifier for YANG modules and submodules. This document updates RFC 7950, RFC 6020, RFC 8407 and RFC 8525.

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 19 April 2024.

Table of Contents

1. Introduction

The current YANG [RFC7950] module update rules require that updates of YANG modules preserve strict backwards compatibility. This causes problems as described in [I-D.ietf-netmod-yang-versioning-reqs]. This document recognizes the need to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which can cause breakage to clients and when importing YANG modules. Accepting that non-backwards-compatible changes do sometimes occur -- e.g., for bugfixes -- it is important to have mechanisms to report when these changes occur, and to manage their effect on clients and the broader YANG ecosystem.

This document defines a flexible versioning solution. Several other documents build on this solution with additional capabilities. [I-D.ietf-netmod-yang-schema-comparison] specifies an algorithm that can be used to compare two revisions of a YANG schema and provide granular information to allow module users to determine if they are impacted by changes between the revisions. The [I-D.ietf-netmod-yang-semver] document extends the module versioning work by introducing a revision label scheme based on semantic versioning. YANG packages [I-D.ietf-netmod-yang-packages] provides a mechanism to group sets of related YANG modules together in order to manage schema and conformance of YANG modules as a cohesive set instead of individually. Finally, [I-D.ietf-netmod-yang-ver-selection] provides a schema selection mechanism that allows a client to choose which schemas to use when interacting with a server from the available schema that are

supported and advertised by the server. These other documents are mentioned here as informative references. Support of the other documents is not required in an implementation in order to take advantage of the mechanisms and functionality offered by this module versioning document.

The document comprises five parts:

* Refinements to the YANG 1.1 module revision update procedure, supported by new extension statements to indicate when a revision contains non-backwards-compatible changes, and an optional revision label.

* Updated guidance for revision selection on imports and a YANG extension statement allowing YANG module imports to document an earliest module revision that may satisfy the import dependency.

* Updates and augmentations to ietf-yang-library to include the revision label in the module and submodule descriptions, to report how "deprecated" and "obsolete" nodes are handled by a server, and to clarify how module imports are resolved when multiple revisions could otherwise be chosen.

* Considerations of how versioning applies to YANG instance data.

* Guidelines for how the YANG module update rules defined in this document should be used, along with examples.

Note to RFC Editor (To be removed by RFC Editor)

Open issues are tracked at https://github.com/netmod-wg/yang-ver-dt/issues.

1.1. Updates to YANG RFCs

This document updates [RFC7950] section 11 and [RFC6020] section 10. Section 3 describes modifications to YANG revision handling and update rules, and Section 4.1 describes a YANG extension statement to describe potential YANG import revision dependencies.

This document updates [RFC7950] section 5.2, [RFC6020] section 5.2 and [RFC8407] section 3.2. Section 3.4.1 describes the use of a revision label in the name of a file containing a YANG module or submodule.

This document updates [RFC7950] section 5.6.5 and [RFC8525]. Section 5.1 defines how a client of a YANG library datastore schema resolves ambiguous imports for modules which are not "import-only".

This document updates [RFC8407] section 4.7. Section 7 provides guidelines on managing the lifecycle of YANG modules that may contain non-backwards-compatible changes and a branched revision history.

This document updates [RFC8525] with augmentations to include revision labels in the YANG library data and two boolean leafs to indicate whether status deprecated and status obsolete schema nodes are implemented by the server.

2. Terminology and Conventions

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.

This document makes use of the following terminology introduced in the YANG 1.1 Data Modeling Language [RFC7950]:

* schema node

In addition, this document uses the following terminology:

* YANG module revision: An instance of a YANG module, uniquely identified with a revision date, with no implied ordering or backwards compatibility between different revisions of the same module.

* Backwards-compatible (BC) change: A backwards-compatible change between two YANG module revisions, as defined in Section 3.1.1

* Non-backwards-compatible (NBC) change: A non-backwards-compatible change between two YANG module revisions, as defined in Section 3.1.2

3. Refinements to YANG revision handling

[RFC7950] and [RFC6020] assume, but do not explicitly state, that the revision history for a YANG module or submodule is strictly linear, i.e., it is prohibited to have two independent revisions of a YANG module or submodule that are both directly derived from the same parent revision.

This document clarifies [RFC7950] and [RFC6020] to explicitly allow non-linear development of YANG module and submodule revisions, so that they MAY have multiple revisions that directly derive from the same parent revision. As per [RFC7950] and [RFC6020], YANG module

and submodule revisions continue to be uniquely identified by their revision date, and hence all revisions of a given module or submodule MUST have unique revision dates.

For a given YANG module revision, revision B is defined as being derived from revision A, if revision A is listed in the revision history of revision B. Although this document allows for a branched revision history, a given YANG module revision history does not contain all revisions in all possible branches, it only lists those from which is was derived, i.e., the module revision's history describes a single path of derived revisions back to the root of the module's revision history.

A corollary to the text above is that the ancestry (derived relationship) between two module or submodule revisions cannot be determined by comparing the module or submodule revision date or label alone - the revision history must be consulted.

A module's name and revision date identifies a specific immutable definition of that module within its revision history. Hence, if a module includes submodules then to ensure that the module's content is uniquely defined, the module's "include" statements SHOULD use "revision-date" substatements to specify the exact revision date of each included submodule. When a module does not include its submodules by revision-date, the revision of submodules used cannot be derived from the including module. Mechanisms such as YANG packages [I-D.ietf-netmod-yang-packages], and YANG library [RFC8525], MAY be used to specify the exact submodule revisions used when the submodule revision date is not constrained by the "include" statement.

[RFC7950] section 11 and [RFC6020] section 10 require that all updates to a YANG module are BC to the previous revision of the module. This document introduces a method to indicate that an NBC change has occurred between module revisions: this is done by using a new "non-backwards-compatible" YANG extension statement in the module revision history.

Two revisions of a module or submodule MAY have identical content except for the revision history. This could occur, for example, if a module or submodule has a branched history and identical changes are applied in multiple branches.

3.1. Updating a YANG module with a new revision

This section updates [RFC7950] section 11 and [RFC6020] section 10 to refine the rules for permissible changes when a new YANG module revision is created.

While a new module revision SHOULD NOT contain NBC changes, they are allowed. For example:

* Bugfixes, particularly where the likely client impact is low or the module is changed to reflect current server behavior.

* To mark nodes as obsolete (or remove them), after a suitable deprecation period.

* To refine new and unstable modules (or new and unstable nodes within existing, stable modules).

* Restructuring a module to add new functionality where the cost of adding the functionality in a BC manner is disproportionate to the expected benefits of greater client backwards compatibility.

A YANG extension, defined in Section 3.2, is used to signal the potential for incompatibility to existing module users and readers.

Note that NBC changes often create problems for clients, thus it is recommended to avoid making them.

As per [RFC7950] and [RFC6020], all published revisions of a module are given a new unique revision date. This applies even for module revisions containing (in the module or included submodules) only changes to any whitespace, formatting, comments or line endings (e.g., DOS vs UNIX).

3.1.1. Backwards-compatible rules

A change between two module revisions is defined as being "backwards-compatible" if the change conforms to the module update rules specified in [RFC7950] section 11 and [RFC6020] section 10, updated by the following rules:

* A "status" "deprecated" statement MAY be added, or changed from "current" to "deprecated", but adding or changing "status" to "obsolete" is a non-backwards-compatible change.

* YANG schema nodes with a "status" "obsolete" substatement MAY be removed from published modules, and the removal is classified as a backwards-compatible change. In some circumstances it may be helpful to retain the obsolete definitions since their identifiers may still be referenced by other modules and to ensure that their identifiers are not reused with a different meaning.

* A statement that is defined using the YANG "extension" statement MAY be added, removed, or changed, if it does not change the semantics of the module. Extension statement definitions SHOULD specify whether adding, removing, or changing statements defined by that extension are backwards-compatible or non-backwards-compatible.

* Any change made to the "revision-date" or "recommended-min" substatements of an "import" statement, including adding new "revision-date" or "recommended-min" substatements, changing the argument of any "revision-date" or "recommended-min" substatetements, or removing any "revision-date" or "recommended-min" substatements, is classified as backwards-compatible.

* Any changes (including whitespace or formatting changes) that do not change the semantic meaning of the module are backwards-compatible.

3.1.2. Non-backwards-compatible changes

Any changes to YANG modules that are not defined by Section 3.1.1 as being backwards-compatible are classified as "non-backwards-compatible" changes.

3.2. non-backwards-compatible extension statement

The "rev:non-backwards-compatible" extension statement is used to indicate YANG module revisions that contain NBC changes.

If a revision of a YANG module contains changes, relative to the preceding revision in the revision history, that do not conform to the module update rules defined in Section 3.1.1, then a "rev:non-backwards-compatible" extension statement MUST be added as a substatement to the "revision" statement.

Adding, modifying or removing a "rev:non-backwards-compatible" extension statement is considered to be a BC change.

3.3. Removing revisions from the revision history

Authors may wish to remove revision statements from a module or submodule. Removal of revision information may be desirable for a number of reasons including reducing the size of a large revision history, or removing a revision that should no longer be used or imported. Removing revision statements is allowed, but can cause issues and SHOULD NOT be done without careful analysis of the potential impact to users of the module or submodule. Doing so can lead to import breakages when import by recommended-min is used.

Moreover, truncating history may cause loss of visibility of when non-backwards-compatible changes were introduced.

An author MAY remove a contiguous sequence of entries from the end (i.e., oldest entries) of the revision history. This is acceptable even if the first remaining (oldest) revision entry in the revision history contains a rev:non-backwards-compatible substatement.

An author MAY remove a contiguous sequence of entries in the revision history as long as the presence or absence of any existing rev:non-backwards-compatible substatements on all remaining entries still accurately reflect the compatibility relationship to their preceding entries remaining in the revision history.

The author MUST NOT remove the first (i.e., newest) revision entry in the revision history.

Example revision history:

```
revision 2020-11-11 {
  rev:label 4.0.0;
  rev:non-backwards-compatible;
}

revision 2020-08-09 {
  rev:label 3.0.0;
  rev:non-backwards-compatible;
}

revision 2020-06-07 {
  rev:label 2.1.0;
}

revision 2020-02-10 {
  rev:label 2.0.0;
  rev:non-backwards-compatible;
}

revision 2019-10-21 {
  rev:label 1.1.3;
}

revision 2019-03-04 {
  rev:label 1.1.2;
}

revision 2019-01-02 {
  rev:label 1.1.1;
}
```

In the revision history example above, removing the revision history entry for 2020-02-10 would also remove the rev:non-backwards-compatible annotation and hence the resulting revision history would incorrectly indicate that revision 2020-06-07 is backwards-compatible with revisions 2019-01-02 through 2019-10-21 when it is not, and so this change cannot be made. Conversely, removing one or more revisions out of 2019-03-04, 2019-10-21 and 2020-08-09 from the revision history would still retain a consistent revision history, and is acceptable, subject to an awareness of the concerns raised in the first paragraph of this section.

3.4. Revision label

Each revision entry in a module or submodule MAY have a revision label associated with it, providing an alternative alias to identify a particular revision of a module or submodule. The revision label could be used to provide an additional versioning identifier associated with the revision.

A revision label scheme is a set of rules describing how a particular type of revision label operates for versioning YANG modules and submodules. For example, YANG Semver [I-D.ietf-netmod-yang-semver] defines a revision label scheme based on Semver 2.0.0 [semver]. Other documents may define other YANG revision label schemes.

Submodules MAY use a revision label scheme. When they use a revision label scheme, submodules MAY use a revision label scheme that is different from the one used in the including module.

The revision label space of submodules is separate from the revision label space of the including module. A change in one submodule MUST result in a new revision label of that submodule and the including module, but the actual values of the revision labels in the module and submodule could be completely different. A change in one submodule does not result in a new revision label in another submodule. A change in a module revision label does not necessarily mean a change to the revision label in all included submodules.

If a revision has an associated revision label, then it may be used instead of the revision date in a "rev:recommended-min" extension statement argument.

A specific revision label identifies a specific revision of the module. If two YANG modules contain the same module name and the same revision label (and hence also the same revision-date) in their latest revision statement, then the file contents of the two modules, including the revision history, MUST be identical.

3.4.1. File names

This section updates [RFC7950] section 5.2, [RFC6020] section 5.2 and [RFC8407] section 3.2

If a revision has an associated revision label, then it is RECOMMENDED that the name of the file for that revision be of the form:

module-or-submodule-name ['#' revision-label] ('.yang' / '.yin')

E.g., acme-router-module#2.0.3.yang

YANG module (or submodule) files may be identified using either the revision-date (as per [RFC8407] section 3.2) or the revision label.

3.4.2. Revision label scheme extension statement

The optional "rev:revision-label-scheme" extension statement is used to indicate which revision label scheme a module or submodule uses. There MUST NOT be more than one revision label scheme in a module or submodule. The mandatory argument to this extension statement:

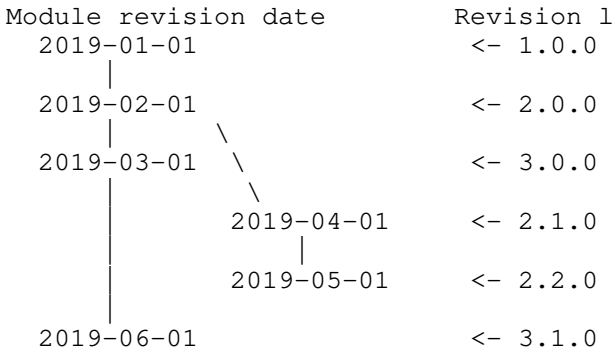
* specifies the revision label scheme used by the module or submodule
* is defined in the document which specifies the revision label scheme
* MUST be an identity derived from "revision-label-scheme-base".

The revision label scheme used by a module or submodule SHOULD NOT change during the lifetime of the module or submodule. If the revision label scheme used by a module or submodule is changed to a new scheme, then all revision label statements that do not conform to the new scheme MUST be replaced or removed.

3.5. Examples for updating the YANG module revision history

The following diagram, explanation, and module history illustrates how the branched revision history, "non-backwards-compatible" extension statement, and revision "label" extension statement could be used:

Example YANG module with branched revision history.



The tree diagram above illustrates how an example module's revision history might evolve, over time. For example, the tree might represent the following changes, listed in chronological order from the oldest revision to the newest revision:

Example module, revision 2019-06-01:

```
module example-module {

  namespace "urn:example:module";
  prefix "prefix-name";
  rev:revision-label-scheme "yangver:yang-semver";

  import ietf-yang-revisions { prefix "rev"; }
  import ietf-yang-semver { prefix "yangver"; }

  description
    "to be completed";

  revision 2019-06-01 {
    rev:label 3.1.0;
    description "Add new functionality.";
  }

  revision 2019-03-01 {
    rev:label 3.0.0;
    rev:non-backwards-compatible;
    description
      "Add new functionality. Remove some deprecated nodes.";
  }

  revision 2019-02-01 {
    rev:label 2.0.0;
    rev:non-backwards-compatible;
    description "Apply bugfix to pattern statement";
  }

  revision 2019-01-01 {
    rev:label 1.0.0;
    description "Initial revision";
  }

  //YANG module definition starts here
}
```

Example module, revision 2019-05-01:

```
module example-module {

  namespace "urn:example:module";
  prefix "prefix-name";
  rev:revision-label-scheme "yangver:yang-semver";

  import ietf-yang-revisions { prefix "rev"; }
  import ietf-yang-semver { prefix "yangver"; }

  description
    "to be completed";

  revision 2019-05-01 {
    rev:label 2.2.0;
    description "Backwards-compatible bugfix to enhancement.";
  }

  revision 2019-04-01 {
    rev:label 2.1.0;
    description "Apply enhancement to older release train.";
  }

  revision 2019-02-01 {
    rev:label 2.0.0;
    rev:non-backwards-compatible;
    description "Apply bugfix to pattern statement";
  }

  revision 2019-01-01 {
    rev:label 1.0.0;
    description "Initial revision";
  }

  //YANG module definition starts here
}
```

4. Guidance for revision selection on imports

[RFC7950] and [RFC6020] allow YANG module "import" statements to optionally require the imported module to have a specific revision date. In practice, importing a module with an exact revision date can be too restrictive because it requires the importing module to be updated whenever any change to the imported module occurs, and hence section Section 7.1 suggests that authors do not restrict YANG module imports to exact revision dates.

Instead, for conformance purposes (section 5.6 of [RFC7950]), the recommended approach for defining the relationship between specific YANG module revisions is to specify the relationships outside of the YANG modules, e.g., via YANG library [RFC8525], YANG packages [I-D.ietf-netmod-yang-packages], a filesystem directory containing a set of consistent YANG module revisions, or a revision control system commit label.

4.1. Recommending a minimum revision for module imports

Although the previous section indicates that the actual relationship constraints between different revisions of YANG modules should be specified outside of the modules, in some scenarios YANG modules are designed to be loosely coupled, and implementors may wish to select sets of YANG module revisions that are expected to work together. For these cases it can be helpful for a module author to provide guidance on a recommended minimum revision that is expected to satisfy an YANG import. E.g., the module author may know of a dependency on a type or grouping that has been introduced in a particular imported YANG module revision. Although there can be no guarantee that all derived future revisions from the particular imported module will necessarily also be compatible, older revisions of the particular imported module are very unlikely to ever be compatible.

This document introduces a new YANG extension statement to provide guidance to module implementors on a recommended minimum module revision of an imported module that is anticipated to be compatible. This statement has been designed to be machine-readable so that tools can parse the minimum revision extension statement and generate warnings if appropriate, but this extension statement does not alter YANG module conformance of valid YANG module versions in any way, and specifically it does not alter the behavior of the YANG module import statement from that specified in [RFC7950].

The ietf-revisions module defines the "recommended-min" extension statement, a substatement to the YANG "import" statement, to allow for a "minimum recommended revision" to be documented:

> The argument to the "recommended-min" extension statement is a revision date or a revision label.

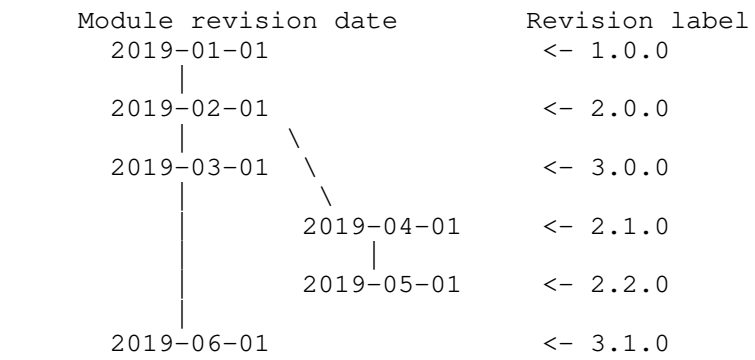
A particular revision of an imported module adheres to an import's "recommended-min" extension statement if the imported module's revision history contains a revision statement with a matching revision date or revision label. Removing entries from a module's revision history may cause a particular revision to no longer satisfy an import's "recommended-min" statement if the revision-date or label is no longer present in the module's revision history; further described in Section 3.3 and Section 7.1.

The "recommended-min" extension statement MAY be specified multiple times, allowing a set of recommended minimum revisions to be documented. Module implementors are recommended to pick a module revision that adheres to any of the "recommended-min" statements.

Adding, modifying or removing a "recommended-min" extension statement is a BC change.

4.1.1. Module import examples

Consider the example module "example-module" from Section 3.5 that is hypothetically available in the following revision/label pairings: 2019-01-01/1.0.0, 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0. The relationship between the revisions is as before:



4.1.1.1. Example 1

This example recommends module revisions for import that match, or are derived from the revision 2019-02-01. E.g., this dependency might be used if there was a new container added in revision 2019-02-01 that is augmented by the importing module. It includes revisions/labels: 2019-02-01/2.0.0, 2019-03-01/3.0.0, 2019-04-01/2.1.0, 2019-05-01/2.2.0 and 2019-06-01/3.1.0.

```
import example-module {
  rev:recommended-min 2019-02-01;
}
```

Alternatively, the first example could have used the revision label "2.0.0" instead, which selects the same set of revisions/labels.

```
import example-module {
  rev:recommended-min 2.0.0;
}
```

4.1.1.2. Example 2

This example recommends module revisions for import that are derived from 2019-04-01 by using the revision label 2.1.0. It includes revisions/labels: 2019-04-01/2.1.0 and 2019-05-01/2.2.0. Even though 2019-06-01/3.1.0 has a higher revision label number than 2019-04-01/2.1.0 it is not a derived revision, and hence it is not a recommended revision for import.

```
import example-module {
  rev:recommended-min 2.1.0;
}
```

4.1.1.3. Example 3

This example recommends module revisions for import that are derived from either 2019-04-01 or 2019-06-01. It includes revisions/labels: 2019-04-01/2.1.0, 2019-05-01/2.2.0, and 2019-06-01/3.1.0.

```
import example-module {
  rev:recommended-min 2019-04-01;
  rev:recommended-min 2019-06-01;
}
```

5. Updates to ietf-yang-library

This document updates YANG 1.1 [RFC7950] and YANG library [RFC8525] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yang-library-revisions, that augments YANG library [RFC8525] with revision labels and two leafs to indicate how a server implements deprecated and obsolete schema nodes.

5.1. Resolving ambiguous module imports

A YANG datastore schema, defined in [RFC8525], can specify multiple revisions of a YANG module in the schema using the "import-only" list, with the requirement from [RFC7950] section 5.6.5 that only a single revision of a YANG module may be implemented.

If a YANG module import statement does not specify a specific revision within the datastore schema then it could be ambiguous as to which module revision the import statement should resolve to. Hence, a datastore schema constructed by a client using the information contained in YANG library may not exactly match the datastore schema actually used by the server.

The following two rules remove the ambiguity:

If a module import statement could resolve to more than one module revision defined in the datastore schema, and one of those revisions is implemented (i.e., not an "import-only" module), then the import statement MUST resolve to the revision of the module that is defined as being implemented by the datastore schema.

If a module import statement could resolve to more than one module revision defined in the datastore schema, and none of those revisions are implemented, then the import MUST resolve to the module revision with the latest revision date.

5.2. YANG library versioning augmentations

The "ietf-yang-library-revisions" YANG module has the following structure (using the notation defined in [RFC8340]):

```
module: ietf-yang-library-revisions
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:module-set/yanglib:module
            /yanglib:submodule:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:module-set
            /yanglib:import-only-module/yanglib:submodule:
    +--ro revision-label?   rev:revision-label
  augment /yanglib:yang-library/yanglib:schema:
    +--ro deprecated-nodes-implemented?   boolean
    +--ro obsolete-nodes-absent?          boolean
```

5.2.1. Advertising revision-label

The ietf-yang-library-revisions YANG module augments the "module" and "submodule" lists in ietf-yang-library with "revision-label" leafs to optionally declare the revision label associated with each module and submodule.

5.2.2. Reporting how deprecated and obsolete nodes are handled

The ietf-yang-library-revisions YANG module augments YANG library with two boolean leafs to allow a server to report how it implements status "deprecated" and status "obsolete" schema nodes. The leafs are:

deprecated-nodes-implemented: If set to "true", this leaf indicates that all schema nodes with a status "deprecated" are implemented equivalently as if they had status "current"; otherwise deviations MUST be used to explicitly remove "deprecated" nodes from the schema. If this leaf is set to "false" or absent, then the behavior is unspecified.

obsolete-nodes-absent: If set to "true", this leaf indicates that the server does not implement any status "obsolete" schema nodes. If this leaf is set to "false" or absent, then the behaviour is unspecified.

Servers SHOULD set both the "deprecated-nodes-implemented" and "obsolete-nodes-absent" leafs to "true".

If a server does not set the "deprecated-nodes-implemented" leaf to "true", then clients MUST NOT rely solely on the "rev:non-backwards-compatible" statements to determine whether two module revisions are backwards-compatible, and MUST also consider whether the status of any nodes has changed to "deprecated" and whether those nodes are implemented by the server.

6. Versioning of YANG instance data

Instance data sets [RFC9195] do not directly make use of the updated revision handling rules described in this document, as compatibility for instance data is undefined.

However, instance data specifies the content-schema of the data-set. This schema SHOULD make use of versioning using revision dates and/or revision labels for the individual YANG modules that comprise the schema or potentially for the entire schema itself (e.g., [I-D.ietf-netmod-yang-packages]).

In this way, the versioning of a content-schema associated with an instance data set may help a client to determine whether the instance data could also be used in conjunction with other revisions of the YANG schema, or other revisions of the modules that define the schema.

7. Guidelines for using the YANG module update rules

The following text updates section 4.7 of [RFC8407] to revise the guidelines for updating YANG modules.

7.1. Guidelines for YANG module authors

All IETF YANG modules MUST include revision label statements for all newly published YANG modules, and all newly published revisions of existing YANG modules. The revision label MUST take the form of a YANG semantic version number [I-D.ietf-netmod-yang-semver].

NBC changes to YANG modules may cause problems to clients, who are consumers of YANG models, and hence YANG module authors SHOULD minimize NBC changes and keep changes BC whenever possible.

When NBC changes are introduced, consideration should be given to the impact on clients and YANG module authors SHOULD try to mitigate that impact.

A "rev:non-backwards-compatible" statement MUST be added if there are NBC changes relative to the previous revision.

Removing old revision statements from a module's revision history could break import by revision, and hence it is RECOMMENDED to retain them. If all dependencies have been updated to not import specific revisions of a module, then the corresponding revision statements can be removed from that module. An alternative solution, if the revision section is too long, would be to remove, or curtail, the older description statements associated with the previous revisions.

The "rev:recommended-min" extension MAY be used in YANG module imports to indicate revision dependencies between modules in preference to the "revision-date" statement, which causes overly strict import dependencies and SHOULD NOT be used.

A module that includes submodules SHOULD use the "revision-date" statement to include specific submodule revisions. The revision of the including module MUST be updated when any included submodule has changed.

In some cases a module or submodule revision that is not strictly NBC by the definition in Section 3.1.2 of this specification may include the "non-backwards-compatible" statement. Here is an example when adding the statement may be desirable:

* A "config false" leaf had its value space expanded (for example, a range was increased, or additional enum values were added) and the author or server implementor feels there is a significant compatibility impact for clients and users of the module or submodule

7.1.1. Making non-backwards-compatible changes to a YANG module

There are various valid situations where a YANG module has to be modified in an NBC way. Here are some guidelines on how non-backwards-compatible changes can be made incrementally, with the assumption that deprecated nodes are implemented by the server, and obsolete nodes are not:

1. The changes should be made gradually, e.g., a data node's status SHOULD NOT be changed directly from "current" to "obsolete" (see Section 4.7 of [RFC8407]), instead the status SHOULD first be marked "deprecated". At some point in the future, when support is removed for the data node, there are two options. The first, and preferred, option is to keep the data node definition in the model and change the status to obsolete. The second option is to simply remove the data node from the model, but this has the risk of breaking modules which import the modified module, and the removed identifier may be accidently reused in a future revision.

2. For deprecated data nodes the "description" statement SHOULD also indicate until when support for the node is guaranteed (if known). If there is a replacement data node, rpc, action or notification for the deprecated node, this SHOULD be stated in the "description". The reason for deprecating the node can also be included in the "description" if it is deemed to be of potential interest to the user.

3. For obsolete data nodes, it is RECOMMENDED to keep the above information, from when the node had status "deprecated", which is still relevant.

4. When obsoleting or deprecating data nodes, the "deprecated" or "obsolete" status SHOULD be applied at the highest possible level in the data tree. For clarity, the "status" statement SHOULD also be applied to all descendent data nodes, but the additional status related information does not need to be repeated if it does not introduce any additional information.

5. NBC changes which can break imports SHOULD be avoided because of the impact on the importing module. The importing modules could get broken, e.g., if an augmented node in the importing module has been removed from the imported module. Alternatively, the schema of the importing modules could undergo an NBC change due to the NBC change in the imported module, e.g., if a node in a grouping has been removed. As described in Appendix B.1, instead of removing a node, that node SHOULD first be deprecated and then obsoleted.

See Appendix B for examples on how NBC changes can be made.

7.2. Versioning Considerations for Clients

Guidelines for clients of modules using the new module revision update procedure:

* Clients SHOULD be liberal when processing data received from a server. For example, the server may have increased the range of an operational node causing the client to receive a value which is outside the range of the YANG model revision it was coded against.

* Clients SHOULD monitor changes to published YANG modules through their revision history, and use appropriate tooling to understand the specific changes between module revision. In particular, clients SHOULD NOT migrate to NBC revisions of a module without understanding any potential impact of the specific NBC changes.

* Clients SHOULD plan to make changes to match published status changes. When a node’s status changes from "current" to "deprecated", clients SHOULD plan to stop using that node in a timely fashion. When a node’s status changes to "obsolete", clients MUST stop using that node.

8. Module Versioning Extension YANG Modules

YANG module with extension statements for annotating NBC changes, revision label, revision label scheme, and importing by revision.

```
<CODE BEGINS> file "ietf-yang-revisions@2022-11-29.yang"
module ietf-yang-revisions {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-revisions";
  prefix rev;

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>

     Author:   Reshad Rahman
               <mailto:reshad@yahoo.com>

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>

     Author:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>

     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>";
  description
    "This YANG 1.1 module contains definitions and extensions to
     support updated YANG revision handling.

     Copyright (c) 2002 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 Revised 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).

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

     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 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.";
```

```
  // RFC Ed.: update the date below with the date of RFC publication
  // and remove this note.
  // RFC Ed.: replace XXXX (inc above) with actual RFC number and
  // remove this note.

  revision 2022-11-29 {
    rev:label "1.0.0-draft-ietf-netmod-yang-module-versioning-08";
    description
      "Initial version.";
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }

  typedef revision-date {
    type string {
      pattern '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|[1-2][0-9]|3[0-1])';
    }
    description
      "A date associated with a YANG revision.

       Matches dates formatted as YYYY-MM-DD.";
    reference
      "RFC 7950: The YANG 1.1 Data Modeling Language";
  }

  typedef revision-label {
    type string {
      length "1..255";
      pattern '[a-zA-Z0-9,\-_.+]+';
      pattern '[0-9]{4}-[0-9]{2}-[0-9]{2}' {
        modifier "invert-match";
        error-message
          "The revision-label must not match a revision-date.";
      }
    }
    description
      "A label associated with a YANG revision.

       Alphanumeric characters, comma, hyphen, underscore, period
       and plus are the only accepted characters. MUST NOT match
       revision-date or pattern similar to a date.";
    reference
      "XXXX: Updated YANG Module Revision Handling;
       Section 3.3, Revision label";
  }

  typedef revision-date-or-label {
    type union {
```

```
        type revision-date;
        type revision-label;
      }
      description
        "Represents either a YANG revision date or a revision label";
    }

    extension non-backwards-compatible {
      description
        "This statement is used to indicate YANG module revisions that
         contain non-backwards-compatible changes.

         The statement MUST only be a substatement of the 'revision'
         statement.  Zero or one 'non-backwards-compatible' statements
         per parent statement is allowed.  No substatements for this
         extension have been standardized.

         If a revision of a YANG module contains changes, relative to
         the preceding revision in the revision history, that do not
         conform to the backwards-compatible module update rules
         defined in RFC-XXX, then the 'non-backwards-compatible'
         statement MUST be added as a substatement to the revision
         statement.

         Conversely, if a revision does not contain a
         'non-backwards-compatible' statement then all changes,
         relative to the preceding revision in the revision history,
         MUST be backwards-compatible.

         A new module revision that only contains changes that are
         backwards-compatible SHOULD NOT include the
         'non-backwards-compatible' statement.  An example of when an
         author might add the 'non-backwards-compatible' statement is
         if they believe a change could negatively impact clients even
         though the backwards compatibility rules defined in RFC-XXXX
         classify it as a backwards-compatible change.

         Add, removing, or changing a 'non-backwards-compatible'
         statement is a backwards-compatible version change.";
      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.2,
         non-backwards-compatible revision extension statement";
    }

    extension label {
      argument revision-label;
      description
```

```
        "The revision label can be used to provide an additional
         versioning identifier associated with a module or submodule
         revision.  One such scheme that could be used is [XXXX:
         ietf-netmod-yang-semver].

         The format of the revision label argument MUST conform to the
         pattern defined for the revision label typedef in this module.

         The statement MUST only be a substatement of the revision
         statement.  Zero or one revision label statements per parent
         statement are allowed.  No substatements for this extension
         have been standardized.

         Revision labels MUST be unique amongst all revisions of a
         module or submodule.

         Adding a revision label is a backwards-compatible version
         change.  Changing or removing an existing revision label in
         the revision history is a non-backwards-compatible version
         change, because it could impact any references to that
         revision label.";
      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3, Revision label";
    }

    extension revision-label-scheme {
      argument revision-label-scheme-base;
      description
        "The revision label scheme specifies which revision label
         scheme the module or submodule uses.

         The mandatory revision-label-scheme-base argument MUST be an
         identity derived from revision-label-scheme-base.

         This extension is only valid as a top-level statement, i.e.,
         given as as a substatement to 'module' or 'submodule'.  No
         substatements for this extension have been standardized.

         This extension MUST be used if there is a revision label
         statement in the module or submodule.

         Adding a revision label scheme is a backwards-compatible
         version change.  Changing a revision label scheme is a
         non-backwards-compatible version change, unless the new
         revision label scheme is backwards-compatible with the
         replaced revision label scheme.  Removing a revision label
         scheme is a non-backwards-compatible version change.";
```

```
      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3.1, Revision label scheme extension statement";
    }

    extension recommended-min {
      argument revision-date-or-label;
      description
        "Recommends the revision of the module that may be imported to
         one that matches or is derived from the specified
         revision-date or revision label.

         The argument value MUST conform to the
         'revision-date-or-label' defined type.

         The statement MUST only be a substatement of the import
         statement.  Zero, one or more 'recommended-min' statements per
         parent statement are allowed.  No substatements for this
         extension have been standardized.

         If specified multiple times, then any module revision that
         satisfies at least one of the 'recommended-min' statements is
         an acceptable recommended revision for import.

         A particular revision of an imported module adheres to an
         import's 'recommended-min' extension statement if the imported
         module's revision history contains a revision statement with a
         matching revision date or revision label.

         Adding, removing or updating a 'recommended-min' statement to
         an import is a backwards-compatible change.";
      reference
        "XXXX: Updated YANG Module Revision Handling; Section 4,
         Recommending a minimum revision for module imports";
    }

    identity revision-label-scheme-base {
      description
        "Base identity from which all revision label schemes are
         derived.";
      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3.1, Revision label scheme extension statement";
    }
  }
  <CODE ENDS>
```

YANG module with augmentations to YANG Library to revision labels

```
<CODE BEGINS> file "ietf-yang-library-revisions@2021-11-04.yang"
module ietf-yang-library-revisions {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions";
  prefix yl-rev;

  import ietf-yang-revisions {
    prefix rev;
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }
  import ietf-yang-library {
    prefix yanglib;
    reference
      "RFC 8525: YANG Library";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>

     Author:   Reshad Rahman
               <mailto:reshad@yahoo.com>

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>

     Author:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>

     Author:   Jason Sterne
               <mailto:jason.sterne@nokia.com>";
  description
    "This module contains augmentations to YANG Library to add module
     level revision label and to provide an indication of how
     deprecated and obsolete nodes are handled by the server.

     Copyright (c) 2002 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 Revised 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).

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

        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 (RFC 2119) (RFC 8174) when, and only when,
        they appear in all capitals, as shown here.";

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX (including in the imports above) with
    // actual RFC number and remove this note.
    // RFC Ed.: please replace label version with 1.0.0 and
    // remove this note.

    revision 2021-11-04 {
      rev:label "1.0.0-draft-ietf-netmod-yang-module-versioning-05";
      description
        "Initial revision";
      reference
        "XXXX: Updated YANG Module Revision Handling";
    }

    // library 1.0 modules-state is not augmented with revision-label

    augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
      description
        "Add a revision label to module information";
      leaf revision-label {
        type rev:revision-label;
        description
          "The revision label associated with this module revision.
           The label MUST match the revision label value in the
           specific revision of the module loaded in this module-set.";
        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 5.2.1, Advertising revision-label";
      }
    }

    augment
      "/yanglib:yang-library/yanglib:module-set/yanglib:module/"
```

```
     + "yanglib:submodule" {
       description
         "Add a revision label to submodule information";
       leaf revision-label {
         type rev:revision-label;
         description
           "The revision label associated with this submodule revision.
            The label MUST match the revision label value in the
            specific revision of the submodule included by the module
            loaded in this module-set.";
         reference
           "XXXX: Updated YANG Module Revision Handling;
            Section 5.2.1, Advertising revision-label";
       }
     }

     augment "/yanglib:yang-library/yanglib:module-set/"
           + "yanglib:import-only-module" {
       description
         "Add a revision label to module information";
       leaf revision-label {
         type rev:revision-label;
         description
           "The revision label associated with this module revision.
            The label MUST match the revision label value in the
            specific revision of the module included in this
            module-set.";
         reference
           "XXXX: Updated YANG Module Revision Handling;
            Section 5.2.1, Advertising revision-label";
       }
     }

     augment "/yanglib:yang-library/yanglib:module-set/"
           + "yanglib:import-only-module/yanglib:submodule" {
       description
         "Add a revision label to submodule information";
       leaf revision-label {
         type rev:revision-label;
         description
           "The revision label associated with this submodule revision.
            The label MUST match the rev:label value in the specific
            revision of the submodule included by the import-only-module
            loaded in this module-set.";
         reference
           "XXXX: Updated YANG Module Revision Handling;
            Section 5.2.1, Advertising revision-label";
       }
```

```
    }

    augment "/yanglib:yang-library/yanglib:schema" {
      description
        "Augmentations to the ietf-yang-library module to indicate how
         deprecated and obsoleted nodes are handled for each datastore
         schema supported by the server.";
      leaf deprecated-nodes-implemented {
        type boolean;
        description
          "If set to true, this leaf indicates that all schema nodes
           with a status 'deprecated' are implemented equivalently as
           if they had status 'current'; otherwise deviations MUST be
           used to explicitly remove deprecated nodes from the schema.
           If this leaf is absent or set to false, then the behavior is
           unspecified.";
        reference
          "XXXX: Updated YANG Module Revision Handling;
           Section 5.2.2, Reporting how deprecated and obsolete nodes
           are handled";
      }
      leaf obsolete-nodes-absent {
        type boolean;
        description
          "If set to true, this leaf indicates that the server does not
           implement any status 'obsolete' schema nodes.  If this leaf
           is absent or set to false, then the behaviour is
           unspecified.";
        reference
          "XXXX: Updated YANG Module Revision Handling; Section 5.2.2,
           Reporting how deprecated and obsolete nodes are handled";
      }
    }
  }
  <CODE ENDS>
```

9. Security considerations

9.1. Security considerations for module revisions

As discussed in the introduction of this document, YANG modules occasionally undergo changes that are not backwards compatible. This occurs in both standards and vendor YANG modules despite the prohibitions in RFC 7950. RFC 7950 also allows nodes to change to status 'obsolete' which can change behavior and compatibility for a client.

The fact that YANG modules change in a non-backwards-compatible manner may have security implications. Such changes should be carefully considered, including the scenarios described below. The rev:non-backwards-compatible extension statement introduced in this document provides an alert that the module or submodule may contain changes that impact users and need to be examined more closely for both compatibility and potential security implications. Flagging the change reduces the risk of introducing silent exploitable vulnerabilities.

When a module undergoes a non-backwards-compatible change, a server may implement different semantics for a given leaf than a client using an older version of the module is expecting. If the particular leaf controls any security functions of the device, or is related to parts of the configuration or state that are sensitive from a security point of view, then the difference in behavior between the old and new revisions needs to be considered carefully. In particular, changes to the default of the leaf should be examined.

Implementors and users should also consider impact to data node access control rules (e.g. The Network Configuration Access Control Model (NACM) [RFC8341]) in the face of non-backwards-compatible changes. Access rules may need to be adjusted when a new module revision is introduced that contains a non-backwards-compatible change.

If the changes to a module or submodule have security implications, it is recommended to highlight those implications in the description of the revision statement.

9.2. Security considerations for the modules defined in this document

The YANG module 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 [RFC8446].

The NETCONF 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.

This document does not define any new protocol or data nodes that are writable.

This document updates YANG Library [RFC8525] with augmentations to include revision labels in the YANG library data and two boolean leafs to indicate whether status deprecated and status obsolete schema nodes are implemented by the server. These read-only augmentations do not add any new security considerations beyond those already present in [RFC8525].

10. IANA Considerations

10.1. YANG Module Registrations

This document requests IANA to registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

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

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

The following YANG module is requested to be registred in the "IANA Module Names" [RFC6020]. Following the format in RFC 6020, the following registrations are requested:

The ietf-yang-revisions module:

Name: ietf-yang-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-revisions

Prefix: rev

Reference: [RFCXXXX]

The ietf-yang-library-revisions module:

Name: ietf-yang-library-revisions

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-library-revisions

Prefix: yl-rev

Reference: [RFCXXXX]

10.2. Guidance for versioning in IANA maintained YANG modules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning YANG modules that are derived from other IANA registries. For example, "iana-if-type.yang" [IfTypeYang] is derived from the "Interface Types (ifType) IANA registry" [IfTypesReg], and "iana-routing-types.yang" [RoutingTypesYang] is derived from the "Address Family Numbers" [AddrFamilyReg] and "Subsequent Address Family Identifiers (SAFI) Parameters" [SAFIReg] IANA registries.

Normally, updates to the registries cause any derived YANG modules to be updated in a backwards-compatible way, but there are some cases where the registry updates can cause non-backward-compatible updates to the derived YANG module. An example of such an update is the 2020-12-31 revision of iana-routing-types.yang [RoutingTypesDecRevision], where the enum name for two SAFI values was changed.

In all cases, IANA MUST follow the versioning guidance specified in Section 3.1, and MUST include a "rev:non-backwards-compatible" substatement to the latest revision statement whenever an IANA maintained module is updated in a non-backwards-compatible way, as described in Section 3.2.

Note: For published IANA maintained YANG modules that contain non-backwards-compatible changes between revisions, a new revision should be published with the "rev:non-backwards-compatible" substatement retrospectively added to any revisions containing non-backwards-compatible changes.

Non-normative examples of updates to enumeration types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an enumeration typedef to obsolete, changing the status of an enum entry to obsolete, removing an enum entry, changing the identifier of an enum entry, or changing the described meaning of an enum entry.

Non-normative examples of updates to enumeration types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new enum entry to the end of the enumeration, changing the status or an enum entry to deprecated, or improving the description of an enumeration that does not change its defined meaning.

Non-normative examples of updates to identity types in IANA maintained modules that would be classified as non-backwards-compatible changes are: Changing the status of an identity to obsolete, removing an identity, renaming an identity, or changing the described meaning of an identity.

Non-normative examples of updates to identity types in IANA maintained modules that would be classified as backwards-compatible changes are: Adding a new identity, changing the status or an identity to deprecated, or improving the description of an identity that does not change its defined meaning.

11. References

11.1. Normative References

[I-D.ietf-netmod-yang-semver]
Clarke, J., Wilton, R., Rahman, R., Lengyel, B., Sterne, J., and B. Claise, "YANG Semantic Versioning", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-semver-12, 2 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-semver-12>.

[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>.

[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>.

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

11.2. Informative References

[AddrFamilyReg]
"Address Family Numbers IANA Registry", <https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>.

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://datatracker.ietf.org/doc/html/draft-clacla-netmod-yang-model-update-06>.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-

ietf-netmod-yang-packages-03, 4 March 2022,
<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-packages-03>.

[I-D.ietf-netmod-yang-schema-comparison]
Andersson, P. and R. Wilton, "YANG Schema Comparison", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-schema-comparison-02, 14 March 2023,
<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-schema-comparison-02>.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Schema Selection", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-ver-selection-00, 17 March 2020,
<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-ver-selection-00>.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-versioning-reqs-08, 26 June 2023,
<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-versioning-reqs-08>.

[IfTypesReg]
"Interface Types (ifType) IANA Registry",
<https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-5>.

[IfTypeYang]
"iana-if-type YANG Module",
<https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

[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>.

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/info/rfc9195>.

[RoutingTypesDecRevision]
"2020-12-31 revision of iana-routing-types.yang",
<https://www.iana.org/assignments/yang-parameters/iana-routing-types@2020-12-31.yang>.

[RoutingTypesYang]
"iana-routing-types YANG Module",
<https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

[SAFIReg] "Subsequent Address Family Identifiers (SAFI) Parameters IANA Registry", <https://www.iana.org/assignments/safi-namespace/safi-namespace.xhtml>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Appendix A. Examples of changes that are NBC

Examples of NBC changes include:

* Deleting a data node, or changing it to status obsolete.

* Changing the name, type, or units of a data node.

* Modifying the description in a way that changes the semantic meaning of the data node.

* Any changes that remove any previously allowed values from the allowed value set of the data node, either through changes in the type definition, or the addition or changes to "must" statements, or changes in the description.

* Adding or modifying "when" statements that reduce when the data node is available in the schema.

* Making the statement conditional on if-feature.

Appendix B. Examples of applying the NBC change guidelines

The following sections give steps that could be taken for making NBC changes to a YANG module or submodule using the incremental approach described in section Section 7.1.1.

The examples are all for "config true" nodes.

B.1. Removing a data node

Removing a leaf or container from the data tree, e.g., because support for the corresponding feature is being removed:

1. The schema node's status is changed to "deprecated" and the node is supported for some period of time (e.g. one year). This is a BC change.

2. When the schema node is not supported anymore, its status is changed to "obsolete" and the "description" updated. This is an NBC change.

B.2. Changing the type of a leaf node

Changing the type of a leaf node. e.g., a "vpn-id" node of type integer being changed to a string:

1. The status of schema node "vpn-id" is changed to "deprecated" and the node is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate that vpn-name is replacing this node.

2. A new schema node, e.g., "vpn-name", of type string is added to the same location as the existing node "vpn-id". This new node has status "current" and its description explains that it is replacing node "vpn-id".

3. During the period of time when both schema nodes are supported, the interactions between the two nodes is outside the scope of this document and will vary on a case by case basis. One possible option is to have the server prevent the new node from being set if the old node is already set (and vice-versa). The new node could have a "when" statement added to it to achieve this. The old node, however, must not have a "when" statement added, or an existing "when" modified to be more restrictive, since this would be an NBC change. In any case, the server could reject the old node from being set if the new node is already set.

4. When the schema node "vpn-id" is not supported anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

B.3. Reducing the range of a leaf node

Reducing the range of values of a leaf-node, e.g., consider a "vpn-id" schema node of type uint32 being changed from range 1..5000 to range 1..2000:

1. If all values which are being removed were never supported, e.g., if a vpn-id of 2001 or higher was never accepted, this is a BC change for the functionality (no functionality change). Even if it is an NBC change for the YANG model, there should be no impact for clients using that YANG model.

2. If one or more values being removed was previously supported, e.g., if a vpn-id of 3333 was accepted previously, this is an NBC change for the YANG model. Clients using the old YANG model will be impacted, so a change of this nature should be done carefully, e.g., by using the steps described in Appendix B.2

B.4. Changing the key of a list

Changing the key of a list has a big impact to the client. For example, consider a "sessions" list which has a key "interface" and there is a need to change the key to "dest-address". Such a change can be done in steps:

1. The status of list "sessions" is changed to "deprecated" and the list is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate the new list that is replacing this list.

2. A new list is created in the same location with the same descendant schema nodes but with "dest-address" as key. Finding an appropriate name for the new list can be difficult. In this case the new list is called "sessions-address", has status "current" and its description should explain that it is replacing list "session".

3. During the period of time when both lists are supported, the interactions between the two lists is outside the scope of this document and will vary on a case by case basis. One possible option is to have the server prevent entries in the new list from being created if the old list already has entries (and vice-versa).

4. When list "sessions" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

B.5. Renaming a node

A leaf or container schema node may be renamed, either due to a spelling error in the previous name or because of a better name. For example a node "ip-adress" could be renamed to "ip-address":

1. The status of the existing node "ip-adress" is changed to "deprecated" and is supported for some period of time (e.g. one year). This is a BC change. The description is updated to indicate the node that is replacing this node.

2. The new schema node "ip-address" is added to the same location as the existing node "ip-adress". This new node has status "current" and its description should explain that it is replacing node "ip-adress".

3. During the period of time when both nodes are available, the interactions between the two nodes is outside the scope of this document and will vary on a case by case basis. One possible option is to have the server prevent the new node from being set if the old node is already set (and vice-versa). The new node could have a "when" statement added to it to achieve this. The old node, however, must not have a "when" statement added, or an existing "when" modified to be more restrictive, since this would be an NBC change. In any case, the server could reject the old node from being set if the new node is already set.

4. When node "ip-adress" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

Contributors

The following people made substantial contributions to this document:

Bo Wu
lana.wubo@huawei.com

Jan Lindblad
jlindbla@cisco.com

Acknowledgments

This document grew out of the YANG module versioning design team that started after IETF 101. The authors, contributors and the following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

Benoit Claise
benoit.claise@huawei.com

Ebben Aries
exa@juniper.net

Juergen Schoenwaelder
j.shoenwaelder@jacobs-university.de

Mahesh Jethanandani
mjethanandani@gmail.com

Michael (Wangzitao)
wangzitao@huawei.com

Per Andersson
perander@cisco.com

Qin Wu
bill.wu@huawei.com

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update]. We would like to thank Kevin D'Souza and Benoit Claise for their initial work in this problem space.

Discussions on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like to thank both Anees Shaikh and Rob Shakir for their input into this problem space.

We would also like to thank Lou Berger, Andy Bierman, Martin Bjorklund, Italo Busi, Tom Hill, Scott Mansfield, and Kent Watsen for their contributions and review comments.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.
Email: rwilton@cisco.com

Reshad Rahman (editor)
Equinix
Email: reshad@yahoo.com

Balazs Lengyel (editor)
Ericsson
Email: balazs.lengyel@ericsson.com

Joe Clarke
Cisco Systems, Inc.
Email: jclarke@cisco.com

Jason Sterne
Nokia
Email: jason.sterne@nokia.com

Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: 4 April 2024

J. Clarke, Ed.
R. Wilton, Ed.
Cisco Systems, Inc.
R. Rahman
Graphiant
B. Lengyel
Ericsson
J. Sterne
Nokia
B. Claise
Huawei

2 October 2023

YANG Semantic Versioning
draft-ietf-netmod-yang-semver-12

Abstract

This document specifies a scheme and guidelines for applying an extended set of semantic versioning rules to revisions of YANG artifacts (e.g., modules and packages). Additionally, this document defines an RFCAAAA-compliant revision-label-scheme for this YANG semantic versioning scheme.

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 4 April 2024.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-netmod-yang-module-versioning] puts forth a number of concepts relating to modified rules for updating YANG modules and submodules, a means to signal when a new revision of a module or submodule has non-backwards-compatible (NBC) changes compared to its previous revision, and a scheme that uses the revision history as a lineage for determining from where a specific revision of a YANG module or submodule is derived. Additionally, section 3.4 of [I-D.ietf-netmod-yang-module-versioning] defines a revision-label which can be used as an alias to provide additional context or as a meaningful label to refer to a specific revision.

This document defines a revision-label scheme that uses extended semantic versioning rules [SemVer] for YANG artifacts (i.e., YANG modules, YANG submodules, and YANG packages [I-D.ietf-netmod-yang-packages]) as well as the revision label definition for using this scheme. The goal being to add a human readable revision label that provides compatibility information for the YANG artifact without needing to compare or parse its body. The label and rules defined herein represent the RECOMMENDED revision label scheme for IETF YANG artifacts.

Note that a specific revision of the SemVer 2.0.0 specification is referenced here (from June 19, 2020) to provide an immutable version. This is because the 2.0.0 version of the specification has changed over time without any change to the semantic version itself. In some cases the text has changed in non-backwards-compatible ways.

2. Terminology and Conventions

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.

Additionally, this document uses the following terminology:

* YANG artifact: YANG modules, YANG submodules, and YANG packages [I-D.ietf-netmod-yang-packages] are examples of YANG artifacts for the purposes of this document.

* SemVer: A version string that corresponds to the rules defined in [SemVer] . This specific camel-case notation is the one used by the SemVer 2.0.0 website and used within this document to distinguish between YANG Semver.

* YANG Semver: A revision-label identifier that is consistent with the extended set of semantic versioning rules, based on [SemVer] , defined within this document.

3. YANG Semantic Versioning

This section defines YANG Semantic Versioning, explains how it is used with YANG artifacts, and describes the rules associated with changing an artifact's semantic version when its contents are updated.

3.1. Relationship Between SemVer and YANG Semver

[SemVer] is completely compatible with YANG Semver in that a SemVer semantic version number is legal according to the YANG Semver rules (though the inverse is not necessarily true). YANG Semver is a superset of the SemVer rules, and allow for limited branching within YANG artifacts. If no branching occurs within a YANG artifact (i.e., you do not use the compatibility modifiers described below), the YANG Semver version label will appear as a SemVer version number.

3.2. YANG Semver Pattern

YANG artifacts that employ semantic versioning as defined in this document MUST use a version string (e.g., in revision-label or as a package version) that corresponds to the following pattern: 'X.Y.Z_COMPAT'. Where:

* X, Y and Z are mandatory non-negative integers that are each less than or equal to 2147483647 (i.e., the maximum signed 32-bit integer value) and MUST NOT contain leading zeroes,

* The '.' is a literal period (ASCII character 0x2e),

* The '_' is an optional single literal underscore (ASCII character 0x5f) and MUST only be present if the following COMPAT element is included,

* COMPAT, if specified, MUST be either the literal string "compatible" or the literal string "non_compatible".

Additionally, [SemVer] defines two specific types of metadata that may be appended to a semantic version string. Pre-release metadata MAY be appended to a YANG Semver string after a trailing '-' character. Build metadata MAY be appended after a trailing '+' character. If both pre-release and build metadata are present, then build metadata MUST follow pre-release metadata. While build metadata MUST be ignored when comparing YANG semantic versions, pre-

release metadata MUST be used during module and submodule development as specified in Section 5 . Both pre-release and build metadata are allowed in order to support all the [SemVer] rules. Thus, a version lineage that follows strict [SemVer] rules is allowed for a YANG artifact.

To signal the use of this versioning scheme, modules and submodules MUST set the revision-label-scheme extension, as defined in [I-D.ietf-netmod-yang-module-versioning] , to the identity "yang-semver". That identity value is defined in the ietf-yang-semver module below.

Additionally, this ietf-yang-semver module defines a typedef that formally specifies the syntax of the YANG Semver.

3.3. Semantic Versioning Scheme for YANG Artifacts

This document defines the YANG semantic versioning scheme that is used for YANG artifacts that employ the YANG Semver label. The versioning scheme has the following properties:

* The YANG semantic versioning scheme is extended from version 2.0.0 of the semantic versioning scheme defined at semver.org [SemVer] to cover the additional requirements for the management of YANG artifact lifecycles that cannot be addressed using the semver.org 2.0.0 versioning scheme alone.

* Unlike the [SemVer] versioning scheme, the YANG semantic versioning scheme supports updates to older versions of YANG artifacts, to allow for bug fixes and enhancements to artifact versions that are not the latest. However, it does not provide for the unlimited branching and updating of older revisions which are documented by the general rules in [I-D.ietf-netmod-yang-module-versioning] .

* YANG artifacts that follow the [SemVer] versioning scheme are fully compatible with implementations that understand the YANG semantic versioning scheme defined in this document.

* If updates are always restricted to the latest revision of the artifact only, then the version numbers used by the YANG semantic versioning scheme are exactly the same as those defined by the [SemVer] versioning scheme.

Every YANG module and submodule versioned using the YANG semantic versioning scheme specifies the module’s or submodule’s semantic version as the argument to the ’rev:revision-label’ statement.

Because the rules put forth in [I-D.ietf-netmod-yang-module-versioning] are designed to work well with existing versions of YANG and allow for artifact authors to migrate to this scheme, it is not expected that all revisions of a given YANG artifact will have a semantic version label. For example, the first revision of a module or submodule may have been produced before this scheme was available.

YANG packages that make use of this YANG Semver will reflect that in the package metadata.

As stated above, the YANG semantic version is expressed as a string of the form: 'X.Y.Z_COMPAT'.

* 'X' is the MAJOR version. Changes in the MAJOR version number indicate changes that are non-backwards-compatible to versions with a lower MAJOR version number.

* 'Y' is the MINOR version. Changes in the MINOR version number indicate changes that are backwards-compatible to versions with the same MAJOR version number, but a lower MINOR version number and no "_compatible" or "_non_compatible" modifier.

* 'Z' is the PATCH version. Changes in the PATCH version number can indicate an editorial change to the YANG artifact. In conjunction with the '_COMPAT' modifier (see below) changes to 'Z' may indicate a more substantive module change. An editorial change is defined to be a change in the YANG artifact's content that does not affect the semantic meaning or functionality provided by the artifact in any way. Some examples include correcting a spelling mistake in the description of a leaf within a YANG module or submodule, non-significant whitespace changes (e.g., realigning description statements or changing indentation), or changes to YANG comments. Note: restructuring how a module uses, or does not use, submodules is treated as an editorial level change on the condition that there is no change in the module's semantic behavior due to the restructuring.

* '_COMPAT' is an additional modifier, unique to YANG Semver (i.e., not valid in [SemVer]), that indicates backwards-compatible, or non-backwards-compatible changes relative to versions with the same MAJOR and MINOR version numbers, but lower PATCH version number, depending on what form modifier '_COMPAT' takes:

  - If the modifier string is absent, the change represents an editorial change.

- If, however, the modifier string is present, the meaning is described below:

- "_compatible" - the change represents a backwards-compatible change

- "_non_compatible" - the change represents a non-backwards-compatible change

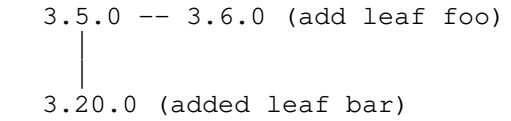
The '_COMPAT' modifier string is "sticky". Once a revision of a module has a modifier in the revision label, then all descendants of that revision with the same X.Y version digits will also have a modifier. The modifier can change from "_compatible" to "_non_compatible" in a descendant revision, but the modifier MUST NOT change from "_non_compatible" to "_compatible" and MUST NOT be removed. The persistence of the "_non_compatible" modifier ensures that comparisons of revision labels do not give the false impression of compatibility between two potentially non-compatible revisions. If "_non_compatible" was removed, for example between revisions "3.3.2_non_compatible" and "3.3.3" (where "3.3.3" was simply an editorial change), then comparing revision labels of "3.3.3" back to an ancestor "3.0.0" would look like they are backwards compatible when they are not (since "3.3.2_non_compatible" was in the chain of ancestors and introduced a non-backwards-compatible change).

The YANG artifact name and YANG semantic version uniquely identify a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG semantic version but different content (and in the case of modules and submodules, different revision dates).

There MUST NOT be multiple versions of a YANG artifact that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier strings. E.g., artifact version "1.2.3_non_compatible" MUST NOT be defined if artifact version "1.2.3" has already been defined.

3.3.1. Branching Limitations with YANG Semver

YANG artifacts that use the YANG Semver revision-label scheme MUST ensure that two artifacts with the same MAJOR version number and no _compatible or _non_compatible modifiers are backwards compatible. Therefore, certain branching schemes cannot be used with YANG Semver. For example, the following branched parent-child module relationship using the following YANG Semver revision labels is not supported:



In this case, given only the revision labels 3.6.0 and 3.20.0 without any parent-child relationship information, one would assume that 3.20.0 is backwards compatible with 3.6.0. But in the illegal example above, 3.20.0 is not backwards compatible with 3.6.0 since 3.20.0 does not contain the leaf foo.

Note that this type of branched parent-child relationship, where two revisions have different backwards compatible changes based on the same parent, is allowed in [I-D.ietf-netmod-yang-module-versioning] .

3.3.2. YANG Semver with submodules

YANG Semver MAY be used to version submodules. Submodule version are separate of any version on the including module, but if a submodule has changed, then the version of the including module MUST also be updated.

The rules for determining the version change of a submodule are the same as those defined in Section 3.2 and Section 3.3 as applied to YANG modules, except they only apply to the part of the module schema defined within the submodule's file.

One interesting case is moving definitions from one submodule to another in a way that does not change the resultant schema of the including module. In this case:

1. The including module has editorial changes

2. The submodule with the schema definition removed has non-backwards-compatible changes

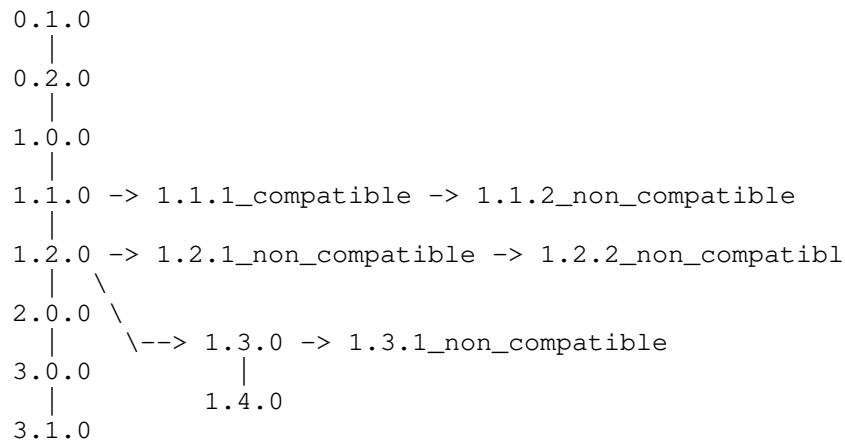
3. The submodule with the schema definitions added has backwards-compatible changes

Note that the meaning of a submodule may change drastically despite having no changes in content or revision due to changes in other submodules belonging to the same module (e.g. groupings and typedefs declared in one submodule and used in another).

3.3.3. Examples for YANG semantic versions

The following diagram and explanation illustrate how YANG semantic versions work.

YANG Semantic versions for an example module:



The tree diagram above illustrates how the version history might evolve for an example module. The tree diagram only shows the parent/child ancestry relationships between the revisions. It does not describe the chronology of the revisions (i.e. when in time each revision was published relative to the other revisions).

The following description lists an example of what the chronological order of the revisions could look like, from oldest revision to newest:

0.1.0 - first pre-release module version

0.2.0 - second pre-release module version (with NBC changes)

1.0.0 - first release (may have NBC changes from 0.2.0)

1.1.0 - added new functionality, leaf "foo" (BC)

1.2.0 - added new functionality, leaf "baz" (BC)

2.0.0 - change existing model for performance reasons, e.g. re-key list (NBC)

1.3.0 - improve existing functionality, added leaf "foo-64" (BC)

1.1.1_compatible - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0. This revision was created after 1.2.0 otherwise it may have been released as 1.2.0. (BC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.3.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

1.2.1_non_compatible - backport NBC fix, rename "baz" to "bar" (NBC)

1.1.2_non_compatible - NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

1.4.0 - introduce new leaf "ghoti" (BC)

3.1.0 - introduce new leaf "wobble" (BC)

1.2.2_non_compatible - backport "wibble". This is a BC change but "non_compatible" modifier is sticky. (BC)

The partial ancestry relationships based on the semantic versioning numbers are as follows:

1.0.0 < 1.1.0 < 1.2.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1_compatible < 1.1.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1_non_compatible < 1.2.2_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 1.3.1_non_compatible

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 1.4.0

There is no ordering relationship between "1.1.1_non_compatible" and either "1.2.0" or "1.2.1_non_compatible", except that they share the common ancestor of "1.1.0".

Looking at the version number alone does not indicate ancestry. The module definition in "2.0.0", for example, does not contain all the contents of "1.3.0". Version "2.0.0" is not derived from "1.3.0".

3.4. YANG Semantic Version Update Rules

When a new revision of an artifact is produced, then the following rules define how the YANG semantic version for the new artifact revision is calculated, based on the changes between the two artifact revisions, and the YANG semantic version of the base artifact revision from which the changes are derived.

The following four rules specify the RECOMMENDED, and REQUIRED minimum, update to a YANG semantic version:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[_compatible|_non_compatible]" SHOULD be updated to "X+1.0.0" unless that version has already been used for this artifact but with different content, in which case the artifact version "X.Y.Z+1_non_compatible" SHOULD be used instead.

2. If an artifact is being updated in a backwards-compatible way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version SHOULD be updated to "X.Y+1.0", unless that version has already been used for this artifact but with different content, when the artifact version SHOULD be updated to "X.Y.Z+1_compatible" instead.

   ii "X.Y.Z_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_non_compatible".

3. If an artifact is being updated in an editorial way, then the next version number depends on the format of the current version number:

   i "X.Y.Z" - the artifact version SHOULD be updated to "X.Y.Z+1"

   ii "X.Y.Z_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version SHOULD be updated to "X.Y.Z+1_non_compatible".

4. YANG artifact semantic version numbers beginning with 0, i.e., "0.X.Y", are regarded as pre-release definitions and need not follow the rules above. Either the MINOR or PATCH version numbers may be updated, regardless of whether the changes are non-backwards-compatible, backwards-compatible, or editorial. See Section 5 for more details on using this notation during module and submodule development.

5. Additional pre-release rules for modules that have had at least one release are specified in Section 5 .

Although artifacts SHOULD be updated according to the rules above, which specify the recommended (and minimum required) update to the version number, the following rules MAY be applied when choosing a new version number:

1. An artifact author MAY update the version number with a more significant update than described by the rules above. For example, an artifact could be given a new MAJOR version number (i.e., X+1.0.0), even though no non-backwards-compatible changes have occurred, or an artifact could be given a new MINOR version number (i.e., X.Y+1.0) even if the changes were only editorial.

2. An artifact author MAY skip version numbers. That is, an artifact's revision history could be 1.0.0, 1.1.0, and 1.3.0 where 1.2.0 is skipped. Note that skipping versions has an impact when importing modules by recommended-min. See Section 4 for more details on importing modules with revision-label version gaps.

Although YANG Semver always indicates when a non-backwards-compatible, or backwards-compatible change may have occurred to a YANG artifact, it does not guarantee that such a change has occurred, or that consumers of that YANG artifact will be impacted by the change. Hence, tooling, e.g., [I-D.ietf-netmod-yang-schema-comparison] , also plays an important role for comparing YANG artifacts and calculating the likely impact from changes.

[I-D.ietf-netmod-yang-module-versioning] defines the "rev:non-backwards-compatible" extension statement to indicate where non-backwards-compatible changes have occurred in the module revision history. If a revision entry in a module's revision history includes the "rev:non-backwards-compatible" statement then that MUST be reflected in any YANG semantic version associated with that revision. However, the reverse does not necessarily hold, i.e., if the MAJOR version has been incremented it does not necessarily mean that a "rev:non-backwards-compatible" statement would be present.

3.5. Examples of the YANG Semver Label

3.5.1. Example Module Using YANG Semver

Below is a sample YANG module that uses the YANG Semver revision-label based on the rules defined in this document.

```
module example-versioned-module {
  yang-version 1.1;
  namespace "urn:example:versioned:module";
  prefix "exvermod";
  rev:revision-label-scheme "ysver:yang-semver";

  import ietf-yang-revisions { prefix "rev"; }
  import ietf-yang-semver { prefix "ysver"; }

  description
    "to be completed";

  revision 2017-08-30 {
    description "Backport 'wibble' leaf";
    rev:revision-label 1.2.2_non_compatible;
  }

  revision 2017-07-30 {
    description "Rename 'baz' to 'bar'";
    rev:revision-label 1.2.1_non_compatible;
    rev:non-backwards-compatible;
  }

  revision 2017-04-20 {
    description "Add new functionality, leaf 'baz'";
    rev:revision-label 1.2.0;
  }

  revision 2017-04-03 {
    description "Add new functionality, leaf 'foo'";
    rev:revision-label 1.1.0;
  }

  revision 2017-02-07 {
    description "First release version.";
    rev:revision-label 1.0.0;
  }

  // Note: YANG Semver rules do not apply to 0.X.Y labels.
  // The following pre-release revision statements would not
  // appear in any final published version of a module. They
  // are removed when the final version is published.
  // During the pre-release phase of development, only a
  // single one of these revision statements would appear

  // revision 2017-01-30 {
  //   description "NBC changes to initial revision";
  //   rev:revision-label 0.2.0;
```

```
      //   rev:non-backwards-compatible; // optional
      //                           // (theoretically no
      //                           // 'previous released version')
      // }

      // revision 2017-01-26 {
      //   description "Initial module version";
      //   rev:revision-label 0.1.0;
      // }

      //YANG module definition starts here
    }
```

3.5.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the YANG Semver revision label based on the rules defined in this document. Note: '\' line wrapping per [RFC8792] .

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-yang-pkg",
    "content-schema": {
      "module": "ietf-yang-packages@2022-03-04"
    },
    "timestamp": "2022-12-06T17:00:38Z",
    "description":  ["Example of a Package  \
       using YANG Semver"],
    "content-data": {
      "ietf-yang-packages:packages": {
        "package": [
          {
            "name": "example-yang-pkg",
            "version": "1.3.1",
            ...
          }
        ]
      }
    }
  }
}
```

Figure 1

4. Import Module by Semantic Version

[I-D.ietf-netmod-yang-module-versioning] allows for imports to be done based on a module or a derived revision of a module. The rev:recommended-min statement can specify either a revision date or a revision label. The YANG Semver revision-label value can be used as the argument to rev:recommended-min . When used as such, any module that contains exactly the same YANG semantic version in its revision history may be used to satisfy the import requirement. For example:

```
import example-module {
  rev:recommended-min 3.0.0;
}
```

Note: the import lookup does not stop when a non-backward-compatible change is encountered. That is, if module B imports a module A at or derived from version 2.0.0, resolving that import will pass through a revision of module A with version "2.1.0_non_compatible" in order to determine if the present instance of module A derives from "2.0.0".

If an import by recommended-min cannot locate the specified revision-label in a given module's revision history, that import will fail. This is noted in the case of version gaps. That is, if a module's history includes "1.0.0", "1.1.0", and "1.3.0", an import from recommended-min at "1.2.0" will be unable to locate the specified revision entry and thus the import cannot be satisfied.

5. Guidelines for Using Semver During Module Development

This section and the IETF-specific sub-section below provides YANG Semver-specific guidelines to consider when developing new YANG modules. As such this section updates [RFC8407] .

Development of a brand new YANG module or submodule outside of the IETF that uses YANG Semver as its revision-label scheme SHOULD begin with a 0 for the MAJOR version component. This allows the module or submodule to disregard strict SemVer rules with respect to non-backwards-compatible changes during its initial development. However, module or submodule developers MAY choose to use the SemVer pre-release syntax instead with a 1 for the MAJOR version component. For example, an initial module or submodule revision-label might be either 0.0.1 or 1.0.0-alpha.1. If the authors choose to use the 0 MAJOR version component scheme, they MAY switch to the pre-release scheme with a MAJOR version component of 1 when the module or submodule is nearing initial release (e.g., a module's or submodule's revision label may transition from 0.3.0 to 1.0.0-beta.1 to indicate it is more mature and ready for testing).

When using pre-release notation, the format MUST include at least one alphabetic component and MUST end with a '.' or '-' and then one or more digits. These alphanumeric components will be used when deciding pre-release precedence. The following are examples of valid pre-release versions:

1.0.0-alpha.1

1.0.0-alpha.3

2.1.0-beta.42

3.0.0-202007.rc.1

When developing a new revision of an existing module or submodule using the YANG Semver revision-label scheme, the intended target semantic version MUST be used along with pre-release notation. For example, if a released module or submodule which has a current revision-label of 1.0.0 is being modified with the intent to make non-backwards-compatible changes, the first development MAJOR version component must be 2 with some pre-release notation such as -alpha.1, making the version 2.0.0-alpha.1. That said, every publicly available release of a module or submodule MUST have a unique YANG Semver revision-label (where a publicly available release is one that could be implemented by a vendor or consumed by an end user). Therefore, it may be prudent to include the year or year and month development began (e.g., 2.0.0-201907-alpha.1). As a module or submodule undergoes development, it is possible that the original intent changes. For example, a 1.0.0 version of a module or submodule that was destined to become 2.0.0 after a development cycle may have had a scope change such that the final version has no non-backwards-compatible changes and becomes 1.1.0 instead. This change is acceptable to make during the development phase so long as pre-release notation is present in both versions (e.g., 2.0.0-alpha.3 becomes 1.1.0-alpha.4). However, on the next development cycle (after 1.1.0 is released), if again the new target release is 2.0.0, new pre-release components must be used such that every revision-label for a given module or submodule MUST be unique throughout its entire lifecycle (e.g., the first pre-release version might be 2.0.0-202005-alpha.1 if keeping the same year and month notation mentioned above).

5.1. Pre-release Version Precedence

As a module or submodule is developed, the scope of the work may change. That is, while a ratified module or submodule with revision-label 1.0.0 is initially intended to become 2.0.0 in its next ratified version, the scope of work may change such that the final version is 1.1.0. During the development cycle, the pre-release versions could move from 2.0.0-some-pre-release-tag to 1.1.0-some-pre-release-tag. This downwards changing of version numbers makes it difficult to evaluate semantic version rules between pre-release versions. However, taken independently, each pre-release version can be compared to the previously ratified version (e.g., 1.1.0-some-pre-release-tag and 2.0.0-some-pre-release-tag can each be compared to 1.0.0). Module and submodule developers SHOULD maintain only one revision statement in a pre-released module or submodule that reflects the latest revision. IETF authors MAY choose to include an appendix in the associated draft to track overall changes to the module or submodule.

5.2. YANG Semver in IETF Modules

All published IETF modules and submodules MUST use YANG semantic versions for their revision-labels.

Development of a new module or submodule within the IETF SHOULD begin with the 0 MAJOR number scheme as described above. When revising an existing IETF module or submodule, the revision-label MUST use the target (i.e., intended) MAJOR and MINOR version components with a 0 PATCH version component. If the intended ratified release will be non-backward-compatible with the current ratified release, the MINOR version component MUST be 0.

5.2.1. Guidelines for IETF Module Development

All IETF modules and submodules in development MUST use the whole document name as a pre-release version string, including the current document revision. For example, if a module or submodule which is currently released at version 1.0.0 is being revised to include non-backwards-compatible changes in draft-user-netmod-foo, its development revision-labels MUST include 2.0.0-draft-user-netmod-foo followed by the document's revision (e.g., 2.0.0-draft-user-netmod-foo-02). This will ensure each pre-release version is unique across the lifecycle of the module or submodule. Even when using the 0 MAJOR version for initial module or submodule development (where MINOR and PATCH can change), appending the draft name as a pre-release component helps to ensure uniqueness when there are perhaps multiple, parallel efforts creating the same module or submodule.

Some draft revisions may not include an update to the YANG modules or submodules contained in the draft. In that case, those modules or submodules that are not updated do not not require a change to their versions. Updates to the YANG Semver version MUST only be done when the revision of the module changes.

See Appendix A for a detailed example of IETF pre-release versions.

5.2.2. Guidelines for Published IETF Modules

For IETF YANG modules and submodules that have already been published, revision-labels MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver version rules specified in Section 3.4 . For example, if a module or submodule started out in the pre-NMDA ([RFC8342]) world, and then had NMDA support added without removing any legacy "state" branches -- and you are looking to add additional new features -- a sensible choice for the target YANG Semver would be 1.2.0 (since 1.0.0 would have been the initial, pre-NMDA release, and 1.1.0 would have been the NMDA revision).

6. YANG Module

This YANG module contains the typedef for the YANG semantic version and the identity to signal its use.

```
<CODE BEGINS> file "ietf-yang-semver@2023-01-17.yang"
module ietf-yang-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
  prefix ysver;
  rev:revision-label-scheme "yang-semver";

  import ietf-yang-revisions {
    prefix rev;
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>
     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>
```

```
        Author:   Reshad Rahman
                  <mailto:reshad@yahoo.com>
        Author:   Balazs Lengyel
                  <mailto:balazs.lengyel@ericsson.com>
        Author:   Jason Sterne
                  <mailto:jason.sterne@nokia.com>
        Author:   Benoit Claise
                  <mailto:benoit.claise@huawei.com>";
    description
      "This module provides type and grouping definitions for YANG
       packages.

       Copyright (c) 2022 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 Revised 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.
    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.
    // RFC Ed. update the rev:revision-label to "1.0.0".

    revision 2023-01-17 {
      rev:label "1.0.0-draft-ietf-netmod-yang-semver-10";
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }

    /*
     * Identities
     */

    identity yang-semver {
      base rev:revision-label-scheme-base;
      description
        "The revision-label scheme corresponds to the YANG Semver
         scheme which is defined by the pattern in the 'version'
```

```
          typedef below. The rules governing this revision-label
          scheme are defined in the reference for this identity.";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }

    /*
     * Typedefs
     */

    typedef version {
      type rev:revision-label {
        pattern '[0-9]+[.][0-9]+[.][0-9]+(_(non_)?compatible)?'
              + '(-[A-Za-z0-9.-]+[.-][0-9]+)?([+][A-Za-z0-9.-]+)?';
      }
      description
        "Represents a YANG semantic version.  The rules governing the
         use of this revision label scheme are defined in the
         reference for this typedef.";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }
  }
  <CODE ENDS>
```

7. Contributors

The following people made substantial contributions to this document:

Bo Wu
lana.wubo@huawei.com

Jan Lindblad
jlindbla@cisco.com

Figure 2

8. Acknowledgments

This document grew out of the YANG module versioning design team that started after IETF 101. The team consists of the following members whom have worked on the YANG versioning project: Balazs Lengyel, Benoit Claise, Bo Wu, Ebben Aries, Jan Lindblad, Jason Sterne, Joe Clarke, Juergen Schoenwaelder, Mahesh Jethanandani, Michael (Wangzitao), Per Andersson, Qin Wu, Reshad Rahman, Tom Hill, and Rob Wilton.

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update] . We would like the thank Kevin D'Souza for his initial work in this problem space.

Discussions on the use of SemVer for YANG versioning has been held with authors of the OpenConfig YANG models based on their own [openconfigsemver] . We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

We would also like to thank Joseph Donahue from the SemVer.org project for his input on SemVer use and overall document readability.

9. Security Considerations

The YANG module 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 [RFC8446] .

The NETCONF 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.

That said, the YANG module in this document does not define any schema nodes (i.e., nothing that can be read or written). It only defines a typedef and an identity. Therefore, there is no need to further protect any nodes with access control.

10. IANA Considerations

10.1. YANG Module Registrations

This document requests IANA to register a URI in the "IETF XML Registry" [RFC3688] . Following the format in RFC 3688, the following registration is requested.

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

Registrant Contact: The IESG.

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

The following YANG module is requested to be registered in the "IANA Module Names" [RFC6020] . Following the format in RFC 6020, the following registrations are requested:

The ietf-yang-semver module:

Name: ietf-yang-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-semver

Prefix: ysver

Reference: [RFCXXXX]

10.2. Guidance for YANG Semver in IANA maintained YANG modules and submodules

Note for IANA (to be removed by the RFC editor): Please check that the registries and IANA YANG modules and submodules are referenced in the appropriate way.

IANA is responsible for maintaining and versioning some YANG modules and submodules, e.g., iana-if-types.yang [IfTypeYang] and iana-routing-types.yang [RoutingTypesYang] .

In addition to following the rules specified in the IANA Considerations section of [I-D.ietf-netmod-yang-module-versioning] , IANA maintained YANG modules and submodules MUST also include a YANG Semver revision label for all new revisions, as defined in Section 3 .

The YANG Semver version associated with the new revision MUST follow the rules defined in Section 3.4 .

Note: For IANA maintained YANG modules and submodules that have already been published, revision labels MUST be retroactively applied to all existing revisions when the next new revision is created, starting at version "1.0.0" for the initial published revision, and then incrementing according to the YANG Semver rules specified in Section 3.4 .

Most changes to IANA maintained YANG modules and submodules are expected to be backwards-compatible changes and classified as MINOR version changes. The PATCH version may be incremented instead when only editorial changes are made, and the MAJOR version would be incremented if non-backwards-compatible changes are made.

Given that IANA maintained YANG modules are versioned with a linear history, it is anticipated that it should not be necessary to use the "_compatible" or "_non_compatible" modifiers to the "Z_COMPAT" version element.

11. References

11.1. 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, <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>.

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[I-D.ietf-netmod-yang-module-versioning] Wilton, R., Rahman, R., Lengyel, B., Clarke, J., and J. Sterne, "Updated YANG Module Revision Handling", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-module-versioning-09, 17 April 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-module-versioning-09>.

11.2. Informative References

[I-D.clacla-netmod-yang-model-update] Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", Work in Progress, Internet-Draft, draft-clacla-netmod-yang-model-update-06, 2 July 2018, <https://datatracker.ietf.org/doc/html/draft-clacla-netmod-yang-model-update-06>.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and B. Wu, "YANG Packages", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-packages-03, 4 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-packages-03>.

[I-D.ietf-netmod-yang-schema-comparison]
Andersson, P. and R. Wilton, "YANG Schema Comparison", Work in Progress, Internet-Draft, draft-ietf-netmod-yang-schema-comparison-02, 14 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-yang-schema-comparison-02>.

[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>.

[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>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[openconfigsemver]
"Semantic Versioning for Openconfig Models",
<http://www.openconfig.net/docs/semver/>.

[SemVer] "Semantic Versioning 2.0.0 (text from June 19, 2020)",
<https://github.com/semver/semver/
blob/8b2e8eec394948632957639dfa99fc7ec6286911/semver.md>.

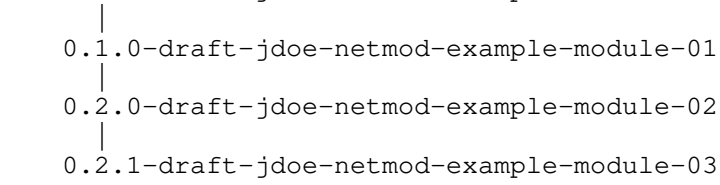
[IfTypeYang]
"iana-if-type YANG Module",
<https://www.iana.org/assignments/iana-if-type/iana-if-type.xhtml>.

[RoutingTypesYang]
"iana-routing-types YANG Module",
<https://www.iana.org/assignments/iana-routing-types/iana-routing-types.xhtml>.

Appendix A. Example IETF Module Development

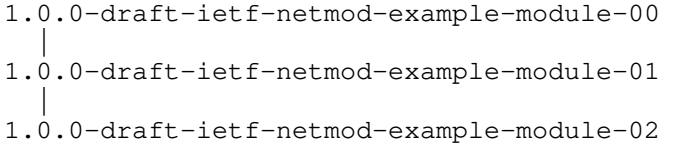
Assume a new YANG module is being developed in the netmod working group in the IETF. Initially, this module is being developed in an individual internet draft, draft-jdoe-netmod-example-module. The following represents the initial version tree (i.e., value of revision-label) of the module as it's being initially developed.

Version lineage for initial module development:



At this point, development stabilizes, and the workgroup adopts the draft. Thus now the draft becomes draft-ietf-netmod-example-module. The initial pre-release lineage continues as follows.

Continued version lineage after adoption:



At this point, the draft is ratified and becomes RFC12345 and the YANG module version becomes 1.0.0.

A time later, the module needs to be revised to add additional capabilities. Development will be done in a backwards-compatible way. Two new individual drafts are proposed to go about adding the capabilities in different ways: draft-jdoe-netmod-exmod-enhancements and draft-asmith-netmod-exmod-changes. These are initially developed in parallel with the following versions.

Parallel development for next module revision (track 1):

```
1.1.0-draft-jdoe-netmod-exmod-enhancements-00
  |
1.1.0-draft-jdoe-netmod-exmod-enhancements-01
```

In parallel with (track 2):

```
1.1.0-draft-asmith-netmod-exmod-changes-00
  |
1.1.0-draft-asmith-netmod-exmod-changes-01
```

At this point, the WG decides to merge some aspects of both and adopt the work in asmith's draft as draft-ietf-netmod-exmod-changes. A single version lineage continues.

```
1.1.0-draft-ietf-netmod-exmod-changes-00
  |
1.1.0-draft-ietf-netmod-exmod-changes-01
  |
1.1.0-draft-ietf-netmod-exmod-changes-02
  |
1.1.0-draft-ietf-netmod-exmod-changes-03
```

The draft is ratified, and the new module version becomes 1.1.0.

Authors' Addresses

Joe Clarke (editor)
Cisco Systems, Inc.
7200-12 Kit Creek Rd
Research Triangle Park, North Carolina
United States of America
Phone: +1-919-392-2867
Email: jclarke@cisco.com

Robert Wilton (editor)
Cisco Systems, Inc.
Email: rwilton@cisco.com

Reshad Rahman
Graphiant
Email: reshad@yahoo.com

Balazs Lengyel
Ericsson
1117 Budapest
Magyar Tudosok Korutja
Hungary
Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia
Email: jason.sterne@nokia.com

Benoit Claise
Huawei
Email: benoit.claise@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 April 2024

K. Larsson
Deutsche Telekom

18 October 2023

Mapping YANG Data to Label-Set Time Series
draft-kll-yang-label-tsdb-00

Abstract

This document proposes a standardized approach for representing YANG-modeled configuration and state data, for storage in Time Series Databases (TSDBs) that identify time series using a label-set. It outlines procedures for translating YANG data representations to fit within the label-centric structures of TSDBs and vice versa. This mapping ensures clear and efficient storage and querying of YANG-modeled data in TSDBs.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/plajjan/draft-kll-yang-label-tsdb.

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 20 April 2024.

Table of Contents

1. Introduction

The aim of this document is to define rules for representing configuration and state data defined using the YANG data modeling language [RFC7950] as time series using a label-centric model.

The majority of modern Time Series Databases (TSDBs) employ a label-centric model. In this structure, time series are identified by a set of labels, each consisting of a key-value pair. These labels facilitate efficient querying, aggregation, and filtering of data over time intervals. Such a model contrasts with the hierarchical nature of YANG-modeled data. The challenge, therefore, lies in ensuring that YANG-defined data, with its inherent structure and depth, can be seamlessly integrated into the flat, label-based structure of most contemporary TSDBs.

This document seeks to bridge this structural gap, laying out rules and guidelines to ensure that YANG-modeled configuration and state data can be effectively stored, queried, and analyzed within label-centric TSDBs.

2. Specification of the Mapping Procedure

Instances of YANG data nodes are mapped to metrics. Only nodes that carry a value are mapped. This includes leafs and presence containers. The hierarchical path to a value, including non-presence containers and lists, form the path that is used as the name of the metric. The path is formed by joining YANG data nodes using _. Special symbols, e.g. -, in node names are replaced with _.

List keys are mapped into labels. The path to the list key is transformed in the same way as the primary name of the metric. Compound keys have each key part as a separate label.

2.1. Example: Packet Counters in IETF Interfaces Model

Consider the in-unicast-pkts leaf from the IETF interfaces model that captures the number of incoming unicast packets on an interface:

Original YANG Instance-Identifier: yang /interfaces/interface[name='eth0']/statistics/in-unicast-pkts

Following the mapping rules defined:

1. The path components, including containers and list names, are transformed into the metric name by joining the node names with _. Special symbols, e.g. - are replaced with _.

Resulting Metric Name: interfaces_interface_statistics_in_unicast_pkts

1. The list key "predicate", which in this case is the interface name (eth0), is extracted and stored as a separate label. The label key represents the complete path to the key.

Resulting Label: interfaces_interface_name = eth0

1. The leaf value, which represents the actual packet counter, remains unchanged and is directly mapped to the value in the time series database.

For instance, if the packet counter reads 5,432,100 packets:

Value: 5432100

1. As part of the standard labels, a server identification string is also included. A typical choice of identifier might be the hostname. For this example, let's assume the device name is router-01:

Label: host = router-01

Final Mapping in the TSDB:

* Metric: interfaces_interface_statistics_in_unicast_pkts
* Value: 5432100
* Labels:
  - host = router-01
  - interfaces_interface_name = eth0

2.2. Mapping values

Leaf values are mapped based on their intrinsic type:

* All integer types are mapped to integers and retain their native representation
  - some implementations only support floats for numeric values
* decimal64 values are mapped to floats and the value should be rounded and truncated as to minimize the loss of information
* Enumeration types are mapped using their string representation.
* String types remain unchanged.

2.3. Choice

Choice constructs from YANG are disregarded and not enforced during the mapping process. Given the temporal nature of TSDBs, where data spans across time, different choice branches could be active in a single data set, rendering validation and storage restrictions impractical.

2.4. Host / device name

There is an implicit host label identifying the server, typically set to the name of the host originating the time series data.

Instance data retrieved from YANG-based servers do not generally identify the server it originates from. As a time series database is likely going to contain data from multiple servers, the host label is used to identify the source of the data.

3. Querying YANG modeled time series data

The process of storing YANG-modeled data in label-centric TSDBs, as defined in the previous sections, inherently structures the data in a way that leverages the querying capabilities of modern TSDBs. This chapter provides guidelines on how to construct queries to retrieve this data effectively.

3.1. 1. *Basic Queries*

To retrieve all data points related to incoming unicast packets from the IETF interfaces model:

* *InfluxQL*: sql SELECT * FROM interfaces_interface_statistics_in_unicast_pkts

* *PromQL*: promql interfaces_interface_statistics_in_unicast_pkts

3.2. 2. *Filtering by Labels*

To retrieve incoming unicast packets specifically for the interface eth0:

* *InfluxQL*: sql SELECT * FROM interfaces_interface_statistics_in_unicast_pkts WHERE interfaces_interface_name = 'eth0'

* *PromQL*: promql interfaces_interface_statistics_in_unicast_pkts{interfaces_interface_name="eth0"}

Similarly, to filter by device / host name:

* *InfluxQL*: sql SELECT * FROM interfaces_interface_statistics_in_unicast_pkts WHERE host = 'router-01'

* *PromQL*: promql interfaces_interface_statistics_in_unicast_pkts{host="router-01"}

3.3. 3. *Time-based Queries*

* *InfluxQL*: sql SELECT * FROM interfaces_interface_statistics_in_unicast_pkts WHERE time > now() - 24h

Prometheus fetches data based on the configured scrape interval and retention policies, so time-based filters in PromQL often center around the range vectors. For data over the last 24 hours:

* *PromQL*: promql interfaces_interface_statistics_in_unicast_pkts[24h]

3.4. 4. *Aggregations*

To get the average number of incoming unicast packets over the last hour:

* *InfluxQL*: sql SELECT MEAN(value) FROM interfaces_interface_statistics_in_unicast_pkts WHERE time > now() - 1h GROUP BY time(10m)

* *PromQL*: promql avg_over_time(interfaces_interface_statistics_in_unicast_pkts[1h])

3.5. 5. *Combining Filters*

To retrieve the sum of incoming unicast packets for eth0 on router-01 over the last day:

* *InfluxQL*: sql SELECT SUM(value) FROM interfaces_interface_statistics_in_unicast_pkts WHERE interfaces_interface_name = 'eth0' AND host = 'router-01' AND time > now() - 24h

* *PromQL*: promql sum(interfaces_interface_statistics_in_unicast_pkts{interfaces_interface_name="eth0", host="router-01"})[24h]

3.6. 6. *Querying Enumeration Types*

In YANG models, enumerations are defined types with a set of named values. The oper-status leaf in the IETF interfaces model is an example of such an enumeration, representing the operational status of an interface.

For instance, the oper-status might have values such as up, down, or testing.

To query interfaces that have an oper-status of up:

* *InfluxQL*: sql SELECT * FROM interfaces_interface_oper_status WHERE value = 'up'

* *PromQL*: promql interfaces_interface_oper_status{value="up"}

Similarly, to filter interfaces with oper-status of down:

* *InfluxQL*: sql SELECT * FROM interfaces_interface_oper_status WHERE value = 'down'

* *PromQL*: promql interfaces_interface_oper_status{value="down"}

This approach allows us to effectively query interfaces based on their operational status, leveraging the enumeration mapping within the TSDB.

4. Requirements on time series databases

This document specifies a mapping to a conceptual representation, not a particular concrete interface. To effectively support the mapping of YANG-modeled data into a label-centric model, certain requirements must be met by the Time Series Databases (TSDBs). These requirements ensure that the data is stored and retrieved in a consistent and efficient manner.

4.1. Support for String Values

Several YANG leaf types carry string values, including the string type itself and all its descendants as well as enumerations which are saved using their string representation.

The chosen TSDB must support the storage and querying of string values. Not all TSDBs inherently offer this capability, and thus, it's imperative to ensure compatibility.

4.2. Sufficient Path Length

YANG data nodes, especially when representing deep hierarchical structures, can result in long paths. When transformed into metric names or labels within the TSDB, these paths might exceed typical character limits imposed by some databases. It's essential for the TSDB to accommodate these potentially long names to ensure data fidelity and avoid truncation or loss of information.

4.3. High Cardinality

Given the possibility of numerous unique label combinations (especially with dynamic values like interface names, device names, etc.), the chosen TSDB should handle high cardinality efficiently. High cardinality can impact database performance and query times, so it's essential for the TSDB to have mechanisms to manage this efficiently.

5. Normative References

[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>.

Author's Address

Kristian Larsson
Deutsche Telekom
Email: kristian@spritelink.net

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: 22 April 2024

J. Lindblad
Cisco

20 October 2023

Philatelist, YANG-based collection and aggregation framework integrating Telemetry data and Time Series Databases
draft-lindblad-tlm-philatelist-00

Abstract

Timestamped telemetry data is collected en masse today. Mature tools are typically used, but the data is often collected in an ad hoc manner. While the dashboard graphs look great, the resulting data is often of questionable quality, not well defined, and hard to compare with seemingly similar data from other organizations.

This document proposes a standard, extensible, cross domain framework for collecting and aggregating timestamped telemetry data in a way that combines YANG, metadata and Time Series Databases to produce more dependable and comparable results.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://janlindblad.github.io/netmod-tlm-philatelist/draft-lindblad-tlm-philatelist.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-lindblad-tlm-philatelist/.

Source for this draft and an issue tracker can be found at https://github.com/janlindblad/netmod-tlm-philatelist.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. The Problem

Many organizations today are collecting large amounts of telemetry data from their networks and data centers for a variety of purposes. Much (most?) of this data is funneled into a Time Series Database (TSDB) for display in a dashboard or further (AI-backed) processing and decision making.

While this data collection is often handled using standard tools, there generally seems to be little commonality when it comes to what is meaured, how the data is aggregated, or definitions of the measured quantities (if any).

Data science issues like adding overlapping quantities, adding quantities of different units of measurement, or quantities with different scopes, are likely common. Such errors are hard to detect given the ad hoc nature of the collection. This often leads to uncertainty regarding the quality of the conclusions drawn from the collected data.

1.2. The Solution

The Philatelist framework proposes to standardize the collection, definitions of the quantities measured and meta data handling to provide a robust ground layer for telemetry collection. The architecture defines a few interfaces, but allows great freedom in the implementations with its plug-in architecture. This allows flexibility enough that any kind of quantitiy can be measured, any kind of collection protocol and mechanism employed, and the data flows aggregated using any kind of operation.

To do this, YANG is used both to describe the quantities being measured, as well as act as the framework for the metadata management. Note that the usa of YANG here does not limit the architecture to traditional YANG-based transport protocols. YANG is used to describe the data, regardless of which format it arrives in.

Initially developed in context of the Power and Energy Efficiency work (POWEFF), we realized both the potential and the need for this collection and aggregation architecture to become a general framework for collection of a variety of metrics.

There is not much point in knowing the "cost side" of a running system (as in energy consumption or CO2-emissions) if that cannot be weighed against the "value side" delivered by the system (as in transported bytes, VPN connections, music streaming hours, or number of cat videos, etc.), which means traditional performance metrics will play an equally important role in the collection.

In this initial version, we have done nothing to pull the proposed YANG modules out of its POWEFF roots and generalize it for general telemetry. We believe the ideas and merits of this framework architecture will be apparent nonetheless in this first version. For the next version, we certainly need to generalize the quantities measured and rename the YANG modules and node names.

1.3. The Philatelist Name

This specification is about a framework for collection, aggregation and interpretation of timestamped telemetry data. The definition of "philatelist" seems close enough.

1. philatelist

noun. [’flætlst’] a collector and student of postage stamps.

Synonyms
- collector
- aggregator

Figure 1: Source: https://www.synonym.com/synonyms/philatelist

2. Conventions and Definitions

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.

This document uses the terminology defined in [RFC7950].

In addition, this document defines the following terms:

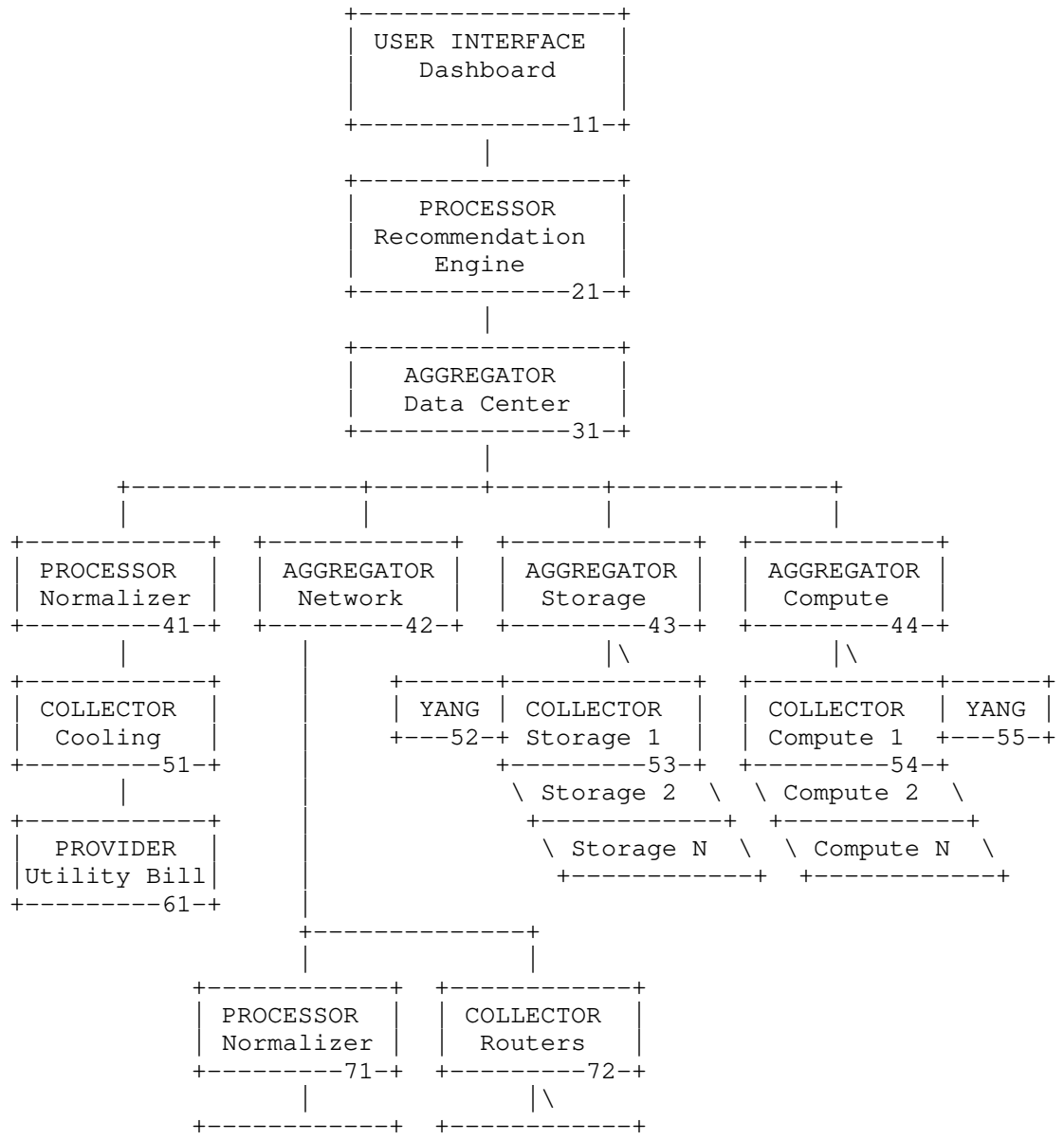
TSDB Time Series Database.

Sensor An entity in a system that delivers a snapshot value of some quantity pertaining to the system. Sensors are identified by their Sensor Path.

Sensor Path A textual representation of the sensor’s address within the system.

3. Architecure Overview

The deployment of a Philatelist framework consists of a collection of plug-in compomnents with well defined interfaces. Here is an example of a deployment. Each box is numbered in the lower right for easy reference.



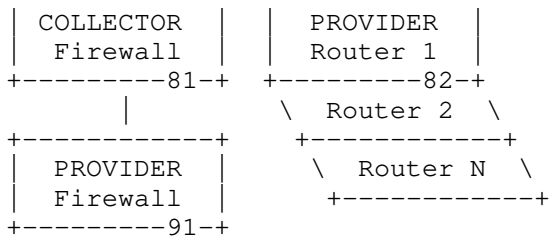


Figure 2: Example Philatelist component deployment.

Each component in the above diagram, represents a logical function. Many boxes could be running within a single server, or they could be fully distrubuted, or anything in between.

Provider components (61, 82, 91) are running on a telemetry source system that supports a YANG-based telemetry data server. The telemetry data flows from the telemetry source system to a Time Series Database (TSDB).

Collector components (51, 72, 81) ensure the Providers are programmed properly to deliver the telemetry data to the TSDB designated by the collector. In some cases this flow may be direct from the source to the TSDB, in other cases, it may be going through the collector. In some cases the collector may be polling the source, in others it may have set up an automatic, periodic subscription.

Many telemetry source systems will not have any on-board YANG-based telemetry server. Such servers will instead be managed by a collector specialized to handle a particular kind of source server (53, 54). These specialized collectors are still responsible to set up a telemetry data stream from them to the collector's TSDB. In this case, the collector will also supply a YANG description (52, 55) of the incoming data stream.

Processor components (21, 41, 71) are transforming the data stream in some way, e.g. converting from one unit of measurement to another, or adjusting the data values recorded to also include some aspect that this particular sensor is not taking into account.

Aggregator components (31, 42, 43, 44) combine the time series telemetry data flows using some operation, e.g. summing, averaging or computing the max or min over them. In this example there are aggregators for Network, Storage, Compute and the entire Data Center

On top of the stack, we may often find a (graphical) user interface (11), for human consumption of the intelligence acquired by the system. Equally relevant is of course an (AI) application making decisions based on findings in the aggregated telemetry flow.

3.1. The Provider Component

A Provider is a source of telemetry data that also offers a YANG-based management interface. Each provider typically has a large number of "sensors" that can be polled or in some cases subscribed to.

One problem with the sensors is that they are spread around inside the source system, and may not be trivial to locate. Also, the metadata asscociated with the sensor is often only missing or only available in human readable form (free form strings), rather than in a strict machine parsable format.

```
/hardware/component[name="psu3"]/.../sensor-data/value
...
/interfaces/interface[name="eth0/0"]/.../out-broadcast-packets
...
/routing/mpls/mpls-label-blocks/.../inuse-labels-count
...
```

Figure 3: Example of scattered potential sensors in a device.

To solve these problems, the Provider YANG module contains a sensor-catalog list. Essentially a list of all interesting sensors available on the system, with their sensor paths and machine parsable units, definition and any other metadata.

An admin user or application can then copy the sensor definition from the sensor catalog and insert into the configuration in the colletor.

```
  +--ro sensor-catalog
      +--ro sensors
        +--ro sensor* [path]
            +--ro path?                    xpath
            +--ro sensor-type?             identityref
            +--ro sensor-location?         something
            +--ro sensor-state?            something
            +--ro sensor-current-reading?  something
            +--ro sensor-precision?        string
```

Figure 4: YANG tree diagram of the Provider sensor-catalog.

Note: The "something" YANG-type is used in many places in this document. That is just a temporarty placeholder we use until we have figured out what the appropriate type should be.

The sensor types are defined as YANG identities, making them maximally extensible. Examples of sensor types might be energy measured in kWh, or energy measured in J, or temperature measured in F, or in C, or in K.

3.2. The Collector Component

Collector components collect data points from sources, typically by periodic polling or subscriptions, and ensure the collected data is stored in a Time Series Database (TSDB). The actual data stream may or may not be passing through the collector component; the collector is responsible for ensuring data flows from the source to the destination TSDB and that the data has a YANG description and is tagged with necessary metadata. How the collector agrees with a source to deliver data in a timely manner is beyond the scope of this document.

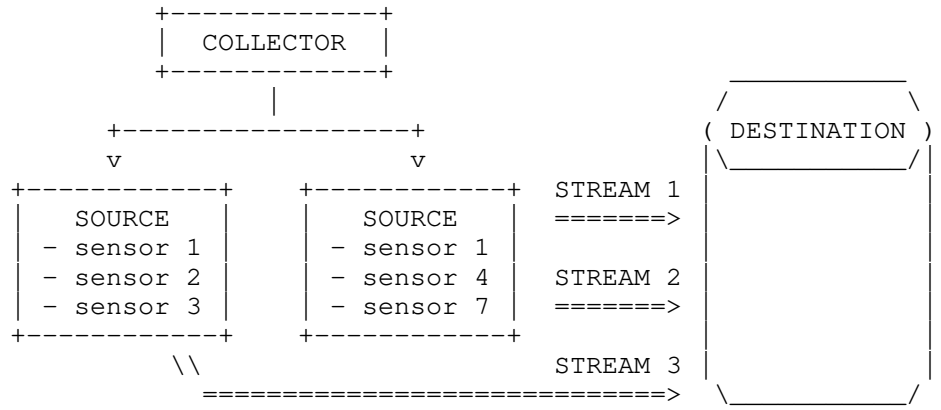


Figure 5: Example of Collector setting up three streams of telemetry data from two sources to one desitination.

Each source holds a number of sensors that may be queried or subscribed to. The collector arranges the sensors into sensour groups that presumably are logically related, and that are collected using the same method. A number of collection methods (some YANG-based, some not) are modeled directly in the ietf-poweff-collector.yang module, but the set is designed to be easily extensible.

```
+-- sensor-groups
|  +-- sensor-group* [id]
|     +-- id?                                  something
|     +-- method?                              identityref
|     +-- get-static-url-once
|     |  +-- url?                              something
|     |  +-- format?                           something
|     +-- gnmi-polling
|     |  +-- encoding?                         something
|     |  +-- protocol?                         something
|     +-- restconf-get-polling
|     |  +-- xxx?                              something
|     +-- netconf-get-polling
|     |  +-- xxx?                              something
|     +-- restconf-yang-push-subscription
|     |  +-- xxx?                              something
|     +-- netconf-yang-push-subscription
|     |  +-- xxx?                              something
|     +-- redfish-polling
|     |  +-- xxx?                              something
|     +-- frequency?                           sample-frequency
|     +-- path* [path]
|        +-- path?                             xpath
|        +-- sensor-type?                      identityref
+-- streams
  +-- stream* [id]
      +-- id?                                  something
      +-- source*                              string
      +-- sensor-group* [name]
      |  +-- name?   -> ../../../sensor-groups/sensor-group/id
      +-- destination?    -> ../../../destinations/destination/id
```

Figure 6: YANG tree diagram of the Collector sensor-groups and streams.

The sensor groups are then arranged into streams from a collection of sources (that support the same set of sensor groups) to a destination. This structure has been chosen with the assumption that there will be many source devices with the same set of sensor groups, and we want to minimize repetition.

3.3. The Processor and Aggregator Components

Processor components take an incoming data flow and transforms it somehow, and possibly augments it with a flow of derived information. The purpose of the transformation could be to convert between different units of measurement, correct for known errors in in the input data, or fill in approximate values where there are holes in the input data.

Aggregator components take multiple incoming data flows and combine them, typically by adding them together, taking possible differences in cadence in the input data flows into account.

Processor and Aggregator components provide a YANG model of the output data, just like the Collector components, so that all data flowing in the system has a YANG description and is associated with metadata.

Note: In the current version of the YANG modules, a Processor is simply an Aggregator with a single input and output. Unless we see a need to keep these two component types separate, we might remove the Processor component and keep it baked in with the Aggregator.

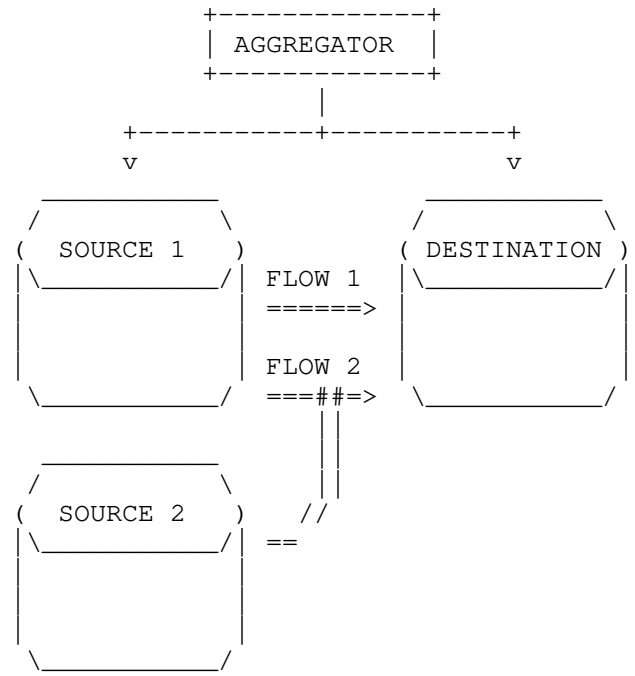


Figure 7: Example of an Aggregator setting up two flows of telemetry data from two sources to one desitination.

In this diagram, the sources and destination look like separate TSDBs, which they might be. They may also be different buckets within the same TSDB.

Each flow is associated with one or more inputs, one output and a series of processing operations. Each input flow and output flow may have an pre-processing or post-processing operation applied to it separately. Then all the input flows are combined using one or more aggregation operations. Some basic operations have been defined in the Aggregator YANG module, but the set of operations has been designed to be maximally extensible.

```
+-- flows
|  +-- flow* [id]
|     +-- id?                                  string
|     +-- (chain-position)?
|        +--:(input)
|        |  +-- input
|        |     +-- source?
|        |           -> ../../../../../sources/source/id
|        +--:(output)
|        |  +-- output
|        |     +-- destination?
|        |           -> ../../../../../destinations/destination/id
|        +--:(middle)
|           +-- middle
|              +-- inputs*
|              |     -> ../../../../flows/flow/id
|              +-- pre-process-inputs?
|              |     -> ../../../../operations/operation/id
|              +-- aggregate?
|              |     -> ../../../../operations/operation/id
|              +-- post-process-output?
|                    -> ../../../../operations/operation/id
+-- operations
  +-- operation* [id]
     +-- id?                                   something
     +-- (op-type)?
       +--:(linear-sum)
       |  +-- linear-sum
       +--:(linear-average)
       |  +-- linear-average
       +--:(linear-max)
       |  +-- linear-max
       +--:(linear-min)
       |  +-- linear-min
       +--:(rolling-average)
       |  +-- rolling-average
       |     +-- timespan?                   something
       +--:(filter-age)
       |  +-- filter-age
       |     +-- min-age?                    something
       |     +-- max-age?                    something
       +--:(function)
           +-- function
             +-- name?                       something
```

Figure 8: YANG tree diagram of the Aggregator flows and operations.

The operations listed above are basic aggregation operations. Linear-sum is just adding all the input sources together, with linear interpolation when their data points don't align perfectly in time. Rolling average is averaging the input flows over a given length of time. The filter-age drops all data points that are outside the min to max age. The function allows plugging in any other function the Aggregator may have defined, but more importantly, the operations choice is easily extended using YANG augment to include any other IETF or vendor specified extensions.

4. YANG-based Telemetry Outlook

Much work has already gone into the area of telemetry, YANG, and even their intersection. E.g. [I-D.draft-ietf-opsawg-collected-data-manifest-01] and [I-D.draft-claise-netconf-metadata-for-collection-03] come to mind.

Even though this work has a solid foundation and shares many or most of the goals with this work, we (the POWEFF team) have not found it easy to apply the above work directly in the practical work we do. So what we have tried to do is a very pragmatic approach to telemetry data collection the way we see it happening on the ground combined with the benefits of Model Driven Telemetry (MDT), in practice meaning YANG-based with additional YANG-modeled metadata.

Many essential data sources in real world deployments do not support any YANG-based interfaces, and that situation is expected to remain for the forseable future, which is why we find it important to be able to ingest data from free form (often REST-based) interfaces, and then add the necessary rigor on the Collector level. Then output the datastreams in formats that existing, mature tools can consume directly.

In particular, this draft depends on the mapping of YANG-based structures to the typical TSDB tag-based formats described in [I-D.draft-kll-yang-label-tsdb-00].

For the evolution of the YANG-based telemetry area, we believe this approach, combining pragmatism in the data flow interfaces with rigor regarding the data content, is key. We would like to make this work fit in with the works of others in the field.

5. YANG Modules

5.1. Base types module for Philatelist

```
<CODE BEGINS>
module ietf-poweff-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-poweff-types";
  prefix ietf-poweff-types;

  organization
    "IETF OPSA (Operations and Management Area) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Editor:  Jan Lindblad
              <mailto:jlindbla@cisco.com>
     Editor:  Snezana Mitrovic
              <mailto:snmitrov@cisco.com>
     Editor:  Marisol Palmero
              <mailto:mpalmero@cisco.com>";
  description
    "This YANG module defines basic quantities, measurement units
    and sensor types for the POWEFF framework.

     Copyright (c) 2021 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).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.";

  revision 2023-10-12 {
    description
      "Initial revision of POWEFF types";
    reference
      "RFC XXXX: ...";
  }

  typedef something { // FIXME: Used when we haven't decided the type yet
    type string;
  }
  typedef xpath {
    type string; // FIXME: Proper type needed
```

```
  }
  typedef sample-frequency {
    type string; // FIXME: Proper type needed
  }

  // ========== SENSOR-CLASS ==============================
  identity sensor-class {
    description "Sensor's relation to the asset it sits on.";
  }
  identity sc-input {
    base sensor-class;
    description "Sensor reports input quantity of the asset it sits
      on.";
  }
  identity sc-output {
    base sensor-class;
    description "Sensor reports output quantity of the asset it sits
      on.";
  }
  identity sc-allocated {
    base sensor-class;
    description "Sensor reports (maximum) allocated quantity of the
      asset it sits on.";
  }

  // ========== SENSOR-QUANTITY ===========================
  identity sensor-quantity {
    description "Sensor's quantity being measured.";
  }
  identity sq-voltage {
    base sensor-quantity;
    description "Sensor reports electric tension, voltage.";
  }
  identity sq-current {
    base sensor-quantity;
    description "Sensor reports electric current.";
  }
  identity sq-power {
    base sensor-quantity;
    description "Sensor reports power draw (energy per unit of time).";
  }
  identity sq-power-apparent {
    base sq-power;
    description "Sensor reports apparent power, i.e. average electrical
      current times voltage (in VA).";
  }
  identity sq-power-true {
    base sq-power;
```

```
      description "Sensor reports true power, i.e. integral over current
        and voltage at each instant in time.";
    }
    identity sq-energy {
      base sensor-quantity;
      description "Sensor reports actual energy drawn by asset.";
    }
    identity sq-co2-emission {
      base sensor-quantity;
      description "Sensor reports CO2 (carbon dioxide) emission by
        asset.";
    }
    identity sq-co2eq-emission {
      base sensor-quantity;
      description "Sensor reports CO2 (carbon dioxide) equivalent
        emission by asset.";
    }
    identity sq-temperature {
      base sensor-quantity;
      description "Sensor reports temperature of asset.";
    }

    // ========== SENSOR-UNIT ==============================
    identity sensor-unit {
      description "Sensor's unit of reporting.";
    }
    identity su-volt {
      base sensor-unit;
      base sq-voltage;
      description "Sensor unit volt, V.";
    }
    identity su-ampere {
      base sensor-unit;
      base sq-current;
      description "Sensor unit ampere, A.";
    }
    identity su-watt {
      base sensor-unit;
      base sq-power;
      description "Sensor unit watt, W.";
    }
    identity su-voltampere {
      base sensor-unit;
      base sq-power;
      description "Sensor unit Volt*Ampere, VA.";
    }
    identity su-kw {
      base sensor-unit;
```

```
      base sq-power;
      description "Sensor unit kilowatt, kW.";
    }
    identity su-joule {
      base sensor-unit;
      base sq-energy;
      description "Sensor unit joule, J.";
    }
    identity su-wh {
      base sensor-unit;
      base sq-energy;
      description "Sensor unit watthour, Wh.";
    }
    identity su-kwh {
      base sensor-unit;
      base sq-energy;
      description "Sensor unit kliowatthour, kWh.";
    }
    identity su-kelvin {
      base sensor-unit;
      base sq-temperature;
      description "Sensor unit kelvin, K.";
    }
    identity su-celsius {
      base sensor-unit;
      base sq-temperature;
      description "Sensor unit celsius, C.";
    }
    identity su-farenheit {
      base sensor-unit;
      base sq-temperature;
      description "Sensor unit farenheit, F.";
    }
    identity su-gram {
      base sensor-unit;
      base sq-co2-emission;
      description "Sensor unit gram, g.";
    }
    identity su-kg {
      base sensor-unit;
      base sq-co2-emission;
      description "Sensor unit kliogram, kg.";
    }
    identity su-ton {
      base sensor-unit;
      base sq-co2-emission;
      description "Sensor unit ton, t.";
    }
```

```
// ========== SENSOR-TYPE ==============================
identity sensor-type {
  description "Sensor's type, i.e. combination of class, quantity and
    unit.";
}
identity st-v-in {
  base sensor-type;
  base sc-input;
  base sq-voltage;
  base su-volt;
  description "Sensor reporting Voltage In to asset.";
}
identity st-v-out {
  base sensor-type;
  base sc-output;
  base sq-voltage;
  base su-volt;
  description "Sensor reporting Voltage Out of asset.";
}
identity st-i-in {
  base sensor-type;
  base sc-input;
  base sq-current;
  base su-ampere;
  description "Sensor reporting Current In to asset.";
}
identity st-i-out {
  base sensor-type;
  base sc-output;
  base sq-current;
  base su-ampere;
  description "Sensor reporting Current Out of asset.";
}
identity st-p-in-apparent-watt {
  base sensor-type;
  base sc-input;
  base sq-power-apparent;
  base su-voltampere;
  description "Sensor reporting Power In to asset as apparent (I*U)
    power.";
}
identity st-p-out-apparent-watt {
  base sensor-type;
  base sc-output;
  base sq-power-apparent;
  base su-voltampere;
  description "Sensor reporting Power Out of asset as apparent (I*U)
    power.";
```

```
    }
    identity st-p-in-true-watt {
      base sensor-type;
      base sc-input;
      base sq-power-true;
      base su-watt;
      description "Sensor reporting Power In to asset as true power.";
    }
    identity st-p-out-true-watt {
      base sensor-type;
      base sc-output;
      base sq-power-true;
      base su-watt;
      description "Sensor reporting Power Out of asset as true power.";
    }
    identity st-p-allocated-watt {
      base sensor-type;
      base sc-allocated;
      base sq-power;
      base su-watt;
      description "Sensor reporting Allocated Power for asset.";
    }
    identity st-w-j {
      base sensor-type;
      base sq-energy;
      base su-joule;
      description "Sensor reporting energy draw of asset in J.";
    }
    identity st-w-wh {
      base sensor-type;
      base sq-energy;
      base su-wh;
      description "Sensor reporting energy draw of asset in Wh.";
    }
    identity st-w-kwh {
      base sensor-type;
      base sq-energy;
      base su-kwh;
      description "Sensor reporting energy draw of asset in kWh.";
    }
    identity st-t-k {
      base sensor-type;
      base sq-temperature;
      base su-kelvin;
      description "Sensor reporting Temperature of asset in K.";
    }
    identity st-t-c {
      base sensor-type;
```

```
      base sq-temperature;
      base su-celsius;
      description "Sensor reporting Temperature of asset in °C.";
    }
    identity st-t-f {
      base sensor-type;
      base sq-temperature;
      base su-farenheit;
      description "Sensor reporting Temperature of asset in °F.";
    }

    // ========== COLLECTION-METHOD ==============================

    identity collection-method;
    identity cm-polled {
      base collection-method;
    }
    identity cm-gnmi {
      base collection-method;
    }
    identity cm-restconf {
      base collection-method;
    }
    identity cm-netconf {
      base collection-method;
    }
    identity cm-redfish {
      base collection-method;
    }
    identity get-static-url-once {
      base collection-method;
    }
    identity gnmi-polling {
      base cm-gnmi;
      base cm-polled;
    }
    identity restconf-get-polling {
      base cm-restconf;
      base cm-polled;
    }
    identity netconf-get-polling {
      base cm-netconf;
      base cm-polled;
    }
    identity restconf-yang-push-subscription {
      base cm-restconf;
    }
    identity netconf-yang-push-subscription {
```

```
      base cm-netconf;
    }
    identity redfish-polling {
      base cm-redfish;
    }
  }
  <CODE ENDS>
```

5.2. Provider interface module for Philatelist

```
  <CODE BEGINS>
  module ietf-poweff-provider {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-poweff-provider";
    prefix ietf-poweff-provider;

    import ietf-poweff-types {
      prefix ietf-poweff-types;
    }

    organization
      "IETF OPSA (Operations and Management Area) Working Group";
    contact
      "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
       WG List:  <mailto:opsawg@ietf.org>
       Editor:  Jan Lindblad
                <mailto:jlindbla@cisco.com>
       Editor:  Snezana Mitrovic
                <mailto:snmitrov@cisco.com>
       Editor:  Marisol Palmero
                <mailto:mpalmero@cisco.com>";
    description
      "This YANG module defines the POWEFF Provider.

       Copyright (c) 2021 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).

       This version of this YANG module is part of RFC XXXX
       (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
       for full legal notices.";
```

```
     revision 2023-10-12 {
       description
         "Initial revision of POWEFF Provider";
       reference
         "RFC XXXX: ...";
     }

     grouping provider-g {
       container sensor-catalog {
         config false;
         container sensors {
           list sensor {
             key path;
             leaf path { type ietf-poweff-types:xpath; }
             leaf sensor-type { type identityref { base ietf-poweff-types:sensor-
type; }}

             leaf sensor-location {
               type ietf-poweff-types:something;
               description
                 "Indicates the current location where the sensor is located
                   in the chassis,typically refers to slot";
             }
             leaf sensor-state { // FIXME: What does this mean?
               type ietf-poweff-types:something;
               description
                 "Current state of the sensor";
             }
             leaf sensor-current-reading { // FIXME: Do we want a copy of the val
ue here?
               type ietf-poweff-types:something;
               description
                 "Current reading of the sensor";
             }
             leaf sensor-precision {
               type string;
               description
                 "Maximum deviation to be considered. This attribute mainly
                 will apply to drawn power, which corresponds to PSU PowerIn
                 measured power or calculated power; assuming discrepancy
                 between Real Power, power collected from a power meter, and
                 power measured or calculated from the metrics provided by
                 the sensors";
             }
             container sensor-thresholds { // FIXME: Is this for generating alarm
s, or what?
               description
                 "Threshold values for the particular sensor.
                 Default values shall beprovided as part of static data
                 but when configurable need to be pulledfrom the device.
                 Ideally, the sensor should allow configuing
```

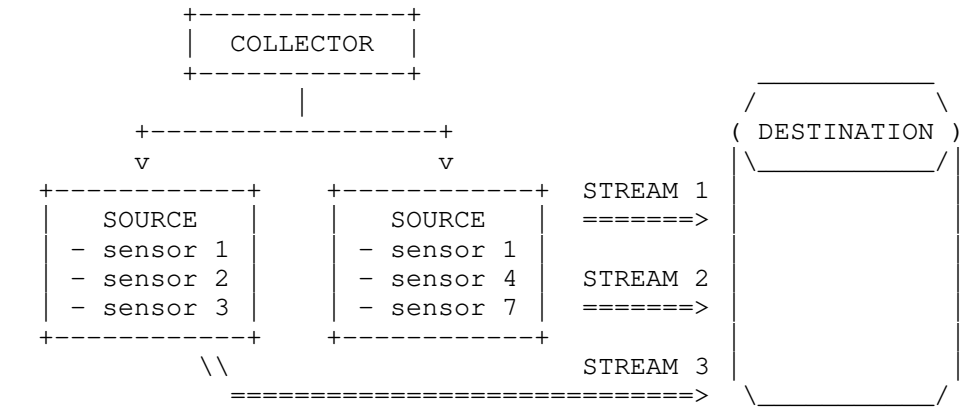
```
                thesethreshold values";

              leaf minor-low {
                type string;
                description
                  "minor-low";
              }
              leaf minor-high {
                type string;
                description
                  "minor-high";
              }
              leaf major-low {
                type string;
                description
                  "major-low";
              }
              leaf major-high {
                type string;
                description
                  "major-high";
              }
              leaf critical-low {
                type string;
                description
                  "critical-low";
              }
              leaf critical-high {
                type string;
                description
                  "critical-high";
              }
              leaf shutdown { // FIXME: What does this mean for a sensor?
                type string;
                description
                  "shutdown";
              }
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

5.3. Collector interface module for Philatelist

```
<CODE BEGINS>
module ietf-poweff-collector {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-poweff-collector";
  prefix ietf-poweff-collector;

  import ietf-poweff-types {
    prefix ietf-poweff-types;
  }

  organization
    "IETF OPSA (Operations and Management Area) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>
     Editor:  Jan Lindblad
              <mailto:jlindbla@cisco.com>
     Editor:  Snezana Mitrovic
              <mailto:snmitrov@cisco.com>
     Editor:  Marisol Palmero
              <mailto:mpalmero@cisco.com>";
  description
    "This YANG module defines the POWEFF Collector.

     Copyright (c) 2021 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).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.";

  revision 2023-10-12 {
    description
      "Initial revision of POWEFF Collector";
    reference
      "RFC XXXX: ...";
  }

/*
```

A COLLECTOR programs one or more SOURCE(s) to generate a STREAM of telemetry data. The STREAM is sent to a specific DESTINATION.

Each STREAM consists of timestamped sensor values from each sensor in a sensor group.




```
*/

  grouping data-endpoint-g {
    leaf url { type ietf-poweff-types:something; }
    leaf organization { type ietf-poweff-types:something; }
    leaf bucket { type ietf-poweff-types:something; }
    container impl-specific {
      list binding {
        key key;
        leaf key { type string; }
        choice value-type {
          leaf value { type string; }
          leaf-list values { type string; ordered-by user; }
          leaf env-var { type string; }
        }
      }
    }
  }

  grouping sensor-group-g {
    leaf method {
      type identityref {
        base ietf-poweff-types:collection-method;
      }
    }
    container get-static-url-once {
```

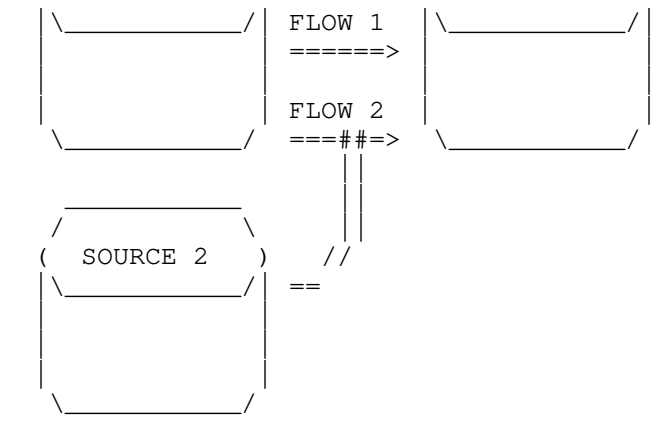
```
          when "derived-from-or-self(../method, 'ietf-poweff-types:get-static-url-
once')";
          leaf url { type ietf-poweff-types:something; }
          leaf format { type ietf-poweff-types:something; } // JSON-IETF, XML, etc
        }
        container gnmi-polling {
          when "derived-from-or-self(../method, 'ietf-poweff-types:gnmi-polling')"
;
          leaf encoding { type ietf-poweff-types:something; } // self-describing-g
pb
          leaf protocol { type ietf-poweff-types:something; } // protocol grpc no-
tls
        }
        container restconf-get-polling {
          when "derived-from-or-self(../method, 'ietf-poweff-types:restconf-get-po
lling')";
          leaf xxx { type string; }
        }
        container netconf-get-polling {
          when "derived-from-or-self(../method, 'ietf-poweff-types:netconf-get-pol
ling')";
          leaf xxx { type string; }
        }
        container restconf-yang-push-subscription {
          when "derived-from-or-self(../method, 'ietf-poweff-types:restconf-yang-p
ush-subscription')";
          leaf xxx { type string; }
        }
        container netconf-yang-push-subscription {
          when "derived-from-or-self(../method, 'ietf-poweff-types:netconf-yang-pu
sh-subscription')";
          leaf xxx { type string; }
        }
        container redfish-polling {
          when "derived-from-or-self(../method, 'ietf-poweff-types:redfish-polling
')";
          leaf xxx { type string; }
        }
        leaf frequency {
          when "derived-from(../method, 'ietf-poweff-types:cm-polled')";
          type ietf-poweff-types:sample-frequency;
        }
        list path {
          key path;
          leaf path { type ietf-poweff-types:xpath; }
          leaf sensor-type { type identityref { base ietf-poweff-types:sensor-type
; }}
          leaf attribution { type string; }
        }
      }

      grouping collector-g {
        container poweff-collector {
          container destinations {
            list destination {
              key id;
              leaf id { type ietf-poweff-types:something; }
              uses data-endpoint-g;
```

```
          }
        }

        container sensor-groups {
          list sensor-group {
            key id;
            leaf id { type ietf-poweff-types:something; }
            uses sensor-group-g;
          }
        }

        container streams {
          list stream {
            key id;
            leaf id { type ietf-poweff-types:something; }
            leaf-list source { type string; } // Implementation specific meaning
, possibly wildcards
            list sensor-group {
              key name;
              leaf name { type leafref { path ../../../sensor-groups/sensor-grou
p/id; }}
            }
            leaf destination { type leafref { path ../../../destinations/destina
tion/id; }}
          }
        }
      }
    }
  }
  <CODE ENDS>
```

5.4. Aggregator interface module for Philatelist

```
  <CODE BEGINS>
  module ietf-poweff-aggregator {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-poweff-aggregator";
    prefix ietf-poweff-aggregator;

    import ietf-poweff-types {
      prefix ietf-poweff-types;
    }
    import ietf-poweff-collector {
      prefix ietf-poweff-collector;
    }

    organization
      "IETF OPSA (Operations and Management Area) Working Group";
    contact
      "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
       WG List:  <mailto:opsawg@ietf.org>
```

```
        Editor:  Jan Lindblad
                 <mailto:jlindbla@cisco.com>
        Editor:  Snezana Mitrovic
                 <mailto:snmitrov@cisco.com>
        Editor:  Marisol Palmero
                 <mailto:mpalmero@cisco.com>";
    description
      "This YANG module defines the POWEFF Aggregator.

       Copyright (c) 2021 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).

       This version of this YANG module is part of RFC XXXX
       (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
       for full legal notices.";

    revision 2023-10-12 {
      description
        "Initial revision of POWEFF Aggregator";
      reference
        "RFC XXXX: ...";
    }

  /*

    An AGGREGATOR ensures data from one or more SOURCE(s) are
    combined into a FLOW using a (sequence of) OPERATIONs (OPs)
    to generate a new data set in the DESTINATION (which could
    be a new collection in the same data storage system as the
    SOURCE).

                  +-------------+
                  | AGGREGATOR  |
                  +-------------+
                         |
              +----------+----------+
              v                     v
       ____________           ____________
      /            \         /            \
     (  SOURCE 1    )       ( DESTINATION )
```




```
   */

     grouping aggregator-g {
       container poweff-aggregator {
         container sources {
           list source {
             key id;
             leaf id { type ietf-poweff-types:something; }
             uses ietf-poweff-collector:data-endpoint-g;
           }
         }
         container destinations {
           list destination {
             key id;
             leaf id { type ietf-poweff-types:something; }
             uses ietf-poweff-collector:data-endpoint-g;
           }
         }
         container flows {
           list flow {
             key id;
             leaf id { type string; }
             choice chain-position {
               container input {
                 leaf source { type leafref { path ../../../../../sources/source/
id; }}
               }
               container output {
                 leaf destination { type leafref { path ../../../../../destinatio
ns/destination/id; }}
               }
               container middle {
                 leaf-list inputs { type leafref { path ../../../../flows/flow/id
; }}
                 leaf pre-process-inputs { type leafref { path ../../../../operat
ions/operation/id; }}
```

```
                leaf aggregate { type leafref { path ../../../../operations/oper
ation/id; }}
                leaf post-process-output { type leafref { path ../../../../opera
tions/operation/id; }}
              }
            }
          }
        }
        container operations {
          list operation {
            key id;
            leaf id { type ietf-poweff-types:something; }
            choice op-type {
              container linear-sum {}
              container linear-average {}
              container linear-max {}
              container linear-min {}
              container rolling-average {
                leaf timespan { type ietf-poweff-types:something; }
              }
              container filter-age {
                leaf min-age { type ietf-poweff-types:something; }
                leaf max-age { type ietf-poweff-types:something; }
              }
              container function {
                leaf name { type ietf-poweff-types:something; }
              }
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

6. Security Considerations

TODO Security

7. IANA Considerations

This document has no IANA actions.

8. References

8.1. Normative References

[I-D.draft-kll-yang-label-tsdb-00]
Larsson, K., "Mapping YANG Data to Label-Set Time Series",
Work in Progress, Internet-Draft, draft-kll-yang-label-

tsdb-00, 18 October 2023,
<https://datatracker.ietf.org/doc/html/draft-kll-yang-label-tsdb-00>.

[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/rfc/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/rfc/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/rfc/rfc8174>.

8.2. Informative References

[I-D.draft-claise-netconf-metadata-for-collection-03]
Claise, B., Nayyar, M., and A. R. Sesani, "Per-Node Capabilities for Optimum Operational Data Collection", Work in Progress, Internet-Draft, draft-claise-netconf-metadata-for-collection-03, 25 January 2022, <https://datatracker.ietf.org/doc/html/draft-claise-netconf-metadata-for-collection-03>.

[I-D.draft-ietf-opsawg-collected-data-manifest-01]
Claise, B., Quilbeuf, J., Lopez, D., Martinez-Casanueva, I. D., and T. Graf, "A Data Manifest for Contextualized Telemetry Data", Work in Progress, Internet-Draft, draft-ietf-opsawg-collected-data-manifest-01, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-collected-data-manifest-01>.

Acknowledgments

Kristian Larsson has provided invaluable insights, experience and validation of the design. Many thanks to the entire POWEFF team for their committment, flexibility and hard work behind this. Hat off to Benoît Claise, who inspires by the extensive work produced in IETF over the years, and in this area in particular.

Author’s Address

Jan Lindblad
Cisco
Email: jlindbla@cisco.com

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: 10 January 2024

Q. Ma
Q. Wu
Huawei
B. Lengyel
Ericsson
H. Li
HPE

9 July 2023

YANG Extension and Metadata Annotation for Immutable Flag
draft-ma-netmod-immutable-flag-08

Abstract

This document defines a way to formally document existing behavior, implemented by servers in production, on the immutability of some system configuration nodes, using a YANG "extension" and a YANG metadata annotation, both called "immutable", which are collectively used to flag which nodes are immutable.

Clients may use "immutable" statements in the YANG, and annotations provided by the server, to know beforehand when certain otherwise valid configuration requests will cause the server to return an error.

The immutable flag is descriptive, documenting existing behavior, not proscriptive, dictating server behavior.

Table of Contents

1. Introduction

This document defines a way to formally document as a YANG extension or YANG metadata an existing model handling behavior that is already allowed in YANG and has been used by multiple standard organizations and vendors. It is the aim to create one single standard solution for documenting modification restrictions on data declared as configuration, instead of the multiple existing vendor and organization specific solutions. See Appendix B for existing implementations.

YANG [RFC7950] is a data modeling language used to model both state and configuration data, based on the "config" statement. However, there exists some system configuration data that cannot be modified by the client (it is immutable), but still needs to be declared as "config true" to:

* allow configuration of data nodes under immutable lists or containers;

* place "when", "must" and "leafref" constraints between configuration and immutable data nodes.

* ensure the existence of specific list entries that are provided and needed by the system, while additional list entries can be created, modified or deleted;

Client attempts to override an immutable system configuration node are always rejected by the server [I-D.ietf-netmod-system-config]. If the server knows that it will always reject the modification because it internally think it immutable, it should document this towards the clients in a machine-readable way.

This document defines a way to formally document existing behavior, implemented by servers in production, on the immutability of some system configuration nodes, using a YANG "extension" [RFC7950] and a YANG metadata annotation [RFC7952], both called "immutable", which are collectively used to flag which nodes are immutable.

The "immutable" YANG extension is used when the behavior is independent of instances and can be described at the schema-level, while the "immutable" metadata annotation is used when the behavior must be described at the YANG "list" or "leaf-list" instance level.

Comment: Should the "immutable" metadata annotation also be returned for nodes described as immutable in the YANG schema?

Immutability is an existing model handling practice. This document does not apply to the server which does not have any immutable system configuration. While in some cases it may be needed, it also has disadvantages, therefore it SHOULD be avoided wherever possible.

The following is a list of already implemented and potential use cases.

UC1 Modeling of server capabilities

UC2 HW based auto-configuration

UC3 Predefined Access control Rules

UC4 Declaring immutable system configuration from an LNE's perspective

Appendix A describes the use cases in detail.

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.

The following terms are defined in [RFC6241]:

* configuration data

The following terms are defined in [RFC7950]:

* data node

* leaf

* leaf-list

* container

* list
* anydata
* anyxml
* interior node
* data tree

The following terms are defined in [RFC8341]:

* access operation
* write access

The following terms are defined in this document:

immutable flag: A read-only state value the server provides to describe system data it considers immutable. In schema, the immutability of data nodes is conveyed via a YANG "extension" statement. In instance representations, the immutability of data nodes is conveyed via a YANG metadata annotation. Both the extension statement and the metadata annotation are called "immutable". Together, they are alternative ways to express the same behavior.

1.2. Applicability

This document focuses on the configuration which can only be created, updated and deleted by the server, thus cannot be created, updated and deleted by the client.

The "immutable" concept defined in this document only documents existing write access restrictions to writable datastores, given the client is never allowed to edit read-only datastores. The immutable annotation information is also visible even in read-only datastores like <system> (if exists), <intended> and <operational> when a "with-immutable" parameter is carried (see Section 6.2), however this only serves as descriptive information about the instance node itself, but has no effect on the handling of the read-only datastore.

A particular data node or instance has the same immutability in all writable datastores. The immutability of data nodes is protocol and user independent. The immutability and configured value of an existing node must only change by software upgrade or hardware resource/license change.

2. Solution Overview

Immutable configuration can only be created by the system regardless of the implementation of the system configuration datastore [I-D.ietf-netmod-system-config]. If the server implements <system>, immutable configuration is present in <system>. It may be updated or deleted depending on factors like software upgrade or hardware resources/license change. Immutable configuration does not affect the contents of <running> by default.

A client may create/delete immutable nodes with same values as found in <system> (if exists) in read-write configuration datastore (e.g., <running>), which merely mean making immutable nodes visible/ invisible in read-write configuration datastore (e.g., <running>).

If a client tries to override immutable nodes with different values from ones in <system> (if exists), an error is always returned. This document allows the existing immutable system nodes to be formally documented by YANG extension or metadata annotation rather than be written as plain text in the description statement.

Servers reject client's request for updating configuration data when they internally think it immutable. The error reporting is performed immediately at an <edit-config> operation time, regardless what the target configuration datastore is. For an example of an "invalid-value" error response, see Appendix A.2.1.

Servers adding the immutable property which does not have any additional semantic meaning is discouraged. For example, a key leaf that is given a value and cannot be modified once a list entry is created.

The "immutable" flag is intended to be descriptive.

3. Use of "immutable" Flag for Different Statements

This section defines what the immutable flag means to the client for each YANG data node statement. Whilst this section describes immutability at the schema level, it applies equally to when the immutable flag is set via the metadata annotation on node instances.

Throughout this section, the word "change" refers to create, update, and delete.

3.1. The "leaf" Statement

When a leaf node is immutable, its value cannot change.

3.2. The "leaf-list" Statement

When a leaf-list data node is immutable, its value cannot change.

When the "immutable" YANG extension statement is used on a leaf-list data node, or if a leaf-list inherits immutability from an ancestor, it means that the leaf-list as a whole cannot change: entries cannot be added, removed, or reordered, in case the leaf-list is "ordered-by user".

3.3. The "container" Statement

When a container data node is immutable, its instance cannot change, unless the immutability of its descendant node is toggled.

By default, as with all interior nodes, immutability is recursively applied to descendants (see Section 4).

3.4. The "list" Statement

When a list data node is immutable, its instance cannot change, unless the immutability of its descendant node is toggled, per the description elsewhere in this section.

By default, as with all interior nodes, immutability is recursively applied to descendants (see Section 4). This statement is applicable only to the "immutable" YANG extension, as the "list" node does not itself appear in data trees.

3.5. The "anydata" Statement

When an anydata data node is immutable, its instance cannot change. Additionally, as with all interior nodes, immutability is recursively applied to descendants (see Section 4).

Descendants for anydata data node is unknown at module design time, they cannot reset the immutability state with "immutable" YANG extension.

3.6. The "anyxml" Statement

When an "anyxml" data node is immutable, its instance cannot change. Additionally, as with all interior nodes, immutability is recursively applied to descendants (see Section 4).

Descendants for anyxml data node is unknown at module design time, they cannot reset the immutability state with "immutable" YANG extension.

4. Immutability of Interior Nodes

Immutability is a conceptual operational state value that is recursively applied to descendants, which may reset the immutability state as needed, thereby affecting their descendants. There is no limit to the number of times the immutability state may change in a data tree.

For example, given the following application configuration XML snippets:

```
<application im:immutable="true">
  <name>predefined-ftp</name>
  <protocol>ftp</protocol>
  <port-number im:immutable="false">69</port-number>
</application>
```

The list entry named "predefined-ftp" is immutable="true", but its child node "port-number" has the immutable="false" (thus the client can override this value). The other child node (e.g., "protocol") not specifying its immutability explicitly inherits immutability from its parent node thus is also immutable="true".

5. "Immutable" YANG Extension

5.1. Definition

If servers always reject client modification attempts to some data node that they internally think immutable and irrelevant to its instance data, an "immutable" YANG extension can be used to formally indicate to the clients.

The "immutable" YANG extension can be a substatement to a "config true" leaf, leaf-list, container, list, anydata or anyxml statement. It has no effect if used as a substatement to a "config false" node, but can be allowed anyway.

The "immutable" YANG extension defines an argument statement named "value" which is a boolean type to indicate that whether the node is immutable or not. If the "immutable" YANG extension is not specified for a particular data node, the default immutability is the same as that of its parent node. The immutability for a top-level data node is "false" by default.

6. "Immutable" Metadata Annotation

6.1. Definition

If servers always reject clients modification to some particular instance that they internally think immutable, an "immutable" metadata annotation can be used to formally indicate to the clients.

The "immutable" metadata annotation takes as an value which is a boolean type, it is not returned unless a client explicitly requests through a "with-immutable" parameter (see Section 6.2). If the "immutable" metadata annotation for data node instances is not specified, the default "immutable" value is the same as the immutability of its parent node in the data tree. The immutable metadata annotation value for a top-level instance node is false if not specified.

Note that "immutable" metadata annotation is used to annotate data node instances. A list may have multiple entries/instances in the data tree, "immutable" can annotate some of the instances as read-only, while others are read-write.

6.2. "with-immutable" Parameter

The YANG model defined in this document (see Section 8) augments the <get-config>, <get> operation defined in RFC 6241, and the <get-data> operation defined in RFC 8526 with a new parameter named "with-immutable". When this parameter is present, it requests that the server includes "immutable" metadata annotations in its response.

This parameter may be used for read-only configuration datastores, e.g., <system> (if exists), <intended> and <operational>, but the "immutable" metadata annotation returned indicates the immutability towards read-write configuration datastores, e.g., <startup>, <candidate> and <running>. If the "immutable" metadata annotation for returned child nodes are omitted, it has the same immutability as its parent node. The immutability of top hierarchy of returned nodes is false by default.

Note that "immutable" metadata annotation is not included in a response unless a client explicitly requests them with a "with-immutable" parameter.

7. Interaction between Immutable Flag and NACM

The server rejects an operation request due to immutability when it tries to perform the operation on the request data. It happens after any access control processing, if the Network Configuration Access Control Model (NACM) [RFC8341]is implemented on a server. For example, if an operation requests to override an immutable configuration data, but the server checks the user is not authorized to perform the requested access operation on the request data, the request is rejected with an "access-denied" error.

8. YANG Module

```
<CODE BEGINS>
 file="ietf-immutable@2023-07-09.yang"
//RFC Ed.: replace XXXX with RFC number and remove this note
  module ietf-immutable {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-immutable";
    prefix im;

    import ietf-yang-metadata {
      prefix md;
    }
    import ietf-netconf {
      prefix nc;
      reference
        "RFC 6241: Network Configuration Protocol (NETCONF)";
    }
    import ietf-netconf-nmda {
      prefix ncds;
      reference
        "RFC 8526: NETCONF Extensions to Support the Network
         Management Datastore Architecture";
    }
    organization
      "IETF Network Modeling (NETMOD) Working Group";

    contact
      "WG Web: <https://datatracker.ietf.org/wg/netmod/>

       WG List: <mailto:netmod@ietf.org>

       Author: Qiufang Ma
               <mailto:maqiufang1@huawei.com>

       Author: Qin Wu
               <mailto:bill.wu@huawei.com>
```

```
         Author: Balazs Lengyel
                 <mailto:balazs.lengyel@ericsson.com>

         Author: Hongwei Li
                 <mailto:flycoolman@gmail.com>";

       description
         "This module defines a YANG extension and a metadata annotation
          both called 'immutable', to allow the server to formally
          document existing behavior on the mutability of some
          configuration nodes. Clients may use 'immutable' extension
          statements in the YANG, and annotations provided by the server
          to know beforehand when certain otherwise valid configuration
          requests will cause the server to return an error.

          Copyright (c) 2023 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 Revised
          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).

          This version of this YANG module is part of RFC HHHH
          (https://www.rfc-editor.org/info/rfcHHHH); see the RFC
          itself for full legal notices.

          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 (RFC 2119)
          (RFC 8174) when, and only when, they appear in all
          capitals, as shown here.";

       revision 2023-05-25 {
         description
           "Initial revision.";
         // RFC Ed.: replace XXXX and remove this comment
         reference
           "RFC XXXX: YANG Extension and Metadata Annotation for
            Immutable Flag";
       }

       extension immutable {
         argument value;
         description
```

```
        "If servers always reject client modification attempts to
         some data node that can only be created, modified and
         deleted by the device itself, an 'immutable' YANG extension
         can be used to formally indicate to the client.

         The statement MUST only be a substatement to a 'config true'
         leaf, leaf-list, container, list, anydata or anyxml
         statement. Zero or one immutable statement per parent
         statement is allowed.

         No substatements are allowed.

         The argument of the 'immutable' statement defines the value,
         indicating whether the node is immutable or not.

         Adding immutable of an existing immutable statement
         is non-backwards compatible changes.
         Other changes to immutable are backwards compatible.";
    }

    md:annotation immutable {
      type boolean;
      description
        "If servers always reject clients modification to some
         particular instance that can only be created, modified and
         deleted by the device itself, an 'immutable' metadata
         annotation can be used to formally indicate to the clients.
         The 'immutable' annotation indicates the immutability of an
         instantiated data node.

         The 'immutable' metadata annotation takes as a value 'true'
         or 'false'. If the 'immutable' metadata annotation for data
         node instances is not specified, the default value is false.
         Explicitly annotating instances as immutable=true has the
         same effect as not specifying this value.";
    }

    grouping with-immutable-grouping {
      description
        "define the with-immutable grouping.";
      leaf with-immutable {
        type empty;
        description
          "If this parameter is present, the server will return the
           'immutable' annotation for configuration that it
           internally thinks it immutable. When present, this
           parameter allows the server to formally document existing
           behavior on the mutability of some configuration nodes.";
```

```
        }
      }
      augment "/ncds:get-data/ncds:input" {
        description
          "Allows the server to include 'immutable' metadata
           annotations in its response to get-data operation.";
        uses with-immutable-grouping;
      }
      augment "/nc:get-config/nc:input" {
        description
          "Allows the server to include 'immutable' metadata
           annotations in its response to get-config operation.";
        uses with-immutable-grouping;
      }
      augment "/nc:get/nc:input" {
        description
          "Allows the server to include 'immutable' metadata
           annotations in its response to get operation.";
        uses with-immutable-grouping;
      }
    }
<CODE ENDS>
```

9. IANA Considerations

9.1. The "IETF XML" Registry

This document registers one XML namespace URN in the 'IETF XML registry', following the format defined in [RFC3688].

```
URI: urn:ietf:params:xml:ns:yang:ietf-immutable
Registrant Contact: The IESG.
XML: N/A, the requested URIs are XML namespaces.
```

9.2. The "YANG Module Names" Registry

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC6020].

```
name: ietf-immutable
prefix: im
namespace: urn:ietf:params:xml:ns:yang:ietf-immutable
RFC: XXXX
// RFC Ed.: replace XXXX and remove this comment
```

10. Security Considerations

The YANG module specified in this document defines a YANG extension and a metadata Annotation. These can be used to further restrict write access but cannot be used to extend access rights.

This document does not define any protocol-accessible data nodes.

Since immutable information is tied to applied configuration values, it is only accessible to clients that have the permissions to read the applied configuration values.

The security considerations for the Defining and Using Metadata with YANG (see Section 9 of [RFC7952]) apply to the metadata annotation defined in this document.

Acknowledgements

Thanks to Kent Watsen, Andy Bierman, Robert Wilton, Jan Lindblad, Reshad Rahman, Anthony Somerset, Lou Berger, Joe Clarke, Scott Mansfield for reviewing, and providing important input to, this document.

References

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, <https://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, <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>.

[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>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[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>.

Informative References

[I-D.ietf-netmod-system-config] Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-02, 4 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-02>.

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

[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>.

[RFC8530] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", RFC 8530, DOI 10.17487/RFC8530, March 2019, <https://www.rfc-editor.org/info/rfc8530>.

[TR-531] ONF, "UML to YANG Mapping Guidelines, <https://wiki.opennetworking.org/download/attachments/376340494/Draft_TR-531_UML-YANG_Mapping_Gdls_v1.1.03.docx?version=5&modificationDate=1675432243513&api=v2>", February 2023.

[TS28.623] 3GPP, "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Solution Set (SS) definitions, <https://www.3gpp.org/ftp/Specs/archive/28_series/28.623/28623-i02.zip>".

[TS32.156] 3GPP, "Telecommunication management; Fixed Mobile Convergence (FMC) Model repertoire, <https://www.3gpp.org/ftp/Specs/archive/32_series/32.156/32156-h10.zip>".

Appendix A. Detailed Use Cases

A.1. UC1 - Modeling of server capabilities

System capabilities might be represented as system-defined data nodes in the model. Configurable data nodes might need constraints specified as "when", "must" or "path" statements to ensure that configuration is set according to the system's capabilities. E.g.,

* A timer can support the values 1,5,8 seconds. This is defined in the leaf-list 'supported-timer-values'.

* When the configurable 'interface-timer' leaf is set, it should be ensured that one of the supported values is used. The natural solution would be to make the 'interface-timer' a leaf-ref pointing at the 'supported-timer-values'.

However, this is not possible as 'supported-timer-values' must be read-only thus config=false while 'interface-timer' must be writable thus config=true. According to the rules of YANG it is not allowed to put a constraint between config true and false data nodes.

The solution is that the supported-timer-values data node in the YANG Model shall be defined as "config true" and shall also be marked with the "immutable" extension making it unchangable. After this the 'interface-timer' shall be defined as a leaf-ref pointing at the 'supported-timer-values'.

A.2. UC2 - HW based auto-configuration - Interface Example

This section shows how to use immutable YANG extension to mark some data node as immutable.

When an interface is physically present, the system will create an interface entry automatically with valid name and type values in <system> (if exists, see [I-D.ietf-netmod-system-config]). The system-generated data is dependent on and must represent the HW present, and as a consequence must not be changed by the client. The data is modelled as "config true" and should be marked as immutable.

Seemingly an alternative would be to model the list and these leaves as "config false", but that does not work because:

* The list cannot be marked as "config false", because it needs to contain configurable child nodes, e.g., ip-address or enabled;

* The key leaf (name) cannot be marked as "config false" as the list itself is config true;

* The type cannot be marked "config false", because we MAY need to reference the type to make different configuration nodes conditionally available.

The immutability of the data is the same for all interface instances, thus following fragment of a fictional interface module including an "immutable" YANG extension can be used:

```
container interfaces {
  list interface {
    key "name";
    leaf name {
      type string;
    }
    leaf type {
      im:immutable;
      type identityref {
        base ianaift:iana-interface-type;
      }
      mandatory true;
    }
    leaf mtu {
      type uint16;
    }
    leaf-list ip-address {
      type inet:ip-address;
    }
  }
}
```

Note that the "name" leaf is defined as a list key which can never been modified for a particular list entry, there is no need to mark "name" as immutable.

A.2.1. Error Response to Client Updating the Value of an Interface Type

This section shows an example of an error response due to the client modifying an immutable configuration.

Assume the system creates an interface entry named "eth0" given that an inerface is inserted into the device. If a client tries to change the type of an interface to a value that doesn't match the real type of the interface used by the system, the request will be rejected by the server:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interface xc:operation="merge"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
        <name>eth0</name>
        <type>ianaift:tunnel</type>
      </interface>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:t="http://example.com/schema/1.2/config">
      /interfaces/interface[name="eth0"]/type
    </error-path>
    <error-message xml:lang="en">
      Invalid type for interface eth0
    </error-message>
  </rpc-error>
</rpc-reply>
```

A.3. UC3 - Predefined Access control Rules

Setting up detailed rules for access control is a complex task. (see [RFC8341]) A vendor may provide an initial, predefined set of groups and related access control rules so that the customer can use access control out-of-the-box. The customer may continue using these predefined rules or may add his own groups and rules. The predefined groups shall not be removed or altered guaranteeing that access control remains usable and basic functions e.g., a system-security-administrator are always available.

The system needs to protect the predefined groups and rules, however, the list "groups" or the list "rule-list" cannot be marked as config=false or with the "immutable" extension in the YANG model because that would prevent the customer adding new entries. Still it

would be good to notify the client in a machine readable way that the predefined entries cannot be modified. When the client retrieves access control data the immutable="true" metadata annotation should be used to indicate to the client that the predefined groups and rules cannot be modified.

A.4. UC4 - Declaring immutable system configuration from an LNE's perspective

An LNE (logical network element) is an independently managed virtual network device made up of resources allocated to it from its host or parent network device [RFC8530]. The host device may allocate some resources to an LNE, which from an LNE's perspective is provided by the system and may not be modifiable.

For example, a host may allocate an interface to an LNE with a valid MTU value as its management interface, so that the allocated interface should then be accessible as the LNE-specific instance of the interface model. The assigned MTU value is system-created and immutable from the context of the LNE.

Appendix B. Existing implementations

There are already a number of full or partial implementations of immutability.

3GPP TS 32.156 [TS32.156] and 28.623 [TS28.623]: Requirements and a partial solution

ITU-T using ONF TR-531[TR-531] concept on information model level but no YANG representation.

Ericsson: requirements and solution

YumaPro: requirements and solution

Nokia: partial requirements and solution

Huawei: partial requirements and solution

Cisco using the concept at least in some YANG modules

Junos OS provides a hidden and immutable configuration group called junos-defaults

Appendix C. Changes between revisions

Note to RFC Editor (To be removed by RFC Editor)

v06 - v07

* Use a Boolean type for the immutable value in YANG extension and metadata annotation

* Define a "with-immutable" parameter and state that immutable metadata annotation is not included in a response unless a client explicitly requests them with a "with-immutable" parameter

* reword the abstract and related introduction section to highlight immutable flag is descriptive

* Add a new section to define immutability of interior nodes, and merge with "Inheritance of Immutable configuration" section

* Add a new section to define what the immutable flag means for each YANG data node

* Define the "immutable flag" term.

* Add an item in the open issues tracking: Should the "immutable" metadata annotation also be returned for nodes described as immutable in the YANG schema so that there is a single source of truth?

v05 - v06

* Remove immutable BGP AS number case

* Fix nits

v04 - v05

* Emphasized that the proposal tries to formally document existing allowed behavior

* Reword the abstract and introduction sections;

* Restructure the document;

* Simplified the interface example in Appendix;

* Add immutable BGP AS number and peer-type configuration example.

* Added temporary section in Appendix B about list of existing non-standard solutions

* Clarified inheritance of immutability

* Clarified that this draft is not dependent on the existence of the <system> datastore.

v03 - v04

* Clarify how immutable flag interacts with NACM mechanism.

v02 - v03

* rephrase and avoid using "server MUST reject" statement, and try to clarify that this documents aims to provide visibility into existing immutable behavior;

* Add a new section to discuss the inheritance of immutability;

* Clarify that deletion to an immutable node in <running> which is instantiated in <system> and copied into <running> should always be allowed;

* Clarify that write access restriction due to general YANG rules has no need to be marked as immutable.

* Add an new section named "Acknowledgements";

* editoral changes.

v01 - v02

* clarify the relation between the creation/deletion of the immutable data node with its parent data node;

* Add a "TODO" comment about the inheritance of the immutable property;

* Define that the server should reject write attempt to the immutable data node at an <edit-config> operation time, rather than waiting until a <commit> or <validate> operation takes place;

v00 - v01

* Added immutable extension

* Added new use-cases for immutable extension and annotation

* Added requirement that an update that means no effective change should always be allowed

* Added clarification that immutable is only applied to read-write datastore

* Narrowed the applied scope of metadata annotation to list/leaf-list instances

Appendix D. Open Issues tracking

* Should the "immutable" metadata annotation also be returned for nodes described as immutable in the YANG schema so that there is a single source of truth?

Authors' Addresses

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Balazs Lengyel
Ericsson
Email: balazs.lengyel@ericsson.com

Hongwei Li
HPE
Email: flycoolman@gmail.com