Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

R. Wilton, Ed.
D. Ball
T. Singh
Cisco Systems
S. Sivaraj
Juniper Networks

July 13, 2020

Common Interface Extension YANG Data Models
draft-ietf-netmod-intf-ext-yang-09

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 January 14, 2021.

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 behaviour.

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 implementors of the YANG module will decide to support all features. Hence separate feature keywords are defined for each logically discrete feature to allow implementors 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 RFC 2119 [RFC2119] RFC 8174 [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 carrier delay 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 carrier-delay {carrier-delay}?
    |  +--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}?
```

2.1. Carrier Delay

The carrier delay 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 carrier delay 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 carrier delay 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 carrier delay 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 augument 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
```

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@2019-11-04.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) 2019 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.

       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 2019-11-04 {
      description
        "Initial revision.";

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

    feature carrier-delay {
      description
        "This feature indicates that configurable interface carrier
         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 Carrier Delay";
    }

    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 carrier-delay 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 Carrier Delay feature.
       */
      container carrier-delay {
        if-feature "carrier-delay";
        description
          "Holds carrier delay 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 carrier signal must be continuously
             down before the interface state is brought down. If not
             configured, the behaviour on loss of carrier 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 carrier signal must be continuously present and error
            free before the interface state is allowed to transition
            from down to up.  If not configured, the behaviour 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 carrier 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 carrier-delay timer.";
        }
        leaf timer-running {
          type enumeration {
            enum none {
              description
                "No carrier delay timer is running.";
            }
            enum up {
              description
                "Carrier-delay up timer is running.  The underlying
                 carrier state is up, but interface state is not
                 reported as up.";
            }
            enum down {
              description
                "Carrier-delay down timer is running.  Interface state
                 is reported as up, but the underlying carrier state is
                 actually down.";
            }
          }
          config false;
          description
            "Reports whether a carrier delay timer is actively running,
             in which case the interface state does not match the
             underlying carrier state.";
        }

        reference "RFC XXXX, Section 2.1 Carrier Delay";
      }
```

```
      /*
       * 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, '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: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 demux 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 to the interface or any child sub-interface.  E.g.,
           a packet 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 [IEEE802.3.2-2019].

```
<CODE BEGINS> file "ietf-if-ethernet-like@2019-11-04.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) 2019 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 2019-11-04 {
      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).";
        }
      }
    }
    <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 carrier delay 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:carrier-delay>
      <if-ext:down>0</if-ext:down>
      <if-ext:up>50</if-ext:up>
    </if-ext:carrier-delay>
  </interface>
</interfaces>
```

The following example shows explicit carrier 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:carrier-delay>
        <if-ext:down>50</if-ext:down>
        <if-ext:up>1000</if-ext:up>
      </if-ext:carrier-delay>
    </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. ChangeLog

XXX, RFC Editor, please delete this change log before publication.

8.1. Version -09

- Fixed IANA section.

8.2. Version -08

- Initial updates after WG LC comments.

8.3. Version -07

- Minor editorial updates

8.4. Version -06

- Remove reservable-bandwidth, based on Acee's suggestion
- Add examples
- Add additional state parameters for carrier-delay and dampening

8.5. Version -05

- Incorporate feedback from Andy Bierman

8.6. Version -04

- Incorporate feedback from Lada, some comments left as open issues.

8.7. Version -03

- Fixed incorrect module name references, and updated tree output

8.8. Version -02

- Minor changes only: Fix errors in when statements, use derived-from-or-self() for future proofing.

9. IANA Considerations

9.1. YANG Module Registrations

The following YANG modules are requested to be registred 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.

10. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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

10.1. 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/carrier-delay/down
- /if:interfaces/if:interface/carrier-delay/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

10.2. 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

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>.

[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>.

11.2. Informative References

[I-D.ietf-netmod-sub-intf-vlan-model] Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaraj, "Sub-interface VLAN YANG Data Models", draft-ietf-netmod-sub-intf-vlan-model-06 (work in progress), November 2019.

[IEEE802.3.2-2019] IEEE WG802.3 - Ethernet Working Group, "IEEE 802.3.2-2019", 2019.

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

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

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

[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>.

Authors' Addresses

Robert Wilton (editor)
Cisco Systems

Email: rwilton@cisco.com

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@cisco.com

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net

Network Working Group
Internet-Draft
Obsoletes: 6991 (if approved)
Intended status: Standards Track
Expires: January 7, 2021

J. Schoenwaelder, Ed.
Jacobs University

July 6, 2020

Common YANG Data Types
draft-ietf-netmod-rfc6991-bis-04

Abstract

This document introduces a collection of common data types to be used with the YANG data modeling language. This document obsoletes RFC 6991.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 7, 2021.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC7950] is a data modeling language used to model configuration and state data manipulated by the Network Configuration Protocol (NETCONF) [RFC6241]. The YANG language supports a small set of built-in data types and provides mechanisms to derive other types from the built-in types.

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

This document adds new type definitions to the YANG modules and obsoletes [RFC6991]. For further details, see the revision statements of the YANG modules in Section 3 and Section 4 and the summary in Appendix A.

This document uses the YANG terminology defined in Section 3 of [RFC7950].

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.

2. Overview

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

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

YANG type	Equivalent SMIv2 type (module)
counter32	Counter32 (SNMPv2-SMI)
zero-based-counter32	ZeroBasedCounter32 (RMON2-MIB)
counter64	Counter64 (SNMPv2-SMI)
zero-based-counter64	ZeroBasedCounter64 (HCNUM-TC)
gauge32	Gauge32 (SNMPv2-SMI)
gauge64	CounterBasedGauge64 (HCNUM-TC)
object-identifier	-
object-identifier-128	OBJECT IDENTIFIER
date-and-time	-
date	-
time	-
hours32	-
minutes32	-
seconds32	-
centiseconds32	TimeInterval (SNMPv2-TC)
milliseconds32	-
microseconds32	-
microseconds64	-
nanoseconds32	-
nanoseconds64	-
timeticks	TimeTicks (SNMPv2-SMI)
timestamp	TimeStamp (SNMPv2-TC)
phys-address	PhysAddress (SNMPv2-TC)
mac-address	MacAddress (SNMPv2-TC)
xpath1.0	-
hex-string	-
uuid	-
dotted-quad	-
yang-identifier	-
revision-identifier	-
node-instance-identifier	-

Table 1: ietf-yang-types

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

YANG type	Equivalent SMIv2 type (module)
ip-version	InetVersion (INET-ADDRESS-MIB)
dscp	Dscp (DIFFSERV-DSCP-TC)
ipv6-flow-label	IPv6FlowLabel (IPV6-FLOW-LABEL-MIB)
port-number	InetPortNumber (INET-ADDRESS-MIB)
as-number	InetAutonomousSystemNumber (INET-ADDRESS-MIB)
ip-address	-
ipv4-address	-
ipv6-address	-
ip-address-no-zone	-
ipv4-address-no-zone	-
ipv6-address-no-zone	-
ip-prefix	-
ipv4-prefix	-
ipv6-prefix	-
domain-name	-
host	-
uri	Uri (URI-TC-MIB)
email-address	-

Table 2: ietf-inet-types

3. Core YANG Derived Types

The ietf-yang-types YANG module references [IEEE802], [ISO9834-1], [RFC2578], [RFC2579], [RFC2856], [RFC3339], [RFC4122], [RFC4502], [RFC5322], [RFC7950], [XPATH], and [XSD-TYPES].

<CODE BEGINS> file "ietf-yang-types@2020-07-06.yang"

```
module ietf-yang-types {

  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-types";
  prefix "yang";

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

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

    Editor:   Juergen Schoenwaelder
              <mailto:j.schoenwaelder@jacobs-university.de>";

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

    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.

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

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

  revision 2020-07-06 {
    description
```

```
      "This revision adds the following new data types:
       - date, time
       - hours32, minutes32, seconds32, centiseconds32, milliseconds32,
       - microseconds32, microseconds64, nanoseconds32, nanoseconds64
       - revision-identifier, node-instance-identifier";
    reference
     "RFC XXXX: Common YANG Data Types";
  }

  revision 2013-07-15 {
    description
     "This revision adds the following new data types:
      - yang-identifier
      - hex-string
      - uuid
      - dotted-quad";
    reference
     "RFC 6991: Common YANG Data Types";
  }

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

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

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

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

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

      In the value set and its semantics, this type is equivalent
      to the Counter32 type of the SMIv2.";
    reference
     "RFC 2578: Structure of Management Information Version 2
                (SMIv2)";
  }

  typedef zero-based-counter32 {
    type yang:counter32;
    default "0";
    description
     "The zero-based-counter32 type represents a counter32
      that has the defined 'initial' value zero.

      A schema node instance of this type will be set to zero (0)
      on creation and will thereafter increase monotonically until
      it reaches a maximum value of 2^32-1 (4294967295 decimal),
      when it wraps around and starts increasing again from zero.

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

      In the value set and its semantics, this type is equivalent
      to the ZeroBasedCounter32 textual convention of the SMIv2.";
    reference
      "RFC 4502: Remote Network Monitoring Management Information
                 Base Version 2";
  }

  typedef counter64 {
    type uint64;
    description
     "The counter64 type represents a non-negative integer
      that monotonically increases until it reaches a
      maximum value of 2^64-1 (18446744073709551615 decimal),
      when it wraps around and starts increasing again from zero.

      Counters have no defined 'initial' value, and thus, a
      single value of a counter has (in general) no information
      content.  Discontinuities in the monotonically increasing
```

```
      value normally occur at re-initialization of the
      management system, and at other times as specified in the
      description of a schema node using this type.  If such
      other times can occur, for example, the instantiation of
      a schema node of type counter64 at times other than
      re-initialization, then a corresponding schema node
      should be defined, with an appropriate type, to indicate
      the last discontinuity.

      The counter64 type should not be used for configuration
      schema nodes.  A default statement SHOULD NOT be used in
      combination with the type counter64.

      In the value set and its semantics, this type is equivalent
      to the Counter64 type of the SMIv2.";
    reference
     "RFC 2578: Structure of Management Information Version 2
                (SMIv2)";
  }

  typedef zero-based-counter64 {
    type yang:counter64;
    default "0";
    description
     "The zero-based-counter64 type represents a counter64 that
      has the defined 'initial' value zero.

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

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

      In the value set and its semantics, this type is equivalent
      to the ZeroBasedCounter64 textual convention of the SMIv2.";
    reference
     "RFC 2856: Textual Conventions for Additional High Capacity
                Data Types";
  }

  typedef gauge32 {
```

```
    type uint32;
    description
     "The gauge32 type represents a non-negative integer, which
      may increase or decrease, but shall never exceed a maximum
      value, nor fall below a minimum value.  The maximum value
      cannot be greater than 2^32-1 (4294967295 decimal), and
      the minimum value cannot be smaller than 0.  The value of
      a gauge32 has its maximum value whenever the information
      being modeled is greater than or equal to its maximum
      value, and has its minimum value whenever the information
      being modeled is smaller than or equal to its minimum value.
      If the information being modeled subsequently decreases
      below (increases above) the maximum (minimum) value, the
      gauge32 also decreases (increases).

      In the value set and its semantics, this type is equivalent
      to the Gauge32 type of the SMIv2.";
    reference
     "RFC 2578: Structure of Management Information Version 2
                (SMIv2)";
  }

  typedef gauge64 {
    type uint64;
    description
     "The gauge64 type represents a non-negative integer, which
      may increase or decrease, but shall never exceed a maximum
      value, nor fall below a minimum value.  The maximum value
      cannot be greater than 2^64-1 (18446744073709551615), and
      the minimum value cannot be smaller than 0.  The value of
      a gauge64 has its maximum value whenever the information
      being modeled is greater than or equal to its maximum
      value, and has its minimum value whenever the information
      being modeled is smaller than or equal to its minimum value.
      If the information being modeled subsequently decreases
      below (increases above) the maximum (minimum) value, the
      gauge64 also decreases (increases).

      In the value set and its semantics, this type is equivalent
      to the CounterBasedGauge64 SMIv2 textual convention defined
      in RFC 2856";
    reference
     "RFC 2856: Textual Conventions for Additional High Capacity
                Data Types";
  }

  /*** collection of identifier-related types ***/
```

```
   typedef object-identifier {
     type string {
       pattern '(([0-1](\.[1-3]?[0-9]))|(2\.(0|([1-9]\d*))))'
             + '(\.(0|([1-9]\d*)))*';
     }
     description
      "The object-identifier type represents administratively
       assigned names in a registration-hierarchical-name tree.

       Values of this type are denoted as a sequence of numerical
       non-negative sub-identifier values.  Each sub-identifier
       value MUST NOT exceed 2^32-1 (4294967295).  Sub-identifiers
       are separated by single dots and without any intermediate
       whitespace.

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

       Although the number of sub-identifiers is not limited,
       module designers should realize that there may be
       implementations that stick with the SMIv2 limit of 128
       sub-identifiers.

       This type is a superset of the SMIv2 OBJECT IDENTIFIER type
       since it is not restricted to 128 sub-identifiers.  Hence,
       this type SHOULD NOT be used to represent the SMIv2 OBJECT
       IDENTIFIER type; the object-identifier-128 type SHOULD be
       used instead.";
     reference
      "ISO9834-1: Information technology -- Open Systems
       Interconnection -- Procedures for the operation of OSI
       Registration Authorities: General procedures and top
       arcs of the ASN.1 Object Identifier tree";
   }

   typedef object-identifier-128 {
     type object-identifier {
       pattern '\d*(\.\d*){1,127}';
     }
     description
      "This type represents object-identifiers restricted to 128
       sub-identifiers.
```

```
      In the value set and its semantics, this type is equivalent
      to the OBJECT IDENTIFIER type of the SMIv2.";
    reference
     "RFC 2578: Structure of Management Information Version 2
                (SMIv2)";
  }

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

  typedef date-and-time {
    type string {
      pattern '\d{4}-(1[0-2]|0[1-9])-(0[1-9]|[1|2][0-9]|3[0-1])'
            + 'T(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9](\.\d+)?'
            + '(Z|[\+\-]((1[0-3]|0[0-9]):([0-5][0-9])|14:00))?';
    }
    description
     "The date-and-time type is a profile of the ISO 8601
      standard for representation of dates and times using the
      Gregorian calendar.  The profile is defined by the
      date-time production in Section 5.6 of RFC 3339.

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

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

      (b) The time-offset -00:00 indicates that the date-and-time
          value is reported in UTC and that the local time zone
          reference point is unknown. The time-offsets +00:00 and Z
          both indicate that the date-and-time value is reported in
          UTC and that the local time reference point is UTC (see RFC
          3339 section 4.3).

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

      This type is not equivalent to the DateAndTime textual
      convention of the SMIv2 since RFC 3339 uses a different
      separator between full-date and full-time and provides
      higher resolution of time-secfrac.

      The canonical format for date-and-time values with a known time
      zone uses a numeric time zone offset that is calculated using
      the device's configured known offset to UTC time.  A change of
      the device's offset to UTC time will cause date-and-time values
      to change accordingly.  Such changes might happen periodically
```

```
      in case a server follows automatically daylight saving time
      (DST) time zone offset changes.  The canonical format for
      date-and-time values with an unknown time zone (usually
      referring to the notion of local time) uses the time-offset
      -00:00, i.e., date-and-time values must be reported in UTC.";
    reference
     "RFC 3339: Date and Time on the Internet: Timestamps
      RFC 2579: Textual Conventions for SMIv2
      XSD-TYPES: XML Schema Part 2: Datatypes Second Edition";
  }

  typedef date {
    type string {
      pattern '\d{4}-(1[0-2]|0[1-9])-(0[1-9]|[1|2][0-9]|3[0-1])'
            + '(Z|[\+\-]((1[0-3]|0[0-9]):([0-5][0-9])|14:00))?';
    }
    description
     "The date type represents a time-interval of the length
      of a day, i.e., 24 hours.

      The date type is compatible with the date XML schema
      type with the following notable exceptions:

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

      (b) The time-offset -00:00 indicates that the date value is
          reported in UTC and that the local time zone reference point
          is unknown. The time-offsets +00:00 and Z both indicate that
          the date value is reported in UTC and that the local time
          reference point is UTC (see RFC 3339 section 4.3).

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

      The canonical format for date values with a known time
      zone uses a numeric time zone offset that is calculated using
      the device's configured known offset to UTC time.  A change of
      the device's offset to UTC time will cause date values
      to change accordingly.  Such changes might happen periodically
      in case a server follows automatically daylight saving time
      (DST) time zone offset changes.  The canonical format for
      date values with an unknown time zone (usually referring
      to the notion of local time) uses the time-offset -00:00,
      i.e., date values must be reported in UTC.";
    reference
     "RFC 3339: Date and Time on the Internet: Timestamps
```

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

  /*
   * DISCUSS:
   * - XML schema seems to use a different canonical format, we
   *   need to take a closer look how to define the canonical format
   *   given that a date really identifies a 24 hour interval and
   *   what XSD means with 'interval midpoint'.
   */

  typedef time {
    type string {
      pattern '(0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9](\.\d+)?'
            + '(Z|[\+\-]((1[0-3]|0[0-9]):([0-5][0-9])|14:00))?';
    }
    description
     "The time type represents an instance of time of zero-duration
      that recurs every day.

      The time type is compatible with the time XML schema
      type with the following notable exceptions:

      (a) The time-offset -00:00 indicates that the time value is
          reported in UTC and that the local time zone reference point
          is unknown. The time-offsets +00:00 and Z both indicate that
          the time value is reported in UTC and that the local time
          reference point is UTC (see RFC 3339 section 4.3).

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

      The canonical format for time values with a known time
      zone uses a numeric time zone offset that is calculated using
      the device's configured known offset to UTC time.  A change of
      the device's offset to UTC time will cause time values
      to change accordingly.  Such changes might happen periodically
      in case a server follows automatically daylight saving time
      (DST) time zone offset changes.  The canonical format for
      time values with an unknown time zone (usually referring
      to the notion of local time) uses the time-offset -00:00,
      i.e., time values must be reported in UTC.";
    reference
     "RFC 3339: Date and Time on the Internet: Timestamps
      XSD-TYPES: XML Schema Part 2: Datatypes Second Edition";
  }
```

```
   typedef hours32 {
     type int32;
     units "hours";
     description
         "A period of time, measured in units of hours.

          The maximum time period that can be expressed is in the
          range [89478485 days 08:00:00 to 89478485 days 07:00:00].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
   }

   typedef minutes32 {
     type int32;
     units "minutes";
     description
         "A period of time, measured in units of minutes.

          The maximum time period that can be expressed is in the
          range [-1491308 days 2:08:00 to 1491308 days 2:07:00].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
   }

   typedef seconds32 {
     type int32;
     units "seconds";
     description
         "A period of time, measured in units of seconds.

          The maximum time period that can be expressed is in the
          range [-24855 days 03:14:08 to 24855 days 03:14:07].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
   }

   typedef centiseconds32 {
     type int32;
     units "centiseconds";
     description
         "A period of time, measured in units of 10^-2 seconds.
```

```
          The maximum time period that can be expressed is in the
          range [-248 days 13:13:56 to 248 days 13:13:56].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
    }

    typedef milliseconds32 {
      type int32;
      units "milliseconds";
      description
          "A period of time, measured in units of 10^-3 seconds.

          The maximum time period that can be expressed is in the
          range [-24 days 20:31:23 to 24 days 20:31:23].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
    }

    typedef microseconds32 {
      type int32;
      units "microseconds";
      description
          "A period of time, measured in units of 10^-6 seconds.

          The maximum time period that can be expressed is in the
          range [-00:35:47 to 00:35:47].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
    }

    typedef microseconds64 {
      type int64;
      units "microseconds";
      description
          "A period of time, measured in units of 10^-6 seconds.

          The maximum time period that can be expressed is in the
          range [-106751991 days 04:00:54 to 106751991 days 04:00:54].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
```

```
   }

   typedef nanoseconds32 {
     type int32;
     units "nanoseconds";
     description
         "A period of time, measured in units of 10^-9 seconds.

          The maximum time period that can be expressed is in the
          range [-00:00:02 to 00:00:02].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
   }

   typedef nanoseconds64 {
     type int64;
     units "nanoseconds";
     description
         "A period of time, measured in units of 10^-9 seconds.

          The maximum time period that can be expressed is in the
          range [-106753 days 23:12:44 to 106752 days 0:47:16].

          This type should be range restricted in situations
          where only non-negative time periods are desirable,
          (i.e., range '0..max').";
   }

   typedef timeticks {
     type uint32;
     description
      "The timeticks type represents a non-negative integer that
       represents the time, modulo 2^32 (4294967296 decimal), in
       hundredths of a second between two epochs.  When a schema
       node is defined that uses this type, the description of
       the schema node identifies both of the reference epochs.

       In the value set and its semantics, this type is equivalent
       to the TimeTicks type of the SMIv2.";
     reference
      "RFC 2578: Structure of Management Information Version 2
                 (SMIv2)";
   }

   typedef timestamp {
     type yang:timeticks;
```

```
    description
     "The timestamp type represents the value of an associated
      timeticks schema node instance at which a specific occurrence
      happened.  The specific occurrence must be defined in the
      description of any schema node defined using this type.  When
      the specific occurrence occurred prior to the last time the
      associated timeticks schema node instance was zero, then the
      timestamp value is zero.

      Note that this requires all timestamp values to be reset to
      zero when the value of the associated timeticks schema node
      instance reaches 497+ days and wraps around to zero.

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

      In the value set and its semantics, this type is equivalent
      to the TimeStamp textual convention of the SMIv2.";
    reference
     "RFC 2579: Textual Conventions for SMIv2";
  }

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

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

      In the value set and its semantics, this type is equivalent
      to the PhysAddress textual convention of the SMIv2.";
    reference
     "RFC 2579: Textual Conventions for SMIv2";
  }

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

```
       In the value set and its semantics, this type is equivalent
       to the MacAddress textual convention of the SMIv2.";
     reference
      "IEEE 802: IEEE Standard for Local and Metropolitan Area
                 Networks: Overview and Architecture
       RFC 2579: Textual Conventions for SMIv2";
   }

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

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

       When a schema node is defined that uses this type, the
       description of the schema node MUST specify the XPath
       context in which the XPath expression is evaluated.";
     reference
      "XPATH: XML Path Language (XPath) Version 1.0";
   }

   /*
    * DISCUSS:
    * - How do we deal with xpath expressions in other encodings
    *   such as JSON. Do we assume an xpath context populated with
    *   module names such that module names can be used to qualify
    *   path expressions. This may need discussion and/or a new
    *   definition.
    * - This interacts with the definition of node-instance-identifier.
    */

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

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

   typedef uuid {
     type string {
       pattern '[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-'
             + '[0-9a-fA-F]{4}-[0-9a-fA-F]{12}';
```

```
    }
    description
     "A Universally Unique IDentifier in the string representation
      defined in RFC 4122.  The canonical representation uses
      lowercase characters.

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

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

  /*** collection of YANG specific types ***/

  typedef yang-identifier {
    type string {
      length "1..max";
      pattern '[a-zA-Z_][a-zA-Z0-9\-_.]*';
      pattern '.|..|[^xX].*|.[^mM].*|..[^lL].*';
    }
    description
      "A YANG identifier string as defined by the 'identifier'
       rule in Section 12 of RFC 6020. An identifier must
       start with an alphabetic character or an underscore
       followed by an arbitrary sequence of alphabetic or
       numeric characters, underscores, hyphens, or dots.

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

```
   typedef revision-identifier {
     type date {
       pattern '\d{4}-(1[0-2]|0[1-9])-(0[1-9]|[1|2][0-9]|3[0-1])';
     }
     description
      "Represents a specific revision of a YANG module by means of
       a date value without a time zone.";
   }

   typedef node-instance-identifier {
     type xpath1.0;
     description
      "Path expression used to represent a data node, action,
       or notification instance-identifier string.

       A node-instance-identifier value is an unrestricted
       YANG instance-identifier expression or the special
       value '/', which refers to the entire accessible tree.

       All the rules for instance-identifier apply, except that
       predicates for keys are optional.  If a key predicate is
       missing, then the node-instance-identifier represents all
       possible server instances for that key.

       This XML Path Language (XPath) expression is evaluated in the
       following context:

          o  The set of namespace declarations are those in scope on
             the leaf element where this type is used.

          o  The set of variable bindings contains one variable,
             'USER', which contains the name of the user of the
             current session.

          o  The function library is the core function library, but
             note that due to the syntax restrictions of an
             instance-identifier, no functions are allowed.

          o  The context node is the root node in the data tree.

       The accessible tree includes actions and notifications tied
       to data nodes.";
   }

   /*
    * DISCUSS:
    * - This is taken from RFC 8341 and the idea is that this definition
    *   is useful without requiring a dependency on NACM
```

```
    * - What does the second bullet actually do? Do we keep this?
    * - This interacts with the definition of xpath1.0.
    */

   /* DISCUSS:
    * - It was suggested to add types for longitude, latitude,
    *   postal code, country-code. Do we go there or do we leave
    *   these for other modules to define? It seems such definitions
    *   should go into draft-ietf-netmod-geo-location.
    */

   /* DISCUSS:
    * - It was suggested to add percentage types but they tend to differ
    *   widely. However, percentages are also widely used.
    */
 }

   <CODE ENDS>
```

4. Internet-Specific Derived Types

The ietf-inet-types YANG module references [RFC0768], [RFC0791], [RFC0793], [RFC0952], [RFC1034], [RFC1123], [RFC1930], [RFC2317], [RFC2460], [RFC2474], [RFC2780], [RFC2782], [RFC3289], [RFC3305], [RFC3595], [RFC3986], [RFC4001], [RFC4007], [RFC4271], [RFC4291], [RFC4340], [RFC4592] [RFC4960], [RFC5017], [RFC5890], [RFC5952], and [RFC6793].

<CODE BEGINS> file "ietf-inet-types@2020-07-06.yang"

```
module ietf-inet-types {

  namespace "urn:ietf:params:xml:ns:yang:ietf-inet-types";
  prefix "inet";

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

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

    Editor:   Juergen Schoenwaelder
              <mailto:j.schoenwaelder@jacobs-university.de>";

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

    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.

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

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

```
    revision 2020-07-06 {
      description
       "This revision adds the following new data types:
        - ip-address-and-prefix
        - ipv4-address-and-prefix
        - ipv6-address-and-prefix
        - email-address";
      reference
       "RFC XXXX: Common YANG Data Types";
    }

    revision 2013-07-15 {
      description
       "This revision adds the following new data types:
        - ip-address-no-zone
        - ipv4-address-no-zone
        - ipv6-address-no-zone";
      reference
       "RFC 6991: Common YANG Data Types";
    }

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

    /*** collection of types related to protocol fields ***/

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

```
      }
      description
       "This value represents the version of the IP protocol.

        In the value set and its semantics, this type is equivalent
        to the InetVersion textual convention of the SMIv2.";
      reference
       "RFC  791: Internet Protocol
        RFC 2460: Internet Protocol, Version 6 (IPv6) Specification
        RFC 4001: Textual Conventions for Internet Network Addresses";
    }

    typedef dscp {
      type uint8 {
        range "0..63";
      }
      description
       "The dscp type represents a Differentiated Services Code Point
        that may be used for marking packets in a traffic stream.

        In the value set and its semantics, this type is equivalent
        to the Dscp textual convention of the SMIv2.";
      reference
       "RFC 3289: Management Information Base for the Differentiated
                  Services Architecture
        RFC 2474: Definition of the Differentiated Services Field
                  (DS Field) in the IPv4 and IPv6 Headers
        RFC 2780: IANA Allocation Guidelines For Values In
                  the Internet Protocol and Related Headers";
    }

    typedef ipv6-flow-label {
      type uint32 {
        range "0..1048575";
      }
      description
       "The ipv6-flow-label type represents the flow identifier or
        Flow Label in an IPv6 packet header that may be used to
        discriminate traffic flows.

        In the value set and its semantics, this type is equivalent
        to the IPv6FlowLabel textual convention of the SMIv2.";
      reference
       "RFC 3595: Textual Conventions for IPv6 Flow Label
        RFC 2460: Internet Protocol, Version 6 (IPv6) Specification";
    }

    typedef port-number {
```

```
      type uint16 {
        range "0..65535";
      }
      description
       "The port-number type represents a 16-bit port number of an
        Internet transport-layer protocol such as UDP, TCP, DCCP, or
        SCTP.  Port numbers are assigned by IANA.  A current list of
        all assignments is available from <http://www.iana.org/>.

        Note that the port number value zero is reserved by IANA.  In
        situations where the value zero does not make sense, it can
        be excluded by subtyping the port-number type.

        In the value set and its semantics, this type is equivalent
        to the InetPortNumber textual convention of the SMIv2.";
      reference
       "RFC  768: User Datagram Protocol
        RFC  793: Transmission Control Protocol
        RFC 4960: Stream Control Transmission Protocol
        RFC 4340: Datagram Congestion Control Protocol (DCCP)
        RFC 4001: Textual Conventions for Internet Network Addresses";
    }

    /*** collection of types related to autonomous systems ***/

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

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

        In the value set and its semantics, this type is equivalent
        to the InetAutonomousSystemNumber textual convention of
        the SMIv2.";
      reference
       "RFC 1930: Guidelines for creation, selection, and registration
```

```
                of an Autonomous System (AS)
        RFC 4271: A Border Gateway Protocol 4 (BGP-4)
        RFC 4001: Textual Conventions for Internet Network Addresses
        RFC 6793: BGP Support for Four-Octet Autonomous System (AS)
                  Number Space";
    }

    /*** collection of types related to IP addresses and hostnames ***/

    typedef ip-address {
      type union {
        type inet:ipv4-address;
        type inet:ipv6-address;
      }
      description
       "The ip-address type represents an IP address and is IP
        version neutral.  The format of the textual representation
        implies the IP version.  This type supports scoped addresses
        by allowing zone identifiers in the address format.";
      reference
       "RFC 4007: IPv6 Scoped Address Architecture";
    }

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

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

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

    typedef ipv6-address {
      type string {
        pattern '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
```

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

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

        The canonical format of IPv6 addresses uses the textual
        representation defined in Section 4 of RFC 5952.  The
        canonical format for the zone index is the numerical
        format as described in Section 11.2 of RFC 4007.";
      reference
       "RFC 4291: IP Version 6 Addressing Architecture
        RFC 4007: IPv6 Scoped Address Architecture
        RFC 5952: A Recommendation for IPv6 Address Text
                  Representation";
    }

    typedef ip-address-no-zone {
      type union {
        type inet:ipv4-address-no-zone;
        type inet:ipv6-address-no-zone;
      }
      description
       "The ip-address-no-zone type represents an IP address and is
        IP version neutral.  The format of the textual representation
        implies the IP version.  This type does not support scoped
        addresses since it does not allow zone identifiers in the
        address format.";
      reference
       "RFC 4007: IPv6 Scoped Address Architecture";
    }

    typedef ipv4-address-no-zone {
      type inet:ipv4-address {
        pattern '[0-9\.]*';
```

```
      }
      description
        "An IPv4 address without a zone index.  This type, derived from
         ipv4-address, may be used in situations where the zone is known
         from the context and hence no zone index is needed.";
    }

    typedef ipv6-address-no-zone {
      type inet:ipv6-address {
        pattern '[0-9a-fA-F:\.]*';
      }
      description
        "An IPv6 address without a zone index.  This type, derived from
         ipv6-address, may be used in situations where the zone is known
         from the context and hence no zone index is needed.";
      reference
       "RFC 4291: IP Version 6 Addressing Architecture
        RFC 4007: IPv6 Scoped Address Architecture
        RFC 5952: A Recommendation for IPv6 Address Text
                  Representation";
    }

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

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

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

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

        The definition of ipv4-prefix does not require that bits,
        which are not part of the prefix, are set to zero. However,
        implementations have to return values in canonical format,
        which requires non-prefix bits to be set to zero. This means
        that 192.0.2.1/24 must be accepted as a valid value but it
        will be converted into the canonical format 192.0.2.0/24.";
    }

    typedef ipv6-prefix {
      type string {
        pattern '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
              + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
              + '(((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}'
              + '(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])))'
              + '(/(([0-9])|([0-9]{2})|(1[0-1][0-9])|(12[0-8])))';
        pattern '(([^:]+:){6}(([^:]+:[^:]+)|(.*\..*))|'
              + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?)'
              + '(/.+)';
      }
      description
       "The ipv6-prefix type represents an IPv6 prefix.
        The prefix length is given by the number following the
        slash character and must be less than or equal to 128.

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

        The canonical format of an IPv6 prefix has all bits of
        the IPv6 address set to zero that are not part of the
        IPv6 prefix.  Furthermore, the IPv6 address is represented
        as defined in Section 4 of RFC 5952.

        The definition of ipv6-prefix does not require that bits,
        which are not part of the prefix, are set to zero. However,
        implementations have to return values in canonical format,
        which requires non-prefix bits to be set to zero. This means
        that 2001:db8::1/64 must be accepted as a valid value but it
        will be converted into the canonical format 2001:db8::/64.";
      reference
       "RFC 5952: A Recommendation for IPv6 Address Text
                  Representation";
    }
```

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

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

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

    typedef ipv6-address-and-prefix {
      type string {
        pattern '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
              + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
              + '(((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}'
              + '(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])))'
              + '(/(([0-9])|([0-9]{2})|(1[0-1][0-9])|(12[0-8])))';
        pattern '(([^:]+:){6}(([^:]+:[^:]+)|(.*\..*))|'
              + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?)'
              + '(/.+)';
      }
      description
       "The ipv6-address-and-prefix type represents an IPv6
        address and an associated ipv4 prefix.
        The prefix length is given by the number following the
        slash character and must be less than or equal to 128.

        A prefix length value of n corresponds to an IP address
        mask that has n contiguous 1-bits from the most
```

```
        significant bit (MSB) and all other bits set to 0.

        The canonical format requires that the IPv6 address is
        represented as defined in Section 4 of RFC 5952.";
      reference
       "RFC 5952: A Recommendation for IPv6 Address Text
                  Representation";
    }

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

    typedef domain-name {
      type string {
        length "1..253";
        pattern
          '((([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9]\.)*'
        + '([a-zA-Z0-9_]([a-zA-Z0-9\-_]){0,61})?[a-zA-Z0-9]\.?)'
        + '|\.';
      }
      description
       "The domain-name type represents a DNS domain name.  The
        name SHOULD be fully qualified whenever possible. This
        type does not support wildcards (see RFC 4592) or
        classless in-addr.arpa delegations (see RFC 2317).

        Internet domain names are only loosely specified.  Section
        3.5 of RFC 1034 recommends a syntax (modified in Section
        2.1 of RFC 1123).  The pattern above is intended to allow
        for current practice in domain name use, and some possible
        future expansion.  Note that Internet host names have a
        stricter syntax (described in RFC 952) than the DNS
        recommendations in RFCs 1034 and 1123, and that systems
        that want to store host names in schema node instances
        using the domain-name type are recommended to adhere to
        this stricter standard to ensure interoperability.

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

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

```
        explicitly or may depend on the configuration of the
        resolver.

        Domain-name values use the US-ASCII encoding.  Their canonical
        format uses lowercase US-ASCII characters.  Internationalized
        domain names MUST be A-labels as per RFC 5890.";
      reference
       "RFC  952: DoD Internet Host Table Specification
        RFC 1034: Domain Names - Concepts and Facilities
        RFC 1123: Requirements for Internet Hosts -- Application
                  and Support
        RFC 2317: Classless IN-ADDR.ARPA delegation
        RFC 2782: A DNS RR for specifying the location of services
                  (DNS SRV)
        RFC 4592: The Role of Wildcards in the Domain Name System
        RFC 5890: Internationalized Domain Names in Applications
                  (IDNA): Definitions and Document Framework";
    }

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

    /*
     * DISCUSS:
     * - Lada suggested to replace the inet:domain-name usage in
     *   the union with a new host-name definition that follows
     *   the NR-LDH definition in RFC 5890.
     */

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

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

```
        Section 6.2.2.1.

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

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

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

        In the value set and its semantics, this type is equivalent
        to the Uri SMIv2 textual convention defined in RFC 5017.";
      reference
       "RFC 3986: Uniform Resource Identifier (URI): Generic Syntax
        RFC 3305: Report from the Joint W3C/IETF URI Planning Interest
                  Group: Uniform Resource Identifiers (URIs), URLs,
                  and Uniform Resource Names (URNs): Clarifications
                  and Recommendations
        RFC 5017: MIB Textual Conventions for Uniform Resource
                  Identifiers (URIs)";
    }

    typedef email-address {
      type string {
        // dot-atom-text "@" ...
        pattern '[a-zA-Z0-9!#$%&'+"'"+'*+/=?^_`{|}~-]+'
              + '(\.[a-zA-Z0-9!#$%&'+"'"+'*+/=?^_`{|}~-]+)*'
              + '@'
              + '[a-zA-Z0-9!#$%&'+"'"+'*+/=?^_`{|}~-]+'
              + '(\.[a-zA-Z0-9!#$%&'+"'"+'*+/=?^_`{|}~-]+)*';
      }
      description
        "The email-address type represents an email address as
         defined as addr-spec in RFC 5322 section 3.4.1.";
      reference
        "RFC 5322: Internet Message Format";
    }

    /*
     * DISCUSS:
     * - It was suggested to add email types following RFC 5322
     *   email-address        (addr-spec, per Section 3.4.1)
     *   named-email-address  (name-addr, per Section 3.4)
```

```
     * - This sounds useful but the devil is in the details,
     *   in particular name-addr is a quite complex construct;
     *   perhaps addr-spec is sufficient, this is also the
     *   format allowed in mailto: URIs (mailto: seems to use
     *   only a subset of addr-spec which may be good enough
     *   here as well).
     * - Need to define a pattern that has a meaningful trade-off
     *   between precision and complexity (there are very tight
     *   pattern that are very long and complex). The current
     *   pattern does not take care of quoted-string, obs-local-part,
     *   domain-literal, obs-domain.
     */

    /*
     * DISCUSS:
     * - There was a request to add types for URI fields (scheme,
     *   authority, path, query, fragment) but it is not clear how
     *   commonly useful these types are, the WG was pretty silent
     *   about this proposal. On the technical side, it is unclear
     *   whether data is represented with percent escapes resolved
     *   or not. (Mahesh's proposal does not spell this out, the
     *   pattern does not allow the % character, which may be wrong.)
     */
  }

   <CODE ENDS>
```

5. IANA Considerations

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

```
URI: urn:ietf:params:xml:ns:yang:ietf-yang-types
Registrant Contact: The NETMOD WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-inet-types
Registrant Contact: The NETMOD WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

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

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

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

6. Security Considerations

This document defines common data types using the YANG data modeling language. The definitions themselves have no security impact on the Internet, but the usage of these definitions in concrete YANG modules might have. The security considerations spelled out in the YANG specification [RFC7950] apply for this document as well.

7. Contributors

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

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

8. Acknowledgments

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

Juergen Schoenwaelder was partly funded by the European Union's Seventh Framework Programme under Grant Agreement ICT-318488 and the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 830927.

9. References

9.1. Normative References

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

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

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

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

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <https://www.rfc-editor.org/info/rfc4007>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[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>.

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

[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>.

[XPATH] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

9.2. Informative References

[IEEE802] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture", IEEE Std. 802-2001.

[ISO9834-1] ISO/IEC, "Information technology -- Open Systems Interconnection -- Procedures for the operation of OSI Registration Authorities: General procedures and top arcs of the ASN.1 Object Identifier tree", ISO/IEC 9834-1:2008, 2008.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, <https://www.rfc-editor.org/info/rfc952>.

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

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC1930] Hawkinson, J. and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)",

BCP 6, RFC 1930, DOI 10.17487/RFC1930, March 1996, <https://www.rfc-editor.org/info/rfc1930>.

[RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, DOI 10.17487/RFC2317, March 1998, <https://www.rfc-editor.org/info/rfc2317>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, DOI 10.17487/RFC2578, April 1999, <https://www.rfc-editor.org/info/rfc2578>.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, DOI 10.17487/RFC2579, April 1999, <https://www.rfc-editor.org/info/rfc2579>.

[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/info/rfc2780>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC2856] Bierman, A., McCloghrie, K., and R. Presuhn, "Textual Conventions for Additional High Capacity Data Types", RFC 2856, DOI 10.17487/RFC2856, June 2000, <https://www.rfc-editor.org/info/rfc2856>.

[RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information Base for the Differentiated Services Architecture", RFC 3289, DOI 10.17487/RFC3289, May 2002, <https://www.rfc-editor.org/info/rfc3289>.

[RFC3305] Mealling, M., Ed. and R. Denenberg, Ed., "Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations", RFC 3305, DOI 10.17487/RFC3305, August 2002, <https://www.rfc-editor.org/info/rfc3305>.

[RFC3595] Wijnen, B., "Textual Conventions for IPv6 Flow Label", RFC 3595, DOI 10.17487/RFC3595, September 2003, <https://www.rfc-editor.org/info/rfc3595>.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, DOI 10.17487/RFC4001, February 2005, <https://www.rfc-editor.org/info/rfc4001>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4502] Waldbusser, S., "Remote Network Monitoring Management Information Base Version 2", RFC 4502, DOI 10.17487/RFC4502, May 2006, <https://www.rfc-editor.org/info/rfc4502>.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <https://www.rfc-editor.org/info/rfc4592>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5017] McWalter, D., Ed., "MIB Textual Conventions for Uniform Resource Identifiers (URIs)", RFC 5017, DOI 10.17487/RFC5017, September 2007, <https://www.rfc-editor.org/info/rfc5017>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

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

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, DOI 10.17487/RFC5952, August 2010, <https://www.rfc-editor.org/info/rfc5952>.

[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>.

[RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <https://www.rfc-editor.org/info/rfc6793>.

[XSD-TYPES] Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>.

Appendix A. Changes from RFC 6991

This version adds new type definitions to the YANG modules. The following new data types have been added to the ietf-yang-types module:

- date, time
- hours32, minutes32, seconds32, centiseconds32, milliseconds32,
- microseconds32, microseconds64, nanoseconds32, nanoseconds64
- revision-identifier, node-instance-identifier

The following new data types have been added to the ietf-inet-types module:

- ip-address-and-prefix, ipv4-address-and-prefix, ipv6-address-and-prefix
- email-address

This version addresses errata 4076 and 5105 of RFC 6991.

Appendix B. Changes from RFC 6021

This version adds new type definitions to the YANG modules. The following new data types have been added to the ietf-yang-types module:

- yang-identifier
- hex-string
- uuid
- dotted-quad

The following new data types have been added to the ietf-inet-types module:

- ip-address-no-zone
- ipv4-address-no-zone
- ipv6-address-no-zone

Author's Address

Juergen Schoenwaelder (editor)
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 14, 2021

R. Wilton, Ed.
D. Ball
T. Singh
Cisco Systems
S. Sivaraj
Juniper Networks

July 13, 2020

Sub-interface VLAN YANG Data Models
draft-ietf-netmod-sub-intf-vlan-model-07

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 orginating 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 January 14, 2021.

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 [RFC2460], 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 RFC 2119 [RFC2119] RFC 8174 [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 seperate 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 [IEEE802.1Qcp-2018].

<CODE BEGINS> file "ietf-if-vlan-encapsulation@2020-07-13.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;
    reference
      "IEEE Std 802.1Qcp-2018: IEEE Standard for Local and
       metropolitan area networks -- Bridges and Bridged Networks --
       Amendment 30: YANG Data Model";
  }

  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) 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 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.

        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 2020-07-13 {
      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 [IEEE802.1Qcp-2018].

```
<CODE BEGINS> file "ietf-if-flexible-encapsulation@2020-07-13.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;
    reference
      "IEEE Std 802.1Qcp-2018: IEEE Standard for Local and
       metropolitan area networks -- Bridges and Bridged Networks --
       Amendment 30: YANG Data Model";
  }

  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) 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 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.

     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 2020-07-13 {
    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 classifed 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>
        </vsi-root>
      </network-instance>
    </network-instances>
    <pseudowires
        xmlns="urn:ietf:params:xml:ns:yang:ietf-pseudowires">
      <pseudowire>
        <name>pw1</name>
```

```
        <configured-pw>
          <peer-ip>2001:db8::50></peer-ip>
          <pw-id>100</pw-id>
        </configured-pw>
      </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. ChangeLog

XXX, RFC Editor, please delete this change log before publication.

9.1. WG version -07 and -06

- Apply markups from WG last call.

9.2. WG version -05

- Incorporate feedback from IEEE 802.1 WG, John Messenger in particular.
- Adding must contraints to ensure outer tags are always matched to C-VLAN and S-VLAN tags.
- Fixed bug where second tag could be matched without outer tag, and where tags must not be specified.

9.3. WG version -04

- Added examples

9.4. WG version -03

- Fix namespace bug in XPath identity references, removed extraneous 'dot1q-tag' containers.

9.5. WG version -02

- Use explicit containers for outer and inner tags rather than lists.

9.6. WG version -01

- Tweaked the abstract.
- Removed unnecessary feature for the L3 sub-interface module.
- Update the 802.1Qcp type references.
- Remove extra tag container for L3 sub-interfaces YANG.

9.7. Version -04

- IEEE 802.1 specific types have been removed from the draft. These are now referenced from the 802.1Qcp draft YANG modules.
- Fixed errors in the xpath expressions.

9.8. Version -03

- Incorporates feedback received from presenting to the IEEE 802.1 WG.
- Updates the modules for double tag matches/rewrites to restrict the outer tag type to S-VLAN and inner tag type to C-VLAN.
- Updates the introduction to indicate primary use case is for IETF forwarding protocols.
- Updates the objectives to make IEEE 802.1Q bridge interoperability a key objective.

10. IANA Considerations

10.1. YANG Module Registrations

The following YANG modules are requested to be registred 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.

11. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol RFC 6241 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH RFC 6242 [RFC6242]. The NETCONF access control model RFC 6536 [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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

11.1. 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

11.2. 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

12. References

12.1. Normative References

[I-D.ietf-netmod-intf-ext-yang]
Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaraj, "Common Interface Extension YANG Data Models", draft-ietf-netmod-intf-ext-yang-08 (work in progress), November 2019.

[IEEE802.1Qcp-2018]
Holness, M., "IEEE Std 802.1Qcp-2018: IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks -- Amendment 30: YANG Data Model", 2018.

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

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

[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>.

12.2. Informative References

[I-D.ietf-bess-l2vpn-yang] Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-10 (work in progress), July 2019.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <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>.

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

[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>.

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

David Ball
Cisco Systems

Email: daviball@cisco.com

Tapraj Singh
Cisco Systems

Email: tapsingh@cisco.com

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net

Network Working Group
Internet-Draft
Updates: 7950,8407,8525 (if approved)
Intended status: Standards Track
Expires: January 11, 2021

R. Wilton, Ed.
R. Rahman, Ed.
Cisco Systems, Inc.
B. Lengyel, Ed.
Ericsson
J. Clarke
Cisco Systems, Inc.
J. Sterne
Nokia
B. Claise
Cisco Systems, Inc.
K. D'Souza
AT&T

July 10, 2020

Updated YANG Module Revision Handling
draft-ietf-netmod-yang-module-versioning-01

Abstract

This document 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 an earliest revision filter to better represent inter-module dependencies. It provides help and 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. This document updates RFC 7950, 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 January 11, 2021.

Table of Contents

1. Introduction

This document defines a solution to the YANG module lifecycle problems described in [I-D.ietf-netmod-yang-versioning-reqs]. Complementary documents provide a complete solution to the YANG versioning requirements, with the overall relationship of the solution drafts described in [I-D.ietf-netmod-yang-solutions].

Specifically, this document recognises a need (within standards organizations, vendors, and the industry) to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which could cause breakage to clients and importing YANG modules. Accepting that non-backwards-compatible changes do sometimes occur, it is important to have mechanisms to report where these changes occur, and to manage their effect on clients and the broader YANG ecosystem.

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.

A YANG extension statement allowing YANG module imports to specify 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 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. Section 3 describes modifications to YANG revision handling and update rules, and Section 4 describes a YANG extension statement to do import by derived revision.

This document updates [RFC7950] section 5.6.5. 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 [RFC7950] section 5.2. Section 3.3 describes the use of a revision label in the name of a file containing a YANG module or submodule.

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.

In addition, this document uses the 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] assumes, but does not explicitly state, that the revision history for a YANG module is strictly linear, i.e., it is prohibited to have two independent revisions of a YANG module that are both directly derived from the same parent revision.

This document clarifies [RFC7950] to explicitly allow non linear development of YANG module revisions, so modules MAY have multiple revisions that directly derive from the same parent revision. As per [RFC7950], YANG module revisions continue to be uniquely identified by the module's revision date, and hence all revisions of a module MUST have unique revision dates.

A corollary to the above is that the relationship between two module revisions cannot be determined by comparing the module revision date alone, and the revision history, or revision label, must also be taken into consideration.

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 [[RFC7895] [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 requires that all updates to a YANG module are BC to the previous revision of the module. This document allows for more flexible evolution of YANG modules: NBC changes between module revisions are allowed and are documented using a new "nbc-changes" YANG extension statement in the module revision history.

Two revisions of a module MAY have identical content except for the revision history. This could occur, for example, if a module 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 to refine the rules for permissible changes when a new YANG module revision is created.

Where pragmatic, updates to YANG modules SHOULD be backwards-compatible, following the definition in Section 3.1.1.

A new module revision MAY contain NBC changes, i.e., the semantics of an existing definition MAY be changed in an NBC way without requiring a new definition with a new identifier. A new module revision with NBC changes MUST include the "rev:nbc-changes" extension substatement to signal the potential for incompatibility to existing module users and readers.

3.1.1. Backwards-compatible changes

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, 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 not a backwards-compatible change.

- Obsolete definitions MAY be removed from published modules, and are classified as backwards-compatible changes. In some circumstances it may be helpful to retain the obsolete definitions to ensure that their identifiers are not reused with a different meaning.

- In statements that have any data definition statements as substatements, those data definition substatements MAY be reordered, as long as they do not change the ordering of any "input" or "output" data definition substatements of "rpc" or "action" statements. If new data definition statements are added, they can be added anywhere in the sequence of existing substatements.

- 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. nbc-changes revision extension statement

The "rev:nbc-changes" 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:nbc-changes" extension statement MUST be added as a substatement to the "revision" statement.

Conversely, if a revision does not contain an "rev:nbc-changes" extension substatement then all changes, relative to the preceding revision in the revision history, MUST be backwards-compatible.

3.3. Revision label

This section updates [RFC7950] section 5.2, it explains how a revision label can be used in the name of a file containing a YANG module or submodule.

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.

YANG Semver [I-D.ietf-netmod-yang-semver] defines a versioning scheme based on Semver 2.0.0 [semver] that can be used as a revision label.

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 an "rev:revision-or-derived" extension statement argument.

If a revision has an associated revision label, then it may be used instead of the revision date in the filename of a YANG module, where it takes the form:

```
module-or-submodule-name [['@' revision-date]|['#' revision-label]]
    ( '.yang' / '.yin' )

  E.g., acme-router-modules@2018-01-25.yang
  E.g., acme-router-modules#2.0.3.yang
```

Two file names, one with the revision date and another with the revision label, MAY exist for the same module (or submodule) revision.

A specific revision-label identifies a specific revision (variant) 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 contents of the two modules MUST be identical.

3.3.1. Revision label scheme extension statement

The "rev:revision-label-scheme" extension statement is used to indicate which revision-label scheme a module or submodule uses. The mandatory argument to this extension statement:

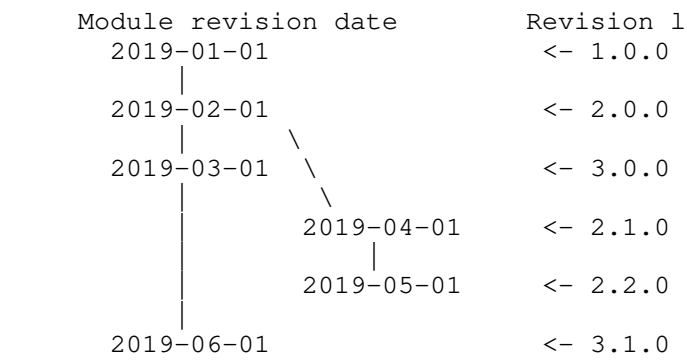
- Specifies the revision-label scheme used by the module or submodule
- Is defined in the document which specifies the revision-label scheme
- MUST be an identity derived from "revision-label-scheme-base"

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.

3.4. Examples for updating the YANG module revision history

The following diagram, explanation, and module history illustrates how the branched revision history, "nbc-changes" 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 oldest revision to newest:

Example module, revision 2019-06-01:

```
    module example-module {

      namespace "urn:example:module";
      prefix "prefix-name";

      import ietf-yang-revisions { prefix "rev"; }

      description
        "to be completed";

      revision 2019-06-01 {
        rev:revision-label 3.1.0;
        description "Add new functionality.";
      }

      revision 2019-04-01 {
        rev:revision-label 3.0.0;
        rev:nbc-changes;
        description
          "Add new functionality. Remove some deprecated nodes.";
      }

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:nbc-changes;
        description "Apply bugfix to pattern statement";
      }

      revision 2019-01-01 {
        rev:revision-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";

      import ietf-yang-revisions { prefix "rev"; }

      description
        "to be completed";

      revision 2019-05-01 {
        rev:revision-label 2.2.0;
        description "Backwards-compatible bugfix to enhancement.";
      }

      revision 2019-03-01 {
        rev:revision-label 2.1.0;
        description "Apply enhancement to older release train.";
      }

      revision 2019-02-01 {
        rev:revision-label 2.0.0;
        rev:nbc-changes;
        description "Apply bugfix to pattern statement";
      }

      revision 2019-01-01 {
        rev:revision-label 1.0.0;
        description "Initial revision";
      }

      //YANG module definition starts here
    }
```

4. Import by derived revision

RFC 7950 allows YANG module "import" statements to optionally require the imported module to have a particular revision date. In practice, importing a module with an exact revision date is often too restrictive because it requires the importing module to be updated whenever any change to the imported module occurs. The alternative choice of using an import statement without any revision date statement is also not ideal because the importing module may not work with all possible revisions of the imported module.

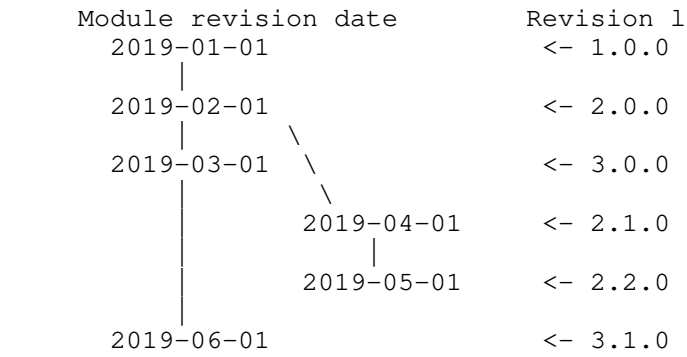
Instead, it is desirable for a importing module to specify a "minimum required revision" of a module that it is compatible with, based on the assumption that later revisions derived from that "minimum required revision" are also likely to be compatible. Many possible changes to a YANG module do not break importing modules, even if the changes themselves are not strictly backwards-compatible. E.g., fixing an incorrect pattern statement or description for a leaf would not break an import, changing the name of a leaf could break an import but frequently would not, but removing a container would break imports if that container is augmented by another module.

The ietf-revisions module defines the "revision-or-derived" extension statement, a substatement to the YANG "import" statement, to allow for a "minimum required revision" to be specified during import:

> The argument to the "revision-or-derived" extension statement is a revision date or a revision label.
>
> A particular revision of an imported module satisfies an import's "revision-or-derived" extension statement if the imported module's revision history contains a revision statement with a matching revision date or revision label.
>
> An "import" statement MUST NOT contain both a "revision-or-derived" extension statement and a "revision-date" statement.
>
> The "revision-or-derived" extension statement MAY be specified multiple times, allowing the import to use any module revision that satisfies at least one of the "revision-or-derived" extension statements.
>
> The "revision-or-derived" extension statement does not guarantee that all module revisions that satisfy an import statement are necessarily compatible, it only gives an indication that the revisions are more likely to be compatible. Hence, NBC changes to an imported module may also require new revisions of any importing modules, updated to accommodation those changes, along with updated import "revision-or-derived" extension statements to depend on the updated imported module revision.

4.1. Module import examples

Consider the example module "example-module" from Section 3.4 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. Example 1

This example selects module revisions 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:revision-or-derived 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:revision-or-derived 2.0.0;
}
```

4.1.2. Example 2

This example selects module revisions 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 valid revision for import.

```
import example-module {
  rev:revision-or-derived 2.1.0;
}
```

4.1.3. Example 3

This example selects revisions 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:revision-or-derived 2019-04-01;
  rev:revision-or-derived 2019-06-01;
}
```

5. Updates to ietf-yang-library

This document updates YANG library [RFC7950] to clarify how ambiguous module imports are resolved. It also defines the YANG module, ietf-yang-library-revisions that augments YANG library [RFC8525] with new revision-label related meta-data.

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] 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: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" list in ietf-yang-library with a "revision-label" leaf to optionally declare the revision label associated wth the particular revision of each module.

5.2.2. Reporting how deprecated and obsolete nodes are handled

The ietf-yang-library-revisions YANG module augments YANG library with two leaves to allow a server to report how it handles status "deprecated" and status "obsolete" nodes. The leaves are:

deprecated-nodes-implemented: If set to "true", this leaf indicates that all schema nodes with a status "deprecated" child statement are implemented equivalently as if they had status "current", or 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" 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" leaves to "true".

If a server does not set the "deprecated-nodes-implemented" leaf to "true", then clients MUST NOT rely solely on the "rev:nbc-changes" 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 [I-D.ietf-netmod-yang-instance-file-format] 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 are RECOMMENDED to 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:nbc-changes" 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 depencencies 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 remove, or curtail, the older description statements associated with the previous revisions.

The "rev:revision-or-derived" extension should 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. The revision-label substatement used in the new module revision MUST indicate the nature of the change, i.e. NBC or BC, to the module's schema tree.

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 the different ways in which this can be done:

- NBC changes can be sometimes be done incrementally using the "deprecated" status to provide clients time to adapt to NBC changes.
- NBC changes are done at once, i.e. without using "status" statements. Depending on the change, this may have a big impact on clients.
- If the server can support multiple revisions of the YANG module or of YANG packages(as specified in [I-D.ietf-netmod-yang-packages]), and allows the client to select the revision (as per [I-D.ietf-netmod-yang-ver-selection]), then NBC changes MAY be done without using "status" statements. Clients would be required to select the revision which they support and the NBC change would have no impact on them

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" and then when support is removed its status MUST be changed to "obsolete". Instead of using the "obsolete" status, the data node MAY be removed from the model but this has the risk of breaking modules which import the modified module.

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.

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@2020-07-06.yang"
 module ietf-yang-revisions {
   yang-version 1.1;
   namespace "urn:ietf:params:xml:ns:yang:ietf-yang-revisions";
   prefix rev;

   import ietf-yang-types {
     prefix yang;
     reference
       "XXXX [ietf-netmod-rfc6991-bis]: Common YANG Data Types";
   }

   organization
     "IETF NETMOD (Network Modeling) Working Group";
   contact
     "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
      WG List:  <mailto:netmod@ietf.org>

      Author:   Benoit Claise
                <mailto:bclaise@cisco.com>

      Author:   Joe Clarke
                <mailto:jclarke@cisco.com>

      Author:   Reshad Rahman
                <mailto:rrahman@cisco.com>

      Author:   Robert Wilton
                <mailto:rwilton@cisco.com>

      Author:   Kevin D'Souza
                <mailto:kd6913@att.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) 2019 IETF Trust and the persons identified as
      authors of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
```

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

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

      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 2020-07-06 {
     description
       "Initial version.";
     reference
       "XXXX: Updated YANG Module Revision Handling";
   }

   typedef revision-label {
     type string {
       length "1..255";
       pattern '[^\s@]+';
       pattern '\d{4}-\d{2}-\d{2}' {
         modifier invert-match;
       }
     }
     description
       "A label associated with a YANG revision.

        Excludes spaces and '@'.  MUST NOT match revision-date.";
     reference
       "XXXX: Updated YANG Module Revision Handling;
        Section 3.3, Revision label";
   }

   typedef revision-date-or-label {
     type union {
       type yang:revision-identifier;
       type revision-label;
     }
     description
```

```
      "Represents either a YANG revision date or a revision label";
  }

  extension nbc-changes {
    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 'nbc-changes' statement per parent statement is
       allowed.
       The statement MUST NOT have any substatements.

       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 RFC-XXX, then
       the 'nbc-changes' statement MUST be added as a substatement to
       the revision statement.

       Conversely, if a revision of a YANG module only contains
       changes, relative to the preceding revision in the revision
       history, that are classified as 'backwards-compatible' then
       the revision statement MUST NOT contain any 'nbc-changes'
       substatement.";

    reference
      "XXXX: Updated YANG Module Revision Handling;
       Section 3.2, nbc-changes revision extension statement";
  }

  extension revision-label {
    argument revision-label;
    description
      "The revision label can be used to provide an additional
       versioning identifier associated with the revision.  E.g., one
       option for a versioning scheme that could be used is [TODO -
       Reference semver draft].

       The format of the revision-label argument MUST conform to the
       pattern defined for the revision-label typedef.

       The statement MUST only be a substatement of the revision
       statement.
       Zero or one revision-label statement per parent statement
       is allowed.
       The statement MUST NOT have any substatements.
```

```
        Revision labels MUST be unique amongst all revisions of a
        module.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3, Revision label";
    }

    extension revision-label-scheme {
      argument revision-label-scheme-identity;
      description
        "The revision label scheme specifies which revision-label scheme
         the module or submodule uses.

         The mandatory revision-label-scheme-identity 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'.

         This extension MUST be used if there is a revision-label
         statement in the module or submodule.

         The statement MUST NOT have any substatements.";

      reference
        "XXXX: Updated YANG Module Revision Handling;
         Section 3.3.1, Revision label scheme extension statement";
    }

    extension revision-or-derived {
      argument revision-date-or-label;
      description
        "Restricts 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 'revision-or-derived' statement per parent
         statement is allowed.
         The statement MUST NOT have any substatements.

         If specified multiple
         times, then any module revision that satisfies at least one of
```

```
        the 'revision-or-derived' statements is an acceptable revision
        for import.

        An 'import' statement MUST NOT contain both a
        'revision-or-derived' extension statement and a
        'revision-date' statement.

        A particular revision of an imported module satisfies an
        import's 'revision-or-derived' extension statement if the
        imported module's revision history contains a revision
        statement with a matching revision date or revision label.

        The 'revision-or-derived' extension statement does not
        guarantee that all module revisions that satisfy an import
        statement are necessarily compatible, it only gives an
        indication that the revisions are more likely to be
        compatible.";

     reference
       "XXXX: Updated YANG Module Revision Handling;
        Section 4, Import by derived revision";
   }

   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@2020-07-06.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:   Benoit Claise
               <mailto:bclaise@cisco.com>

     Author:   Joe Clarke
               <mailto:jclarke@cisco.com>

     Author:   Reshad Rahman
               <mailto:rrahman@cisco.com>

     Author:   Robert Wilton
               <mailto:rwilton@cisco.com>

     Author:   Kevin D'Souza
               <mailto:kd6913@att.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) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

      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 revision-label version with 1.0.0 and
  // remove this note.
  revision 2020-07-06 {
    rev:revision-label 0.1.0;
    description
      "Initial revision";
    reference
      "XXXX: Updated YANG Module Revision Handling";
  }

  augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
    description
      "Augmentation modules with a revision label";
    leaf revision-label {
      type rev:revision-label;
      description
        "The revision label associated with this module revision.
         The label MUST match the rev: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: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' child statement are implemented
         equivalently as if they had status 'current', or 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.";

      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' nodes.  If this leaf is
         set to false or absent, 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. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following individuals are (or have been) members of the design team and have worked on the YANG versioning project:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne
- Joe Clarke
- Juergen Schoenwaelder

- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton
- Bo Wu

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update].

Discussons on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

We would also like to thank Martin Bjorklund, Jan Lindblad and Italo Busi for their contributions.

10. Security Considerations

The document does not define any new protocol or data model. There are no security considerations beyond those specified in [RFC7950].

11. IANA Considerations

11.1. YANG Module Registrations

The following YANG module is requested to be registred in the "IANA Module Names" registry:

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]

11.2. Instructions

We may need to give new instructions to IANA e.g. how to review revision-label statements to make sure they are accurate? TBD

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc6991-bis]
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-rfc6991-bis-04 (work in progress), July 2020.

[I-D.ietf-netmod-yang-semver]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning", draft-ietf-netmod-yang-semver-00 (work in progress), March 2020.

[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>.

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

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

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

[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>.

[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>.

12.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-instance-file-format]
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-12 (work in progress), April 2020.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", draft-ietf-netmod-yang-packages-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-solutions]
Wilton, R., "YANG Versioning Solution Overview", draft-ietf-netmod-yang-solutions-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-ver-selection]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Schema Selection", draft-ietf-netmod-yang-ver-selection-00 (work in progress), March 2020.

[I-D.ietf-netmod-yang-versioning-reqs]
Clarke, J., "YANG Module Versioning Requirements", draft-ietf-netmod-yang-versioning-reqs-03 (work in progress), June 2020.

[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>.

[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 change or reduce 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 guidance for how some of these NBC changes could be made to a YANG module. 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 node's status is changed to "deprecated" and it is supported for at least one year. This is a BC change.
2. When the node is not available anymore, its status is changed to "obsolete" and the "description" updated, this is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. The new revision of the YANG module has the node's status changed to "obsolete" and the "description" updated, this is an NBC change.
2. Clients which require the data node select the YANG package containing the schema version they use

B.2. Changing the type of a leaf node

Changing the type of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed to a string:

1. The status of node "vpn-id" is changed to "deprecated" and the node should be available for at least one year. This is a BC change.

2. A new 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 where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server may prevent the new node from being set if the old node is already set (and vice-versa). The new node may have a when statement to achieve this. The old node must not have a when statement since this would be an NBC change, but the server could reject the old node from being set if the new node is already set.

    2. If the new node is set and a client does a get or get-config operation on the old node, the server could map the value. For example, if the new node "vpn-name" has value "123" then the server could return integer value 123 for the old node "vpn-id". However, if the value can not be mapped then the configuration would be incomplete, this is outside the scope of this document.

4. When node "vpn-id" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. In the new revision of the YANG module, the status of node "vpn-id" is changed to "obsolete". This is an NBC change.

2. New node "vpn-name" is added to the same location as described above.

3. Clients which require the data node select the YANG package containing the schema version they use

4. A server should not map between the nodes "vpn-id" and "vpn-name", i.e. if a client creates a data instance with "vpn-name" then that data instance should not be visible to a client using a module revision which has "vpn-id" (and vice-versa).

B.3. Reducing the range of a leaf node

Reducing the range of values of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed from type uint32 to type uint16:

1. If all values which are being removed were never supported, e.g. if a vpn-id of 65536 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 65536 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 should be available for at least one year. This is a BC change.

2. A new list is created in the same location with the same data but with "dest-address" as key. Finding an appropriate name for the new list can be tricky especially if the name of the existing list was perfect. 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 where both lists are available, how the server behaves when either list is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

1. A server could prevent the new list from being set if the old list already has entries (and vice-versa).

2. If the new list is set and a client does a get or get-config operation on the old list, the server could map the entries. However if the new list has entries which would lead to duplicate keys in the old list, the mapping can not be done.

4. When list "sessions" is not available anymore, its status is changed to "obsolete" and the "description" is updated. This is an NBC change.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. The new revision of the YANG module has the list "sessions" modified to have "dest-address" as key, this is an NBC change.

2. Clients which require the previous functionality select the older module revision

B.5. Renaming a node

A leaf-node or a container 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 the node should be available for at least one year. This is a BC change.

2. The new 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 where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:

    1. A server could 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 to achieve this. The old node must not have a when statement since this would be an NBC change, but

the server could reject the old node from being set if the new node is already set.

2. If the new node is set and a client does a get or get-config operation on the old node, the server could use the value of the new node. For example, if the new node "ip-address" has value X then the server may return value X for the old node "ip-adress".

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.

If the server can support NBC revisions of the YANG module simultaneously using version selection [I-D.ietf-netmod-yang-ver-selection], then the changes can be done immediately:

1. The new revision of the YANG module has the node with the new name replacing the node with the old name, this is an NBC change.

2. Clients which require the previous node name select the older module revision

B.6. Changing a default value

Appendix C. Changes between revisions

Note to RFC Editor (To be removed by RFC Editor)

v00 - v01

- Removed status-description
- Allowed both revision-date and revision-label in the filename.
- New extension revision-label-scheme
- To include submodules, inclusion by revision-date changed from MUST to SHOULD
- Submodules can use revision label scheme and it can be same or different as the including module's scheme
- Addressed various comments provided at WG adoption on rev-00

Authors' Addresses

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Reshad Rahman (editor)
Cisco Systems, Inc.

Email: rrahman@cisco.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

Benoit Claise
Cisco Systems, Inc.

Email: bclaise@cisco.com

Kevin D'Souza
AT&T

Email: kd6913@att.com

Network Working Group
Internet-Draft
Updates: 8407 (if approved)
Intended status: Standards Track
Expires: January 14, 2021

B. Claise
J. Clarke, Ed.
R. Rahman
R. Wilton, Ed.
Cisco Systems, Inc.
B. Lengyel
Ericsson
J. Sterne
Nokia
K. D'Souza
AT&T

July 13, 2020

YANG Semantic Versioning
draft-ietf-netmod-yang-semver-01

Abstract

This document specifies a scheme and guidelines for applying a modified set of semantic versioning rules to revisions of YANG modules. Additionally, this document defines a revision-label for this modified semver 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 January 14, 2021.

Table of Contents

1. Introduction

[I-D.ietf-netmod-yang-module-versioning] puts forth a number of concepts relating to modified rules for updating modules, a means to signal when a new revision of a module has non-backwards-compatible (NBC) changes compared to its previous revision, and a versioning scheme that uses the revision history as a lineage for determining from where a specific revision of a YANG module is derived. Additionally, section 3.3 of [I-D.ietf-netmod-yang-module-versioning] defines a revision label which can be used as an overlay or alias to provide additional context or an additional way to refer to a specific revision.

This document defines a revision-label scheme that uses modified [semver] rules for YANG artifacts (i.e., YANG modules and YANG packages [I-D.ietf-netmod-yang-packages]) as well as the revision label definition for using this scheme. The goal of this is to add a human readable version label that provides compatibility information for the YANG artifact without one needing to compare or parse its body. The label and rules defined herein represent the RECOMMENDED revision label scheme for IETF YANG artifacts.

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 packages [I-D.ietf-netmod-yang-packages], and YANG schema elements are examples of YANG artifacts for the purposes of this document.

3. YANG Semantic Versioning

This section defines YANG Semantic Versioning, explains how it is used with YANG artifacts, and the rules associated with changing an artifact's semantic version number when its contents are updated.

3.1. YANG Semantic Versioning 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 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 present if the following COMPAT element is included

- COMPAT, if it is 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 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 by YANG semver parsers, pre-release metadata MUST be used during module development and MUST be considered base on Section 5. Both pre-release and build metadata are allowed in order to support all of 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 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 version string.

3.2. 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 lifecyles 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 limited 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 versioned using the YANG semantic versioning scheme specifies the module's semantic version number 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 may have been produced before this scheme was available.

YANG packages that make use of this semantic versioning scheme will have their semantic version as the value of the "revision_label" property.

As stated above, the YANG semver version number is expressed as a string of the form: 'X.Y.Z_COMPAT'; where X, Y, and Z each represent non-negative integers smaller than 2147483647 without leading zeroes, and _COMPAT represents an optional suffix of either "_compatible" or "_non_compatible".

- '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 patch "_compatible" or "_non_compatible" modifier.

- 'Z_COMPAT' is the PATCH version and modifier. Changes in the patch version number can indicate editorial, 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. 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. An example is correcting a spelling mistake in the description of a leaf within a YANG module. 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.

* 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

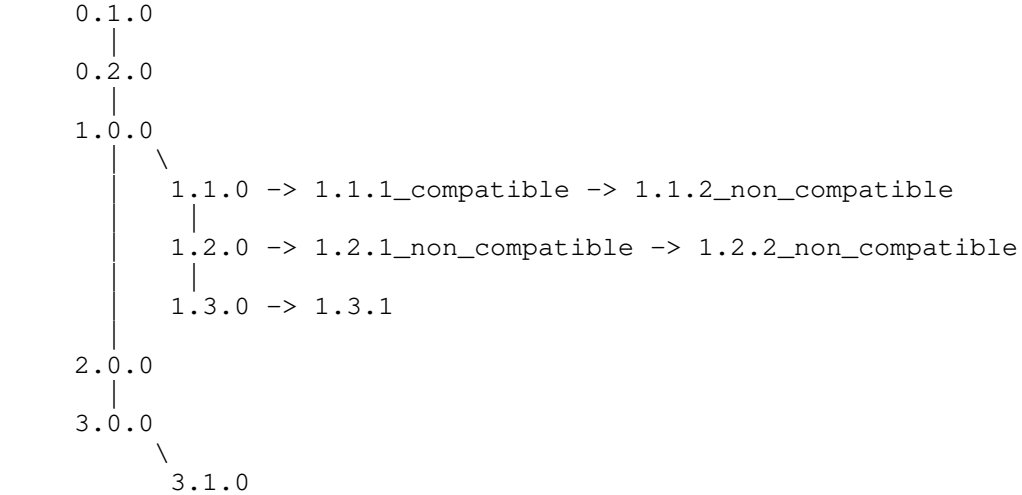
The YANG artifact name and YANG semantic version number uniquely identify a revision of said artifact. There MUST NOT be multiple instances of a YANG artifact definition with the same name and YANG semantic version number but different content (and in the case of modules, different revision dates).

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.

3.2.1. Examples for YANG semantic version numbers

The following diagram and explanation illustrates how YANG semantic version numbers work.

Example YANG semantic version numbers for an example artifact:



Assume the tree diagram above illustrates how an example YANG module's version history might evolve. For example, the tree might represent the following changes, listed in chronological order from oldest revision to newest:

0.1.0 - first beta module version

0.2.0 - second beta 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)

1.3.0 - improve existing functionality, added leaf "foo-64" (BC)

1.3.1 - improve description wording for "foo-64" (Editorial)

1.1.1_compatible - backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0 (BC)

2.0.0 - change existing model for performance reasons, e.g. re-key list (NBC)

1.1.2_non_compatible - NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.2.1_non_compatible - backport NBC fix, changing "baz" to "bar"

1.2.2_non_compatible - backport "wibble". This is a BC change but "non_compatible" modifier is sticky.

3.1.0 - introduce new leaf "wobble" (BC)

The partial ordering relationships based on the semantic versioning numbers can be defined as follows:

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 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

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, the module definition in 2.0.0 does not necessarily contain the contents of 1.3.0. However, the module revision history in 2.0.0 may well indicate that it was edited from module version 1.3.0.

3.3. YANG Semantic Version Update Rules

When a new revision of an artifact is produced, then the following rules define how the YANG semantic version number for the new artifact revision is calculated, based on the changes between the two artifact revisions, and the YANG semantic version number of the base artifact revision from which the changes are derived:

1. If an artifact is being updated in a non-backwards-compatible way, then the artifact version "X.Y.Z[_compatible|_non_compatible]" MUST be updated to "X+1.0.0" unless that artifact version has already been defined with different content, in which case the artifact version "X.Y.Z+1_non_compatible" MUST be used instead.

2. Under some circumstances (e.g., to avoid adding a "_compatible" modifier) an artifact author MAY also update the MAJOR version when the only changes are backwards-compatible. This is where tooling is important to highlight all changes. Because, while avoiding the "_compatible" and "_non_compatible" modifiers have a clear advantage, bumping a MAJOR version when changes are entirely backwards-compatible may confuse end users.

3. 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 MUST be updated to "X.Y+1.0", unless that artifact version has already been defined with different content, when the artifact version MUST be updated to "X.Y.Z+1_compatible"" instead.

   ii "X.Y.Z_compatible" - the artifact version MUST be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version MUST be updated to "X.Y.Z+1_non_compatible".

4. 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 MUST be updated to "X.Y.Z+1"

   ii "X.Y.Z_compatible" - the artifact version MUST be updated to "X.Y.Z+1_compatible".

   iii "X.Y.Z_non_compatible" - the artifact version MUST be updated to "X.Y.Z+1_non_compatible".

5. YANG artifact semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta 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 development.

3.4. Examples of the YANG Semver Label

3.4.1. 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 "yangver:yang-semver";

  import ietf-yang-revisions { prefix "rev"; }
  import ietf-yang-semver { prefix "yangver"; }

  description
    "to be completed";

  revision 2018-02-28 {
    description "Added leaf 'wobble'";
    rev:revision-label "3.1.0";
  }

  revision 2017-12-31 {
    description "Rename 'baz' to 'bar', added leaf 'wibble'";
    rev:revision-label "3.0.0";
    rev:nbc-changes;
```

```
        }

        revision 2017-10-30 {
          description "Change the module structure";
          rev:revision-label "2.0.0";
          rev:nbc-changes;
        }

        revision 2017-08-30 {
          description "Clarified description of 'foo-64' leaf";
          rev:revision-label "1.3.1";
        }

        revision 2017-07-30 {
          description "Added leaf foo-64";
          rev:revision-label "1.3.0";
        }

        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 2017-04-03 {
          description "First release version.";
          rev:revision-label "1.0.0";
        }

        // Note: semver rules do not apply to 0.X.Y labels.

        revision 2017-01-30 {
          description "NBC changes to initial revision";
          semver:module-version "0.2.0";
        }

        revision 2017-01-26 {
          description "Initial module version";
          semver:module-version "0.1.0";
        }

        //YANG module definition starts here
```

3.4.2. Example of Package Using YANG Semver

Below is an example YANG package that uses the semver revision label based on the rules defined in this document.

```
   {
     "ietf-yang-instance-data:instance-data-set": {
       "name": "example-yang-pkg",
       "target-ptr": "TBD",
       "timestamp": "2018-09-06T17:00:00Z",
       "description": "Example IETF package definition",
       "content-data": {
         "ietf-yang-package:yang-package": {
           "name": "example-yang-pkg",
           "version": "1.3.1",
           ...
}
```

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:revision-or-derived statement can specify either a revision date or a revision label. When importing by semver, the YANG semver revision label value MAY be used as an argument to rev:revision-or-derived. In so, any module which has that semver label as its latest revision label or has that label in its revision history can be used to satisfy the import requirement. For example:

```
        import example-module {
          rev:revision-or-derived "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.

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 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 to disregard

strict semver rules with respect to non-backwards-compatible changes during its initial development. However, module developers MAY choose to use the semver pre-release syntax instead with a 1 for the MAJOR version component. For example, an initial module 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 is nearing initial release (e.g., a module'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 '.' 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 using the YANG semver revision-label scheme, the intended target semver version MUST be used along with pre-release notation. For example, if a released module 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 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 undergoes development, it is possible that the original intent changes. For example, a 1.0.0 version of a module 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 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

[TODO: Describe precedence considering there could be changes during development and parallel development tracks.]

5.2. YANG Semver in IETF Modules

Net new module development within the IETF SHOULD begin with the 0 MAJOR number scheme as described above. When revising an existing IETF module, 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.

All IETF modules in development MUST use the whole document name as a pre-release version string, including the current document revision. For example, if a module 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. Even when using the 0 MAJOR version for initial module 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.

If a module is being revised and the original module never had a revision-label (i.e., you wish to start using YANG semver in future module revisions), choose a semver value that makes the most sense based on the module's history. For example, if a module 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).

See Appendix A for a detailed example of IETF pre-release versions.

6. YANG Module

This YANG module contains the typedef for the YANG semantic version.

<CODE BEGINS> file "ietf-yang-semver@2019-09-06.yang"

```
  module ietf-yang-semver {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-semver";
    prefix yangver;
    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>";
    description
      "This module provides type and grouping definitions for YANG
       packages.

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

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

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.

    revision 2020-06-30 {
      rev:revision-label "1.0.0-draft-ietf-netmod-yang-semver-01";
      description
        "Initial revision";
      reference
        "RFC XXXX: YANG Semantic Versioning.";
    }
```

```
    /*
     * Identities
     */

    identity yang-semver {
      base rev:revision-label-scheme-base-identity;
      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 string {
        pattern '\d+[.]\d+[.]\d+(_(non_)?compatible)?(-[\w\d.]+)?([+][\w\d\.]+)?'
;
      }
      description
        "Represents a YANG semantic version number.  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

This document grew out of the YANG module versioning design team that started after IETF 101. The design team consists of the following members whom have worked on the YANG versioning project:

- Balazs Lengyel
- Benoit Claise
- Ebben Aries
- Jason Sterne

- Joe Clarke
- Juergen Schoenwaelder
- Mahesh Jethanandani
- Michael (Wangzitao)
- Qin Wu
- Reshad Rahman
- Rob Wilton

The initial revision of this document was refactored and built upon [I-D.clacla-netmod-yang-model-update].

Discussons 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.

8. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

9. IANA Considerations

None.

10. References

10.1. Normative References

[I-D.ietf-netmod-yang-module-versioning]
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "Updated YANG Module Revision Handling", draft-ietf-netmod-yang-module-versioning-00 (work in progress), March 2020.

[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>.

[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>.

10.2. Informative References

[I-D.clacla-netmod-yang-model-update]
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

[I-D.ietf-netmod-yang-packages]
Wilton, R., Rahman, R., Clarke, J., Sterne, J., and W. Bo, "YANG Packages", draft-ietf-netmod-yang-packages-00 (work in progress), March 2020.

[openconfigsemver]
"Semantic Versioning for Openconfig Models", <http://www.openconfig.net/docs/semver/>.

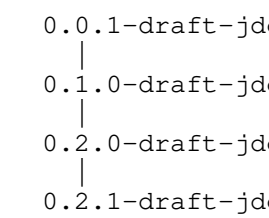
[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[semver] "Semantic Versioning 2.0.0", <https://www.semver.org>.

Appendix A. Example IETF Module Development

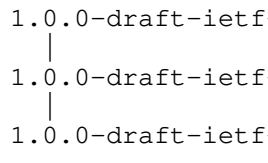
Assume a new YANG module is being developed in the netmod working group in the IETF. Initially, this module is being developed in an individual internet draft, draft-jdoe-netmod-example-module. The following represents the initial version tree (i.e., value of revision-label) of the module as it's being 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.

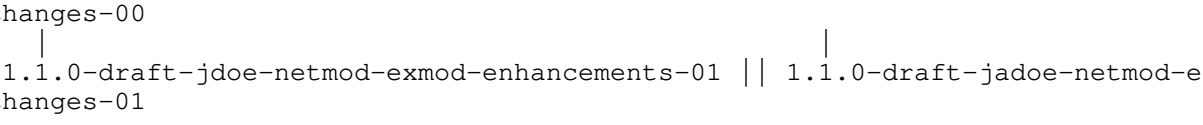
Continued version lineage after adoption:



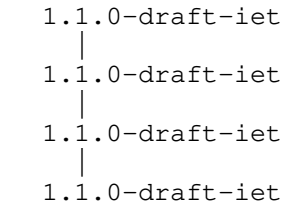
At this point, the draft is ratified and becomes RFC12345 and the YANG module version number becomes 1.0.0.

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-jadoe-netmod-exmod-changes. These are initially developed in parallel with the following versions.

Parallel development for next module revision:



At this point, the WG decides to merge some aspects of both and adopt the work in jadoe's draft as draft-ietf-netmod-exmod-changes. A single version lineage continues.



The draft is ratified, and the new module version becomes 1.1.0.

Authors' Addresses

Benoit Claise
Cisco Systems, Inc.
De Kleetlaan 6a b1
1831 Diegem
Belgium

Phone: +32 2 704 5622
Email: bclaise@cisco.com

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

Reshad Rahman
Cisco Systems, Inc.

Email: rrahman@cisco.com

Robert Wilton (editor)
Cisco Systems, Inc.

Email: rwilton@cisco.com

Balazs Lengyel
Ericsson
Magyar Tudosok Korutja
1117 Budapest
Hungary

Phone: +36-70-330-7909
Email: balazs.lengyel@ericsson.com

Jason Sterne
Nokia

Email: jason.sterne@nokia.com

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
United States of America

Email: kd6913@att.com

NETMOD Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2021

Q. Wu
Huawei
B. Claise
Cisco
L. Geng
Z. Du
China Mobile

July 8, 2020

Telemetry Data Self Explanation Tags
draft-tao-netmod-yang-node-tags-03

Abstract

This document defines a method to tag data node associated with telemetry data in YANG Modules. This YANG data node tagging method can be used to provide input, instruction, indication to selection filter and filter queries of operational state on a server during a "pub/sub" service for YANG datastore updates and provide multiple dimensional network visibility analysis when the state of all subscriptions of a particular Subscriber to be fetched is huge, so that the amount of data to be streamed out to the destination can be greatly reduced and only targeted to the characteristics data.

An extension statement to be used to indicate YANG data node self explanation tags that SHOULD be added by the module implementation automatically (i.e., outside of configuration).

A YANG module [RFC7950] is defined, which augments Module tag model and provides a list of data node entries to allow for adding or removing of data node self explanation tags as well as viewing the set of self explanation tags associated with a YANG module.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at 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 January 9, 2021.

Table of Contents

1. Introduction

As described [I.D-ietf-netmod-module-tags], the use of tags for classification and organization is fairly ubiquitous not only within IETF protocols, but in the internet itself (e.g., "#hashtags"). A module tag defined in [I.D-ietf-netmod-module-tags] is a string associated only with a module name at module level.

At the time of writing this document (2020), there are many data models that have been specified or are being specified by various different SDOs and Open Souce community. They cover many of the networking protocols and techniques. However data objects defined by these technology specific data models might represent a portion of fault, configuration, accounting, performance, security management categories information (e.g., performance metric specific data object type) in various different ways, lack consistent classification criteria and representation granularity,e.g., sensor data in hardware model is defined with fine granularity with value scale and value precision while interface model only provides statistics data for specific interface type.

This document defines data node self explanation tags and associates them with data nodes within YANG module, which

- Provide dictionary meaning for specific targeted data nodes;
- Indicate relationship between data nodes within the same YANG module or from different YANG modules;
- Identify key performance metric scale, precision, statistics operation;
- Identify specific service or feature, data source.

The data node self explanation tags can be used by the client to provide input, instruction, indication to selection filter and filter queries of configuration or operational state on a server based on these data node tags,.e.g.,return specific object type of operational state related to system-management. NETCONF clients can discover data objects with data node self explanation tags supported by a NETCONF server via <get-schema> operation. The data node self explanation tag capability can also be advertised via capability notification Model [I-D.netconf-notification-capabilities] by the NETCONF server or some place where offline document are kept. These

self explanation tags may be registered as well as assigned during the module definition; assigned by implementations; or dynamically defined and set by users.

This document defines a YANG module [RFC7950] which augments module tag model and provides a list of data node entries to allow for adding or removing of self explanation tags as well as viewing the set of self explanation tags associated with a data node within YANG modules.

This document defines an extension statement to be used to indicate self explanation tags that SHOULD be added by the module implementation automatically (i.e., outside of configuration).

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

1.1. Data Node tags Use Cases

The following is a list of already implemented and potential use cases.

1.1.1. Multiple Dimensional Performance Measurement Information Tagging

Data node tags can be used to express multiple dimensional performance metric and properties associated with YANG data nodes or data objects modelled with YANG (See Figure 1).

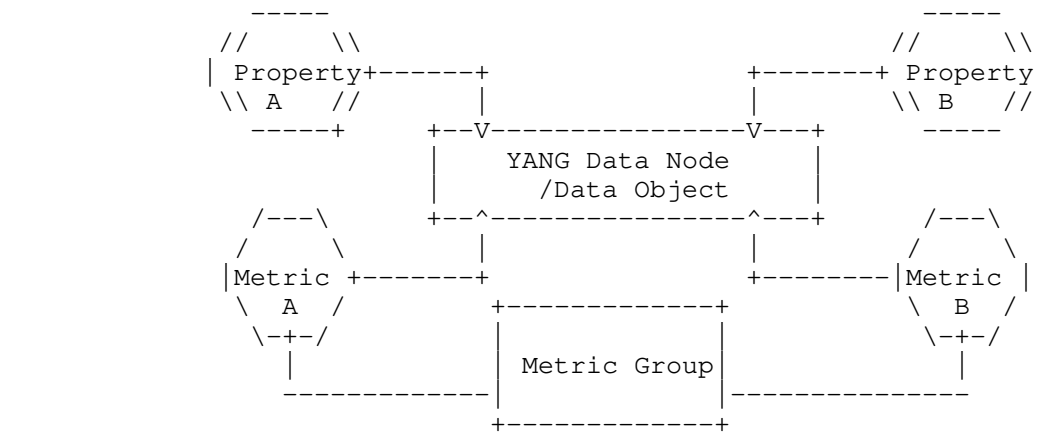


Figure 1

The use of data node tags would be to help filter different discrete categories of YANG data nodes across YANG modules supported by a

device. If data nodes across YANG modules are suitably tagged and learnt by the client from a live server, then an XPath query can be used by the client to list all related data nodes supported by a device with the same characteristics. Data node tags can also be used to help coordination when clients are interacting with various different devices with the same categories of YANG data node across different YANG modules. For example, one management client could mark some specific data node across modules implemented in various different devices with the same metric group tag, so consistent representation and reporting can be provided for YANG data nodes belonging to the same metric group (see Figure 2).

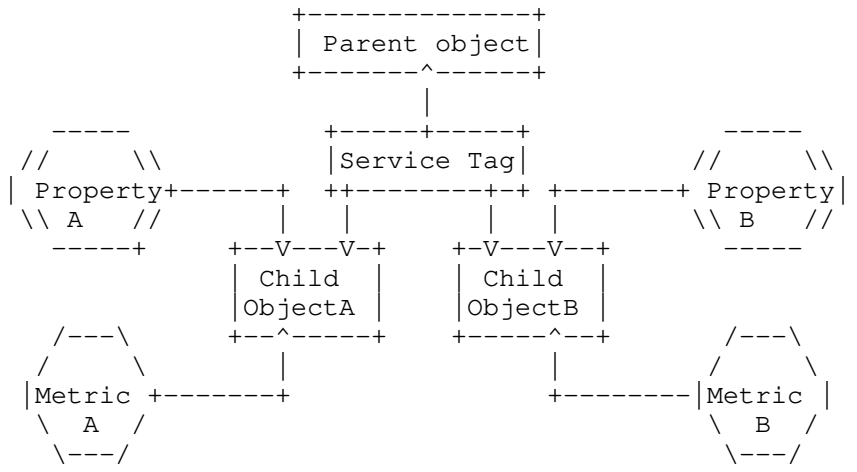
Object Name	Property Name	Metric Group	Metric Name	Module
tunnel-svc	name	-	-	tunnel
tunnel-svc	create-time	-	-	tunnel
tunnel-svc	modified-time	-	-	
tunnel-svc	-	lsp-ping-pm	avg-latency	tunnel-pm
tunnel-svc	-	lsp-ping-pm	packet-loss	tunnel-pm
tunnel-svc	-	lsp-ping-pm	min-latency	tunnel-pm
tunnel-svc	-	lsp-ping-pm	max-latency	tunnel-pm
tunnel-svc	-	lsp-ping-pm	transmitted -packet	tunnel-pm

Metric Group	Metric Name	Metric Precision	Metric Scale	Metric Unit
lsp-ping-pm	avg-latency	1	1	ms
lsp-ping-pm	packet-loss	1	1	percentile
lsp-ping-pm	min-latency	1	1	ms
lsp-ping-pm	max-latency	1	1	ms
lsp-ping-pm	transmitted	1	1	

Figure 2

1.1.2. Correlated Information Tagging

Another example is the management client could mark some data node across different level of YANG modules implemented in the device, the management system with the same service tag (e.g., L3VPN Service), so root cause can be identified efficiently during service-level agreements and performance monitoring or network failure troubleshooting.



Service Tag	Metric Group	Metric Name	Module	Level
L3VPN	L3VPN	maximum-routes	L3VPN	Service
L3VPN	OSPF-Process	total-active routes	OSPF	Device
L3VPN	RIP-Process	total-active routes	RIP	Device
L3VPN	BGP-Process	total-active routes	BGP	Device

1.2. 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.

2. Data Node Tag Values

All data node tags SHOULD begin with a prefix indicating who owns their definition. An IANA registry (Section 7.1) is used to support registering data node tag prefixes. Currently 3 prefixes are defined.

No further structure is imposed by this document on the value following the registered prefix, and the value can contain any YANG type 'string' characters except carriage-returns, newlines and tabs. Therefore, designers, implementers, and users are free to add or not add any structure they may require to their own tag values.

2.1. IETF Tags Prefix

An IETF tag is a data node tag that has the prefix "ietf:". All IETF data node tags are registered with IANA in a registry defined later in this document (Section 7.2).

2.2. Vendor Tags Prefix

A vendor tag is a tag that has the prefix "vendor:". These tags are defined by the vendor that implements the module, and are not registered; however, it is RECOMMENDED that the vendor include extra identification in the tag to avoid collisions such as using the enterpise or organization name following the "vendor:" prefix (e.g., vendor:vendor-defined-classifier).

2.3. User Tags Prefix

A user tag is any tag that has the prefix "user:". These tags are defined by the user/administrator and are not meant to be registered. Users are not required to use the "user:" prefix; however, doing so is RECOMMENDED as it helps avoid prefix collisions.

2.4. Reserved Tags Prefix

Any tag not starting with the prefix "ietf:", "vendor:" or "user:" is reserved for future use. These tag values are not invalid, but simply reserved in the context of specifications (e.g., RFCs).

3. Data Node Tag Management

Tags can become associated with a data node within YANG module in a number of ways. Tags may be defined and associated at module design time, at implementation time without the need of live server, or via user administrative control . As the main consumer of data node tags are users, users may also remove any tag from a live server, no matter how the tag became associated with a data node within a YANG module.

3.1. Module Design Tagging

A data node definition MAY indicate a set of data node tags to be added by the module implementer. These design time tags are indicated using the node-tag extension statement.

If the data node is defined in an IETF standards track document, the data node tags MUST be IETF Tags (2.1). Thus, new data node can drive the addition of new IETF tags to the IANA registry defined in Section 7.2, and the IANA registry can serve as a check against duplication.

3.2. Implementation Tagging

An implementation MAY include additional tags associated with data node within a YANG module. These tags SHOULD be IETF Tags (i.e., registered) or vendor specific tags.

3.3. User Tagging

Data node tags of any kind, with or without a prefix, can be assigned and removed by the user from a live server using normal configuration mechanisms. In order to remove a data node tag from the operational datastore, the user adds a matching "masked-tag" entry for a given data node within the ietf-data-node-tags Module.

4. Tags Module Structure

4.1. Tags Module Tree

The tree associated with the "ietf-data-node-tags" module follows. The meaning of the symbols can be found in [RFC8340].

```
        module: ietf-data-node-tags
        augment /tags:module-tags/tags:module:
          +--rw self-explanation-node-tags
             +--rw self-explanation-node* [node-name]
                +--rw node-name             nacm:node-instance-identifier
                +--rw opm-tag               tags:tag
                +--rw metric-precision      tags:tag
                +--rw metric-scale          tags:tag
                +--rw operation-type        tags:tag
                +--rw service-tag*          tags:tag
                +--rw task-tag*             tags:tag
                +--rw multi-source-tag      tags:tag
                +--rw data-source           tags:tag
```

5. YANG Module

```
<CODE BEGINS> file "ietf-self-explanation-node-tags@2019-05-03.yang"
    module ietf-self-explanation-node-tags {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-self-explanation-node-tags";
     prefix ntags;
     import ietf-netconf-acm  { prefix nacm; }
     import ietf-module-tags  { prefix tags; }
     organization
       "IETF NetMod Working Group (NetMod)";
     contact
       "WG Web:  <https://tools.ietf.org/wg/netmod/>
        WG List: <mailto:netmod@ietf.org>
        Editor:  Qin Wu <mailto:bill.wu@huawei.com>
        Editor:  Benoit Claise <mailto:bclaise@cisco.com>
        Editor:  Liang Geng <mailto:gengliang@chinamobile.com>
        Editor:  Zongpeng Du <mailto:duzongpeng@chinamobile.com>";
     // RFC Ed.: replace XXXX with actual RFC number and
     // remove this note.

     description
       "This module describes a mechanism associating self-explanation
        tags with YANG data node within YANG modules. Tags may be IANA
        assigned or privately defined.

        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 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://tools.ietf.org/html/rfcXXXX); see the RFC itself for
        full legal notices.";

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

     revision 2019-05-03 {
       description
         "Initial revision.";
       reference "RFC XXXX: YANG Data Node Tags";
     }
  typedef tag {
    type string {
      length "1..max";
      pattern '[a-zA-Z_][a-zA-Z0-9\-_]*:[\S ]+';
    }
    description
      "A tag value is composed of a standard prefix followed by any type
       'string' value that does not include carriage return, newline or
       tab characters.";
  }
    typedef metric-precision {
      type int8 {
        range "-8 .. 9";
      }
      description
        "A node using this data type represents a sensor value
         precision range.

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

         If a node of this type contains a value in the range 1 to 9,
         it represents the number of decimal places in the fractional
         part of an associated sensor-value fixed-point number.
         If a node of this type contains a value in the range -8 to -1,
         it represents the number of accurate digits in the associated
         sensor-value fixed-point number.

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

         Server implementers must choose a value for the associated
```

```
         sensor-value-precision node so that the precision and accuracy
         of the associated sensor-value node is correctly indicated.

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

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

```
          value 7;
          description
            "Measurement scaling factor of 10^-6.";
        }
        enum milli {
          value 8;
          description
            "Measurement scaling factor of 10^-3.";
        }
        enum units {
          value 9;
          description
            "Measurement scaling factor of 10^0.";
        }
        enum kilo {
          value 10;
          description
            "Measurement scaling factor of 10^3.";
        }
        enum mega {
          value 11;
          description
            "Measurement scaling factor of 10^6.";
        }
        enum giga {
          value 12;
          description
            "Measurement scaling factor of 10^9.";
        }
        enum tera {
          value 13;
          description
            "Measurement scaling factor of 10^12.";
        }
        enum peta {
          value 14;
          description
            "Measurement scaling factor of 10^15.";
        }
        enum exa {
          value 15;
          description
            "Measurement scaling factor of 10^18.";
        }
        enum zetta {
          value 16;
          description
            "Measurement scaling factor of 10^21.";
```

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

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

   identity metric-unit {
       description
        "Base identity for measurement unit.";
      }

     identity ac-volts {
      base metric-unit;
       description
        "Identity for a measure of electric potential (alternating current).";
     }
     identity dc-volts {
      base metric-unit;
       description
        "Identity for a measure of electric potential (direct current).";
     }
     identity amperes {
      base metric-unit;
       description
        "Identity for a measure of electric current.";
     }
     identity power {
      base metric-unit;
       description
        "Identity for a measure of power.";
     }
     identity hertz {
```

```
      base metric-unit;
       description
        "Identity for a measure of frequency.";
     }
     identity celsius {
      base metric-unit;
       description
        "Identity for a measure of temperature.";
     }
     identity rpm {
      base metric-unit;
       description
        "Identity for a measure of shaft revolutions per minute.";
     }
 extension opm-tag {
       argument tag;
       description
         "The argument 'tag' is of type 'tag'. This extension statement
          is used by module authors to indicate the opm tags that SHOULD be
          added automatically by the system. As such the origin of the
          value for the pre-defined tags should be set to 'system'
          [RFC8342].";
     }
 extension metric-scale{
       argument tag;
       description
         "The argument 'tag' is of type 'tag'.The metric-scale can be
         used to provide an additional metric scale (e.g., Measurement
         scaling factor of 10^0, 10^-3,10^3) information associated with
         the performance metric data node tag.";
     }
 extension metric-precision {
       argument tag;
       description
         "The argument 'tag' is of type 'tag'.The metric-precision can be
         used to provide an additional metric precision (e.g., the range -8 to -1
,
         0, the range 1 to 9) information associated with the performance metric
         data node tag.";
     }

 extension statistics-operation {
       argument tag;
       description
         "The argument 'tag' is of type 'tag'.The statistics-operation can be
         used to provide an additional statistics operation type(e.g., sum,
         min, max,last) information associated with the performance metric
         data node tag.";
     }
```

```
 extension service-tag {
       argument tag;
       description
         "The argument 'tag' is of type 'tag'.The service-tag can be
         used to provide a service classification information (e.g., tunnel,
         l3vpn,l2vpn) information associated with YANG data node.";
     }

 extension task-tag {
       argument tag;
       description
         "The argument 'tag' is of type 'tag'.The task-tag can be
         used to provide a task classification information (e.g., fault managemen
t,
         performance measurement) information associated with YANG data node.";
     }
 extension data-source {
       argument tag;
       description
         "The argument 'tag' is of type 'tag'.The data-source-type can be
         used to provide an additional data source type (e.g., connectivity,
         resource, hardware,qos,policy) information associated with
         the performance metric data node tag.";
     }
 extension multi-source-tag {
       argument tag;
       description
         "The argument 'tag' is of type 'tag'.The multi-source-tag can be
         used to provide an additional multiple source aggregation
         information associated with the performance metric data node
         or interface related data node.";
     }

  augment "/tags:module-tags/tags:module" {
    description
      "Augment the Tags module with data node tag attributes";
     container self-explanation-node-tags {
       description
         "Contains the list of data nodes and their associated tags";
       list self-explanation-node {
         key "node-name";
         description
           "A list of self-explanation nodes and their associated tags";
         leaf node-name {
           type nacm:node-instance-identifier;
           mandatory true;
           description
             "The YANG data node name.";
         }
```

```
          leaf opm-tag {
            type tags:tag;
            description
              "Tags associated with the data node within YANG module. See
               the IANA 'YANG Data Node Tag Prefixes' registry for reserved
               prefixes and the IANA'IETF YANG Data Node Tags' registry for
               IETF tags.

               The 'operational' state [RFC8342] view of this list is
               constructed using the following steps:

               1) System tags (i.e., tags of 'system' origin) are added.
               2) User configured tags (i.e., tags of 'intended' origin)
               are added.
               3) Any tag that is equal to a masked-tag is removed.";
          }
          leaf metric-precision {
            type tags:tag;
            description
              "The numeric expression precision of performance
               metric related data node.";
          }
         leaf metric-scale {
             type tags:tag;
             description
              "The measurement scale of performance
               metric related data node.";
          }
          leaf operation-type{
           type tags:tag;
           description
            "Statistics operation of performance metric related
            data node, e.g., average,min, max,sum, threshold.
            If the operation type is threshold type, the corresponding
            data object support threshold handling,e.g.,scan all interfaces
             for a certain type every 5 seconds and check the counters or
            status to cross threshold, return an array of interface entries
            that match the search.If the operation type is average,min,max, sum,
            it indicate the data object supports statistics operation, e.g.,
            scan all interfaces for a certain type every 5 seconds up to 60 second
s,
            only return min, average, max, sum value of specific data object rathe
r than
            the values that are current at the end of 60 seconds.";
          }
          leaf service-tag {
           type tags:tag;
           description
            "The node-service-tag can be used to provide a service
            classification information (e.g., tunnel, l3vpn,l2vpn)
```

```
           information associated with YANG data node.";
         }
         leaf task-tag {
          type tags:tag;
          description
           "The node-task-tag can be used to provide a task
           classification information (e.g., fault management,
           performance measurement) information associated with
           YANG data node.";
         }
        leaf multi-source-tag {
            type tags:tag;
            description
             "The mulitple source tag can be used to identify multiple source
              aggregation tye(e.g., line card,member link in an aggregated
              Ethernet interface) related to performance metric related
              data node or interface related to data node). Two source
              aggregation source types are supported, one is aggregation
              which groups data from two or multiple different data objects,
              the other is membership which identitfy each data object(e.g.,
              linecard, member link from multiple source aggregation.";
         }
        leaf data-source {
            type tags:tag;
            description
             "The data source type (e.g., connectivity,resource, hardware
             ,qos,policy) associated with the performance metric data node
             within YANG module.";
         }
       }
      }
     }
  }
<CODE ENDS>
```

6. Guidelines to Model Writers

This section updates [RFC8407].

6.1. Define Standard Tags

A module MAY indicate, using node-tag extension statements, a set of tags that are to be automatically associated with it (i.e., not added through configuration).

```
module example-module-A {
       //...
       import ietf-data-node-tags { prefix ntags; }
       container top {
        ntags:opm-tag "ietf:object-type";
        list X {
         leaf foo {
          ntags:opm-tag "ietf:property";
         }
        }
      container Y {
       ntags:opm-tag "ietf:metric";
      leaf bar {
       ntags:statistics-operation "ietf:avg";
       ntags:metric-scale "ietf:milli";
       }
     }
   }
 // ...
}
```

The module writer can use existing standard tags, or use new tags defined in the model definition, as appropriate. For IETF standardized modules new data node tags MUST be assigned in the IANA registry defined below, see Section Section 7.2.

7. IANA Considerations

7.1. YANG Data Node Tag Prefixes Registry

IANA is asked to create a new registry "YANG Data Node Tag Prefixes" grouped under a new "Protocol" category named "YANG Data Node Tag Prefixes".

This registry allocates tag prefixes. All YANG data node tags SHOULD begin with one of the prefixes in this registry.

Prefix entries in this registry should be short strings consisting of lowercase ASCII alpha-numeric characters and a final ":" character.

The allocation policy for this registry is Specification Required [RFC8126]. The Reference and Assignee values should be sufficient to identify and contact the organization that has been allocated the prefix.

The initial values for this registry are as follows.

Prefix	Description	Reference	Assignee
ietf:	IETF Tags allocated in the IANA IETF YANG Data Node Tags registry	[This document]	IETF
vendor:	Non-registered tags allocated by the module implementer.	[This document]	IETF
user:	Non-registered tags allocated by and for the user.	[This document]	IETF

Other standards organizations (SDOs) wishing to allocate their own set of tags should allocate a prefix from this registry.

7.2. IETF YANG Data Node Tags Registry

IANA is asked to create four new registries "IETF YANG Data Node Tags","IETF Metric Precision Tags","IETF Statistics Operation Tags","Node Service Tag" grouped under a new "Protocol" category "IETF YANG Data Node Tags". These four registries should be included below "YANG Data Node Tag Prefixes" when listed on the same page.

Four registries allocate tags that have the registered prefix "ietf:". New values should be well considered and not achievable through a combination of already existing IETF tags.

The allocation policy for these four registries is IETF Review [RFC8126].

The initial values for these eight registries are as follows.

Data Node Tag	Description	Reference
ietf:object-type	Relates to object type (e.g., interfaces).	[This document]
ietf:metric	Relates to performance metric info (e.g., ifstatistics).	[This document]
ietf:metric-group	Represent metric group (e.g., flow statistics).	[This document]
ietf:property	Represents a object	[This

	property (e.g.,ifindex).	document]

Metric Precision	Description	Reference
ietf:minus-eight	Relates to metric precision of performance metric	[This document]
ietf:minus-seven	Relates to metric precision of performance metric	[This document]
ietf:minus-sixe	Relates to metric precision of performance metric	[This document]
ietf:minus-five	Relates to metric precision of performance metric	[This document]
ietf:minus-four	Relates to metric precision of performance metric	[This document]
ietf:minus-three	Relates to metric precision of performance metric	[This document]
ietf:minus-two	Relates to metric precision of performance metric	[This document]
ietf:minus-one	Relates to metric precision of performance metric	[This document]
ietf:zero	Relates to metric precision of performance metric	[This document]
ietf:one	Relates to metric precision of performance metric	[This document]
ietf:two	Relates to metric precision of performance metric	[This document]
ietf:three	Relates to metric precision of performance metric	[This document]
ietf:four	Relates to metric precision of performance metric	[This document]
ietf:five	Relates to metric precision of performance metric	[This document]

ietf:six	Relates to metric precision of performance metric	[This document]
ietf:seven	Relates to metric precision of performance metric	[This document]
ietf:eight	Relates to metric precision of performance metric	[This document]
ietf:nine	Relates to metric precision of performance metric	[This document]

Metric scale	Description	Reference
ietf:yocto	Relates to metric scale of performance metric	[This document]
ietf:zepto	Relates to metric scale of performance metric	[This document]
ietf:atto	Relates to metric scale of performance metric	[This document]
ietf: femto	Relates to metric scale of performance metric	[This document]
ietf: pico	Relates to metric scale of performance metric	[This document]
ietf: nano	Relates to metric scale of performance metric	[This document]
ietf: micro	Relates to metric scale of performance metric	[This document]
ietf: milli	Relates to metric scale of performance metric	[This document]
ietf: units	Relates to metric scale of performance metric	[This document]
ietf: kilo	Relates to metric scale of performance metric	[This document]
ietf: mega	Relates to metric scale of performance metric	[This document]

ietf: giga	Relates to metric scale of performance metric	[This document]
ietf: tera	Relates to metric scale of performance metric	[This document]
ietf: peta	Relates to metric scale of performance metric	[This document]
ietf: exa	Relates to metric scale of performance metric	[This document]
ietf: zetta	Relates to metric scale of performance metric	[This document]
ietf: yotta	Relates to metric scale of performance metric	[This document]

Operation Type Tag	Description	Reference
ietf:avg	Relates to statistics operation(e.g.,average, min, max, sum,etc)	[This document]
ietf:sum	Relates to statistics operation(e.g.,average, min, max, sum,etc)	[This document]
ietf:min	Relates to statistics operation(e.g.,average, min, max, sum,etc)	[This document]
ietf:max	Relates to statistics operation(e.g.,average, min, max, sum,etc)	[This document]
ietf:threshold	Relates to statistics operation(e.g.,average, min, max, threshold,etc)	[This document]

Parent Tag	Description	Reference
ietf:member	Relates to multiple source aggregation type(e.g., lag, linecard, sub inf)	[This document]
ietf:agg	Relates to multiple source aggregation type(e.g.,agg)	[This document]

Data Source	Description	Reference
ietf:service-flow	Relates to data source type(e.g., microburst).	[This document]
ietf:topo	Relates to data source type(e.g., topology).	[This document]
ietf:resource	Relates to data source type info (e.g., interface,queue).	[This document]
ietf:policy	Relates to data source type info (e.g., acl, routing policy	[This document]
ietf:hardware	Relates to data source type (e.g.,optical module).	[This document]

Service Tag	Description	Reference
ietf:l3vpn	Relates to service offering(e.g.,l3vpn l2vpn,tunnel,etc)	[This document]
ietf:l2vpn	Relates to service offering(e.g.,l3vpn l2vpn,tunnel,etc)	[This document]
ietf:te-tunnel	Relates to service offering(e.g.,l3vpn l2vpn,tunnel,etc)	[This document]

Task Tag	Description	Reference
ietf:vpn-diag	Relates to vpn serivce diagonostic function	[This document]
ietf:vpn-fullfilment	Relates to vpn service fullfillment function	[This document]
ietf:vpn-assurance	Relates to vpn service assurance function	[This document]

7.3. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

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

7.4. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
name:
   ietf-self-explaination-node-tags
namespace:
   urn:ietf:params:xml:ns:yang:ietf-self-explaination-node-tags
prefix:
   ntags
reference:
   RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
   this note.)
```

8. Security Considerations

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

This document adds the ability to associate data node tag meta-data with YANG modules. This document does not define any actions based on these associations, and none are yet defined, and therefore it does not by itself introduce any new security considerations.

Users of the data node tag-meta data may define various actions to be taken based on the data node tag meta-data. These actions and their definitions are outside the scope of this document. Users will need to consider the security implications of any actions they choose to define.

9. Contributors

The authors would like to thank Ran Tao for his major contributions to the initial modeling and use cases.

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>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

10.2. Informative References

[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>.

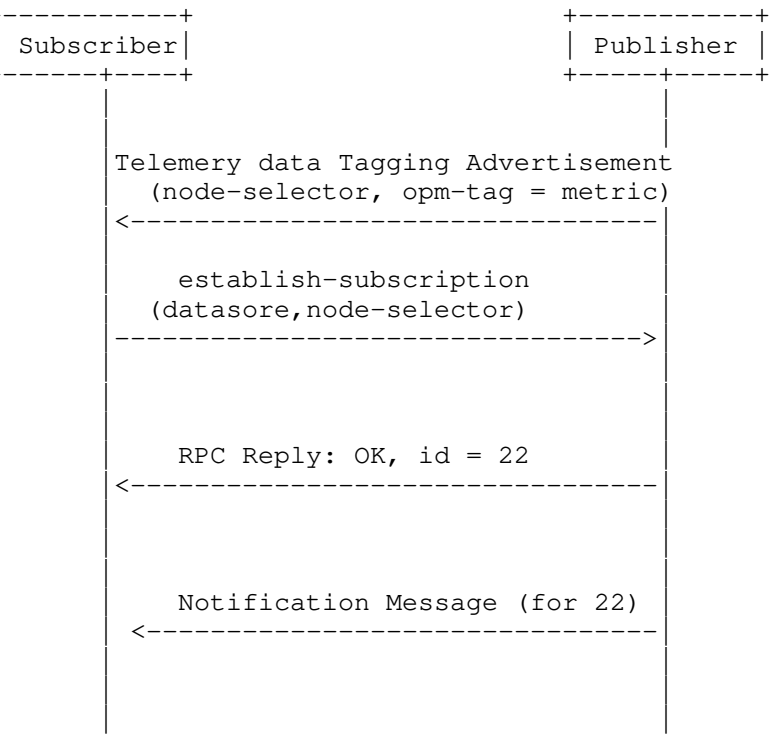
[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[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>.

[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>.

Appendix A. Targeted data object subscription example

The following subsections provides targeted data object subscription example.The subscription "id" values of 22 used below is just an example. In production, the actual values of "id" might not be small integers.



The publisher advertise telemetry data node capability to the subscriber to instruct the receiver to subscribe targeted data object

with specific characteristics (e.g., performance metric related data object) and specific data path corresponding to the targeted data object.

The following XML example [W3C.REC-xml-20081126] illustrates the advertisment of the list of available target objects:

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=\
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Defines the notification capabilities of an acme-router.
    The router only has running, and operational datastores.
    Every change can be reported on-change from running, but
    only config=true nodes and some config=false data from operational.
    Statistics are not reported based on timer based trigger and counter
    threshold based trigger.
  </description>
  <content-data>
    <system-capabilities \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities" \
      xmlns:inc=\
        "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities" \
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics/if:in-errors\
          </node-selector>
          <sec:self-describing-capabilities>
            <sec:self-tag-id>bandwith</sec:self-tag-id>
            <sec:opm-tag>metric</sec:opm-tag>
          </sec:self-describing-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

With telemetry data tagging information carried in the Telemetry data Tagging Advertisement, the subscriber identifies targeted data object

and associated data path to the datastore node and sends a establish-subscription RPC to subscribe specific data objects that are interests to the client application from the publisher.

```
<netconf:rpc message-id="101"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
      xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <yp:datastore
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
      /if:interfaces/if:interface/if:statistics/if:in-errors
    </yp:datastore-xpath-filter>
    <yp:periodic>
      <yp:period>500</yp:period>
    </yp:periodic>
  </establish-subscription>
</netconf:rpc>
```

The publisher returns specific object type of operational state related to the subscriber.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Benoit Claise
Cisco
De Kleetlaan 6a b1
Diegem 1831
Belgium

Email: bclaise@cisco.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

Zongpeng Du
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: duzongpeng@chinamobile.com