Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

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

July 2, 2018

Common Interface Extension YANG Data Models
draft-ietf-netmod-intf-ext-yang-06

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. These properties are common to many types of interfaces on network routers and switches and are implemented by multiple network equipment vendors with similar semantics, even though some of the features are not formally defined in any published standard.

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 draft is to provide a standard namespace and path for these configuration items regardless of the underlying interface type. For example a standard namespace and path 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 draft 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 internet draft are:

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

ietf-interfaces-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

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

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

2. Objectives

The aim of the YANG modules contained in this draft is to provide standard definitions for common interface based configuration on network devices.

The expectation is that the YANG leaves that are being defined are fairly widely implemented by network vendors. However, the features described here are mostly not backed by formal standards because they are fairly basic in their behavior and do not need to interoperate with other devices. Where required a concise explanation of the expected behavior is also provided to ensure consistency between vendors.

3. Interfaces Common Module

The Interfaces Common 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 parent interface leaf useable for all types of sub-interface that are children of parent interfaces.

The "ietf-interfaces-common" YANG module has the following structure:

```
module: ietf-interfaces-common
  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 l2-mtu?            uint16 {configurable-l2-mtu}?
    +--rw forwarding-mode?   identityref {forwarding-mode}?
  augment /if:interfaces/if:interface:
    +--rw parent-interface    if:interface-ref {sub-interfaces}?
```

3.1. 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 3.2 to provide effective control of unstable links and unwanted state transitions.

The principal 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-state/if:interface/oper-status or /if:interfaces-state/if:interfaces/last-change leaves for the interface that the feature is operating on. One obvious side effect of using this feature that is worth noting 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.

3.2. 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 nominally 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. Implementations are not required to use a penalty of 1000 units in their dampening algorithm, but should ensure that the Suppress Threshold and Reuse Threshold values are scaled relative to the nominal 1000 unit penalty to ensure that the same configuration

values provide consistent behaviour. The configurable values are described in more detail below.

3.2.1. 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 the default or configured suppress threshold, the interface is placed in a dampened state.

3.2.2. Half-Life Period

The half-life period determines how fast the accumulated penalties can decay exponentially. Any penalties that have been accumulated on a flapping interface are reduced by half after each half-life period.

3.2.3. 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 dampened state and allowed to go up.

3.2.4. Maximum Suppress Time

The maximum suppress time represents the maximum amount of time an interface can remain dampened when a 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.

3.3. 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].

3.4. 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.

3.5. Layer 2 MTU

A layer 2 MTU configuration leaf (l2-mtu) is provided to specify the maximum size of a layer 2 frame that may be transmitted or received on an interface. The layer 2 MTU includes the overhead of the layer 2 header and the maximum length of the payload, but excludes any frame check sequence (FCS) bytes. The payload MTU available to higher layer protocols is calculated from the l2-mtu leaf after taking the layer 2 header size into account.

For Ethernet interfaces carrying 802.1Q VLAN tagged frames, the l2-mtu excludes the 4-8 byte overhead of any known (e.g. explicitly matched by a child sub-interface) 801.1Q VLAN tags.

3.6. 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.

3.7. 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.

YANG Modules can conditionally use this leaf as a simple mechanism to determine whether particular types of configuration are valid. YANG modules can write 'must' statements to check whether the forwarding mode leaf has been configured, and if it is, then validate that the specified configuration is consistent with any forwarding mode that has also been configured. E.g., a layer 2 QoS policy YANG module could ensure that it is only applied to a interface forwarding traffic at layer 2 by checking whether the forwarding-mode leaf exists, and if it does then also ensure that it has been set to 'layer-2-forwarding'.

The following forwarding modes are defined:

- Optical Layer - Traffic is being forwarded at the optical layer. This includes DWDM or OTN based switching.

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

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

4. 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-interfaces-ethernet-like" YANG module has the following structure:

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

5. Interfaces Common YANG Module

This YANG module augments the interface container defined in RFC 8343 [RFC8343].

```
<CODE BEGINS> file "ietf-interfaces-common@2018-07-02.yang"
module ietf-interfaces-common {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces-common";

  prefix if-cmn;

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

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

   import iana-if-type {
     prefix ianaift;
   }

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

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

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

      WG Chair: Joel Jaeggli
                <mailto:joelja@gmail.com>

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

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

   description
     "This module contains common definitions for extending the IETF
      interface YANG model (RFC 7223) with common configurable layer 2
      properties.

      Copyright (c) 2016-2018 IETF Trust and the persons identified
      as authors of the code.  All rights reserved.

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

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

   revision 2018-07-02 {
     description
       "Initial version";
```

```
    reference "Internet draft: draft-ietf-netmod-intf-ext-yang-06";
  }

  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 XXX, Section 3.1 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 XXX, Section 3.2 Dampening";
  }

  feature loopback {
    description
      "This feature indicates that configurable interface loopback
       is supported.";
    reference "RFC XXX, Section 3.4 Loopback";
  }

  feature configurable-l2-mtu {
    description
      "This feature indicates that the device supports configuring
       layer 2 MTUs on interfaces.  Such MTU configurations include
       the layer 2 header overheads (but exclude any FCS overhead).
       The payload MTU available to higher layer protocols is either
       derived from the layer 2 MTU, taking into account the size of
       the layer 2 header, or is further restricted by explicit layer
       3 or protocol specific MTU configuration.";
    reference "RFC XXX, Section 3.5 Layer 2 MTU";
  }

  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 XXX, Section 3.6 Sub-interface";
```

```
   }

   feature forwarding-mode {
     description
       "This feature indicates that the device supports the
        configurable forwarding mode leaf";
     reference "RFC XXX, Section 3.7 Forwarding Mode";
   }

   /*
    * 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 XXX, Section 3.6 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 XXX, section 3.4";
   }
   identity loopback-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 XXX, section 3.4";
   }
   identity loopback-line {
     base loopback;
     description
       "All ingress traffic received on the interface is internally
        looped back within the interface to the egress path.";
```

```
    reference "RFC XXX, section 3.4";
  }
  identity loopback-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 XXX, section 3.4";
  }

  identity forwarding-mode {
    description "Base identity for forwarding-mode options.";
    reference "RFC XXX, section 3.7";
  }
  identity optical-layer {
    base forwarding-mode;
    description
      "Traffic is being forwarded at the optical layer.  This
       includes DWDM or OTN based switching.";
    reference "RFC XXX, section 3.7";
  }
  identity layer-2-forwarding {
    base forwarding-mode;
    description
      "Layer 2 based forwarding, such as Ethernet/VLAN based
       switching, or L2VPN services.";
    reference "RFC XXX, section 3.7";
  }
  identity network-layer {
    base forwarding-mode;
    description
      "Network layer based forwarding, such as IP, MPLS, or L3VPNs.";
    reference "RFC XXX, section 3.7";
  }


  /*
   * 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.";
          }
        }
        default "none";
        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 XXX, Section 3.1 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 XXX, Section 3.2 Dampening";
      leaf half-life {
        type uint32;
        units seconds;
        description
          "The Time (in seconds) after which a penalty reaches half
           its original value. Once the interface has been assigned
           a penalty, the penalty is decreased by half after the
           half-life period. For some devices, the allowed values may
           be restricted to particular multiples of seconds. The
```

```
           default value is vendor/device specific.";
        reference "RFC XXX, Section 3.3.2 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 nominally 1000 units.";
        reference "RFC XXX, Section 3.2.3 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 nominally 1000 units.";
        reference "RFC XXX, Section 3.2.1 Suppress Threshold";
      }

      leaf max-suppress-time {
        type uint32;
        units seconds;
        description
          "Maximum time (in seconds) that an interface can be
           suppressed. This value effectively acts as a ceiling that
           the penalty value cannot exceed.  The default value is
           vendor/device specific.";
        reference "RFC XXX, Section 3.2.4 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 XXX, Section 3.2 Dampening";
      }

      leaf suppressed {
        type boolean;
        default "false";
```

```
        config false;
        description
          "Represents whether the interface is suppressed (i.e. held
           down) because the 'penalty' leaf value exceeds the
           'suppress' leaf.";
        reference "RFC XXX, Section 3.2 Dampening";
      }

      leaf time-remaining {
        when '../suppressed = "true"' {
          description
            "Only suppressed interfaces should 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 XXX, Section 3.2 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 XXX, Section 3.3 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 XXX, Section 3.4 Loopback";
    }

    /*
     * Many types of interfaces support a configurable layer 2 MTU.
     */
    leaf l2-mtu {
      if-feature "configurable-l2-mtu";
      type uint16 {
        range "64 .. 65535";
      }
      description
        "The maximum size of layer 2 frames that may be transmitted
         or received on the interface (excluding any FCS overhead).
         In the case of Ethernet interfaces it also excludes the
         4-8 byte overhead of any known (i.e. explicitly matched by
         a child sub-interface) 801.1Q VLAN tags.";
      reference "RFC XXX, Section 3.5 Layer 2 MTU";
    }
```

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

       description
         "The forwarding mode that the interface is operating in.";
       reference "RFC XXX, Section 3.7 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
       "Add 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 XXX, Section 3.6 Sub-interface";
     }
   }
 }
```

<CODE ENDS>

6. 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, VLAN sub-interfaces, Switch Virtual interfaces, and Pseudo-Wire Head-End interfaces.

```
<CODE BEGINS> file "ietf-interfaces-ethernet-like@2017-07-03.yang"
module ietf-interfaces-ethernet-like {
  yang-version 1.1;

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

  prefix ethlike;

  import ietf-interfaces {
    prefix if;
  }

  import ietf-yang-types {
    prefix yang;
  }

  import iana-if-type {
    prefix ianaift;
  }

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

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

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

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

     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.

     Copyright (c) 2016 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

  revision 2017-07-03 {
    description "Updated to conform to NMDA architecture";

    reference
      "Internet draft: draft-ietf-netmod-intf-ext-yang-05";
  }

  /*
   * 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:l2vlan') 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 {
        type yang:mac-address;
        description
```

```
            "The MAC address of the interface.";
          }

          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.";
          }

          container statistics {
            config false;
            description
              "Packet statistics that apply to all Ethernet-like
               interfaces";
            leaf in-drop-unknown-dest-mac-pkts {
              type yang:counter64;
              units frames;
              description
                "A count of the number of frames that were well formed,
                 but otherwise dropped because the destination MAC
                 address did not pass any ingress destination MAC address
                 filter.

                 For consistency, frames counted against this drop
                 counters 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 counters in this
                 container 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>
```

7. 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].

7.1. 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-cmn="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <if-cmn:carrier-delay>
      <if-cmn:down>0</if-cmn:down>
      <if-cmn:up>1000</if-cmn:up>
    </if-cmn: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-cmn="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
    <interface>
      <name>eth0</name>
      <type>ianaift:ethernetCsmacd</type>
      <if-cmn:carrier-delay>
        <if-cmn:down>50</if-cmn:down>
        <if-cmn:up>1000</if-cmn:up>
      </if-cmn:carrier-delay>
    </interface>
  </interfaces>
</config>
```

7.2. 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>
    <dampening
     xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces-common">
      <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>
```

7.3. 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>
      <ethernet-like
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces-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 an explicit MAC address being configured on interface eth0.

```
<?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-interfaces-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>
    <ethernet-like
  xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces-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>
```

8. Acknowledgements

The authors wish to thank Eric Gray, Ing-Wher Chen, Juergen Schoenwaelder, Ladislav Lhotka, Mahesh Jethanandani, Michael Zitao, Neil Ketley, Qin Wu, William Lupton, Xufeng Liu, and Andy Bierman for their helpful comments contributing to this draft.

9. ChangeLog

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

9.1. Version -06

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

9.2. Version -05

- Incorporate feedback from Andy Bierman

9.3. Version -04

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

9.4. Version -03

- Fixed incorrect module name references, and updated tree output

9.5. Version -02

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

10. IANA Considerations

This document defines several new YANG module and the authors politely request that IANA assigns unique names to the YANG module files contained within this draft, and also appropriate URIs in the "IETF XML Registry".

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. interfaces-common.yang

The interfaces-common 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:

- /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/l2-mtu
- /if:interfaces/if:interface/forwarding-mode

Normally devices will not allow the parent-interface leaf to be changed after the interfce has been created. If an implementation

did allow the parent-interface leaf to be changed then it could cause all traffic on the affected interface to be dropped. The affected leaf is:

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

11.2. interfaces-ethernet-like.yang

Generally, the configuration nodes in the interfaces-ethernet-like YANG module are concerned with configuration that is common across all types of Ethernet-like interfaces. Currently, the module 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

12. References

12.1. 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>.

[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>.

12.2. 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-03 (work in progress), October 2017.

[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>.

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@juniper.net

Selvakumar Sivaraj
Juniper Networks

Email: ssivaraj@juniper.net