Internet Engineering Task Force (IETF)
Request for Comments: 8343
Obsoletes: 7223
Category: Standards Track
ISSN: 2070-1721

M. Bjorklund
Tail-f Systems

March 2018

A YANG Data Model for Interface Management

Abstract

This document defines a YANG data model for the management of network interfaces. It is expected that interface-type-specific data models augment the generic interfaces data model defined in this document. The data model includes definitions for configuration and system state (status information and counters for the collection of statistics).

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

This document obsoletes RFC 7223.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8343.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

This document defines a YANG data model [RFC7950] for the management of network interfaces. It is expected that interface-type-specific data models will augment the generic interfaces data model defined in this document.

Network interfaces are central to the management of many Internet protocols. Thus, it is important to establish a common data model for how interfaces are identified, configured, and monitored.

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

This version of the interfaces data model supports the Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Summary of Changes from RFC 7223

The "/interfaces-state" subtree with "config false" data nodes is deprecated. All "config false" data nodes are now present in the "/interfaces" subtree.

Servers that do not implement NMDA, or that wish to support clients that do not implement NMDA, MAY implement the deprecated "/interfaces-state" tree.

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.

The following terms are used within this document:

- system-controlled interface: An interface is said to be system-controlled if the system creates and deletes the interface independently of what has been explicitly configured. Examples are interfaces representing physical hardware that appear and disappear when hardware (e.g., a line card or hot-pluggable wireless interface) is added or removed. System-controlled interfaces may also appear if a certain functionality is enabled (e.g., a loopback interface might appear if the IP protocol stack is enabled).

- user-controlled interface: An interface is said to be user-controlled if the creation of the interface is controlled by adding explicit interface configuration to the intended configuration and the removal of the interface is controlled by removing explicit interface configuration from the intended configuration. Examples are VLAN interfaces configured on a system-controlled Ethernet interface.

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

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

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

- augment
- data model
- data node

1.3. Tree Diagrams

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

2. Objectives

This section describes some of the design objectives for the model presented in Section 5.

- It is recognized that existing implementations will have to map the interface data model defined in this memo to their proprietary native data model. To facilitate such mappings, the data model should be simple.
- The data model should be suitable for new implementations to use as is, without requiring a mapping to a different native model.
- References to interfaces should be as simple as possible, preferably by using a single leafref.
- The mapping to ifIndex [RFC2863] used by the Simple Network Management Protocol (SNMP) to identify interfaces must be clear.
- The model must support interface layering: both (1) simple layering, where one interface is layered on top of exactly one other interface, and (2) more complex scenarios, where one interface results from the aggregation of N other interfaces or when N interfaces are multiplexed over one other interface.
- The data model should support the pre-provisioning of interface configuration; that is, it should be possible to configure an interface whose physical interface hardware is not present on the device. It is recommended that devices that support dynamic addition and removal of physical interfaces also support pre-provisioning.
- The data model should support physical interfaces as well as logical interfaces.
- The data model should include read-only counters in order to gather statistics for sent and received octets and packets, received packets with errors, and packets that could not be sent due to errors.

3. Interfaces Data Model

This document defines the YANG module "ietf-interfaces", which has the following structure, excluding the deprecated "/interfaces-state" subtree:

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

3.1. The Interface List

The data model for interfaces presented in this document uses a flat list of interfaces ("/interfaces/interface"). Each interface in the list is identified by its name. Furthermore, each interface has a mandatory "type" leaf.

The "iana-if-type" module [RFC7224] defines YANG identities for the interface types in the IANA-maintained "ifType definitions" registry.

It is expected that interface-type-specific data models augment the interface list and possibly use the "type" leaf to make the augmentation conditional.

As an example of such an interface-type-specific augmentation, consider this YANG snippet. For a more complete example, see Appendix A.

```
  import interfaces {
      prefix "if";
  }
  import iana-if-type {
    prefix ianaift;
  }

  augment "/if:interfaces/if:interface" {
      when "if:type = 'ianaift:ethernetCsmacd'";

      container ethernet {
          leaf duplex {
              ...
          }
      }
  }
```

For system-controlled interfaces, the "name" is the device-specific name of the interface.

If the device supports arbitrarily named user-controlled interfaces, then the server will advertise the "arbitrary-names" feature. If the server does not advertise this feature, the names of user-controlled interfaces MUST match the device's naming scheme. How a client can learn the naming scheme of such devices is outside the scope of this document. See Appendices F.1 and F.2 for examples.

When a system-controlled interface is created in the operational state by the system, the system tries to apply the interface configuration in the intended configuration with the same name as the new interface. If no such interface configuration is found, or if the configured type does not match the real interface type, the system creates the interface without applying explicit configuration.

When a user-controlled interface is created, the configuration determines the name of the interface.

Depending on the operating system and the physical attachment point to which a network interface may be attached or removed, it may be impossible for an implementation to provide predictable and consistent names for system-controlled interfaces across insertion/removal cycles as well as in anticipation of initial insertion. The ability to provide configurations for such interfaces is therefore dependent on the implementation and cannot be assumed in all cases.

3.2. Interface References

An interface is identified by its name, which is unique within the server. This property is captured in the "interface-ref" typedef, which other YANG modules SHOULD use when they need to reference an interface.

3.3. Interface Layering

There is no generic mechanism for how an interface is configured to be layered on top of some other interface. It is expected that interface-type-specific models define their own data nodes for interface layering by using "interface-ref" types to reference lower layers.

Below is an example of a model with such nodes. For a more complete example, see Appendix B.

```
  import interfaces {
      prefix "if";
  }
  import iana-if-type {
    prefix ianaift;
  }

  augment "/if:interfaces/if:interface" {
      when "if:type = 'ianaift:ieee8023adLag'";

      leaf-list slave-if {
          type if:interface-ref;
          must "/if:interfaces/if:interface[if:name = current()]"
             + "/if:type = 'ianaift:ethernetCsmacd'" {
              description
                  "The type of a slave interface must be
                   'ethernetCsmacd'.";
          }
      }
      // other bonding config params, failover times, etc.
  }
```

While the interface layering is configured in interface-type-specific models, two generic state data leaf-lists, "higher-layer-if" and "lower-layer-if", represent a read-only view of the interface layering hierarchy.

4. Relationship to the IF-MIB

If the device implements the IF-MIB [RFC2863], each entry in the "/interfaces/interface" list in the operational state is typically mapped to one ifEntry. The "if-index" leaf MUST contain the value of the corresponding ifEntry's ifIndex.

In most cases, the "name" of an "/interfaces/interface" entry is mapped to ifName. The IF-MIB allows two different ifEntries to have the same ifName. Devices that support this feature and also support the data model defined in this document cannot have a 1-1 mapping between the "name" leaf and ifName.

The configured "description" of an "interface" has traditionally been mapped to ifAlias in some implementations. This document allows this mapping, but implementers should be aware of the differences in the value space and persistence for these objects. See the YANG module definition of the leaf "description" in Section 5 for details.

The IF-MIB also defines the writable object ifPromiscuousMode. Since this object typically is not implemented as a configuration object by SNMP agents, it is not mapped to the "ietf-interfaces" module.

The ifMtu object from the IF-MIB is not mapped to the "ietf-interfaces" module. It is expected that interface-type-specific YANG modules provide interface-type-specific MTU leafs by augmenting the "ietf-interfaces" model.

There are a number of counters in the IF-MIB that exist in two versions: one with 32 bits and one with 64 bits. The 64-bit versions were added to support high-speed interfaces with a data rate greater than 20,000,000 bits/second. Today's implementations generally support such high-speed interfaces; hence, only 64-bit counters are provided in this data model. Note that the server that implements this module and an SNMP agent may differ in the time granularity in which they provide access to the counters. For example, it is common that SNMP implementations cache counter values for some time.

The objects ifDescr and ifConnectorPresent from the IF-MIB are not mapped to the "ietf-interfaces" module.

The following table lists the YANG data nodes with corresponding objects in the IF-MIB.

YANG data node in /interfaces/interface	IF-MIB object
name	ifName
type	ifType
description	ifAlias
admin-status	ifAdminStatus
oper-status	ifOperStatus
last-change	ifLastChange
if-index	ifIndex
link-up-down-trap-enable	ifLinkUpDownTrapEnable
phys-address	ifPhysAddress
higher-layer-if and lower-layer-if	ifStackTable
speed	ifSpeed and ifHighSpeed
discontinuity-time	ifCounterDiscontinuityTime
in-octets	ifHCInOctets
in-unicast-pkts	ifHCInUcastPkts
in-broadcast-pkts	ifHCInBroadcastPkts
in-multicast-pkts	ifHCInMulticastPkts
in-discards	ifInDiscards
in-errors	ifInErrors
in-unknown-protos	ifInUnknownProtos
out-octets	ifHCOutOctets
out-unicast-pkts	ifHCOutUcastPkts
out-broadcast-pkts	ifHCOutBroadcastPkts
out-multicast-pkts	ifHCOutMulticastPkts
out-discards	ifOutDiscards
out-errors	ifOutErrors

YANG Data Nodes and Related IF-MIB Objects

5. Interfaces YANG Module

This YANG module imports typedefs from [RFC6991].

<CODE BEGINS> file "ietf-interfaces@2018-02-20.yang"

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

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

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

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

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

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

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

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

     This version of this YANG module is part of RFC 8343; see
     the RFC itself for full legal notices.";

  revision 2018-02-20 {
    description
      "Updated to support NMDA.";
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }

  revision 2014-05-08 {
    description
      "Initial revision.";
    reference
      "RFC 7223: A YANG Data Model for Interface Management";
  }

  /*
   * Typedefs
   */

  typedef interface-ref {
    type leafref {
      path "/if:interfaces/if:interface/if:name";
```

```
    }
    description
      "This type is used by data models that need to reference
       interfaces.";
  }

  /*
   * Identities
   */

  identity interface-type {
    description
      "Base identity from which specific interface types are
       derived.";
  }

  /*
   * Features
   */

  feature arbitrary-names {
    description
      "This feature indicates that the device allows user-controlled
       interfaces to be named arbitrarily.";
  }
  feature pre-provisioning {
    description
      "This feature indicates that the device supports
       pre-provisioning of interface configuration, i.e., it is
       possible to configure an interface whose physical interface
       hardware is not present on the device.";
  }
  feature if-mib {
    description
      "This feature indicates that the device implements
       the IF-MIB.";
    reference
      "RFC 2863: The Interfaces Group MIB";
  }

  /*
   * Data nodes
   */

  container interfaces {
    description
      "Interface parameters.";
```

```
      list interface {
        key "name";

        description
          "The list of interfaces on the device.

           The status of an interface is available in this list in the
           operational state.  If the configuration of a
           system-controlled interface cannot be used by the system
           (e.g., the interface hardware present does not match the
           interface type), then the configuration is not applied to
           the system-controlled interface shown in the operational
           state.  If the configuration of a user-controlled interface
           cannot be used by the system, the configured interface is
           not instantiated in the operational state.

           System-controlled interfaces created by the system are
           always present in this list in the operational state,
           whether or not they are configured.";

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

             A device MAY restrict the allowed values for this leaf,
             possibly depending on the type of the interface.
             For system-controlled interfaces, this leaf is the
             device-specific name of the interface.

             If a client tries to create configuration for a
             system-controlled interface that is not present in the
             operational state, the server MAY reject the request if
             the implementation does not support pre-provisioning of
             interfaces or if the name refers to an interface that can
             never exist in the system.  A Network Configuration
             Protocol (NETCONF) server MUST reply with an rpc-error
             with the error-tag 'invalid-value' in this case.

             If the device supports pre-provisioning of interface
             configuration, the 'pre-provisioning' feature is
             advertised.

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

```
            When a configured user-controlled interface is created by
            the system, it is instantiated with the same name in the
            operational state.

            A server implementation MAY map this leaf to the ifName
            MIB object.  Such an implementation needs to use some
            mechanism to handle the differences in size and characters
            allowed between this leaf and ifName.  The definition of
            such a mechanism is outside the scope of this document.";
          reference
            "RFC 2863: The Interfaces Group MIB - ifName";
        }

        leaf description {
          type string;
          description
            "A textual description of the interface.

            A server implementation MAY map this leaf to the ifAlias
            MIB object.  Such an implementation needs to use some
            mechanism to handle the differences in size and characters
            allowed between this leaf and ifAlias.  The definition of
            such a mechanism is outside the scope of this document.

            Since ifAlias is defined to be stored in non-volatile
            storage, the MIB implementation MUST map ifAlias to the
            value of 'description' in the persistently stored
            configuration.";
          reference
            "RFC 2863: The Interfaces Group MIB - ifAlias";
        }

        leaf type {
          type identityref {
            base interface-type;
          }
          mandatory true;
          description
            "The type of the interface.

            When an interface entry is created, a server MAY
            initialize the type leaf with a valid value, e.g., if it
            is possible to derive the type from the name of the
            interface.

            If a client tries to set the type of an interface to a
            value that can never be used by the system, e.g., if the
            type is not supported or if the type does not match the
```

```
           name of the interface, the server MUST reject the request.
           A NETCONF server MUST reply with an rpc-error with the
           error-tag 'invalid-value' in this case.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifType";
      }

      leaf enabled {
        type boolean;
        default "true";
        description
          "This leaf contains the configured, desired state of the
           interface.

           Systems that implement the IF-MIB use the value of this
           leaf in the intended configuration to set
           IF-MIB.ifAdminStatus to 'up' or 'down' after an ifEntry
           has been initialized, as described in RFC 2863.

           Changes in this leaf in the intended configuration are
           reflected in ifAdminStatus.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifAdminStatus";
      }

      leaf link-up-down-trap-enable {
        if-feature if-mib;
        type enumeration {
          enum enabled {
            value 1;
            description
              "The device will generate linkUp/linkDown SNMP
               notifications for this interface.";
          }
          enum disabled {
            value 2;
            description
              "The device will not generate linkUp/linkDown SNMP
               notifications for this interface.";
          }
        }
        description
          "Controls whether linkUp/linkDown SNMP notifications
           should be generated for this interface.
```

```
            If this node is not configured, the value 'enabled' is
            operationally used by the server for interfaces that do
            not operate on top of any other interface (i.e., there are
            no 'lower-layer-if' entries), and 'disabled' otherwise.";
          reference
            "RFC 2863: The Interfaces Group MIB -
                       ifLinkUpDownTrapEnable";
        }

        leaf admin-status {
          if-feature if-mib;
          type enumeration {
            enum up {
              value 1;
              description
                "Ready to pass packets.";
            }
            enum down {
              value 2;
              description
                "Not ready to pass packets and not in some test mode.";
            }
            enum testing {
              value 3;
              description
                "In some test mode.";
            }
          }
          config false;
          mandatory true;
          description
            "The desired state of the interface.

             This leaf has the same read semantics as ifAdminStatus.";
          reference
            "RFC 2863: The Interfaces Group MIB - ifAdminStatus";
        }

        leaf oper-status {
          type enumeration {
            enum up {
              value 1;
              description
                "Ready to pass packets.";
            }
            enum down {
              value 2;
```

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

           This leaf has the same semantics as ifOperStatus.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifOperStatus";
      }

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

```
          reference
            "RFC 2863: The Interfaces Group MIB - ifLastChange";
        }

        leaf if-index {
          if-feature if-mib;
          type int32 {
            range "1..2147483647";
          }
          config false;
          mandatory true;
          description
            "The ifIndex value for the ifEntry represented by this
             interface.";
          reference
            "RFC 2863: The Interfaces Group MIB - ifIndex";
        }

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

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

        leaf-list lower-layer-if {
          type interface-ref;
          config false;
```

```
      description
        "A list of references to interfaces layered underneath this
         interface.";
      reference
        "RFC 2863: The Interfaces Group MIB - ifStackTable";
    }

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

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

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

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

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

        leaf in-unicast-pkts {
          type yang:counter64;
          description
            "The number of packets, delivered by this sub-layer to a
             higher (sub-)layer, that were not addressed to a
             multicast or broadcast address at this sub-layer.

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

        leaf in-broadcast-pkts {
          type yang:counter64;
          description
            "The number of packets, delivered by this sub-layer to a
             higher (sub-)layer, that were addressed to a broadcast
             address at this sub-layer.

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

        leaf in-multicast-pkts {
          type yang:counter64;
          description
            "The number of packets, delivered by this sub-layer to a
             higher (sub-)layer, that were addressed to a multicast
             address at this sub-layer.  For a MAC-layer protocol,
             this includes both Group and Functional addresses.
```

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

        leaf in-discards {
          type yang:counter32;
          description
            "The number of inbound packets that were chosen to be
             discarded even though no errors had been detected to
             prevent their being deliverable to a higher-layer
             protocol.  One possible reason for discarding such a
             packet could be to free up buffer space.

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

        leaf in-errors {
          type yang:counter32;
          description
            "For packet-oriented interfaces, the number of inbound
             packets that contained errors preventing them from being
             deliverable to a higher-layer protocol.  For character-
             oriented or fixed-length interfaces, the number of
             inbound transmission units that contained errors
             preventing them from being deliverable to a higher-layer
             protocol.

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

        leaf in-unknown-protos {
          type yang:counter32;
```

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

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


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

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

      leaf out-unicast-pkts {
        type yang:counter64;
        description
          "The total number of packets that higher-level protocols
           requested be transmitted and that were not addressed
           to a multicast or broadcast address at this sub-layer,
           including those that were discarded or not sent.

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

```
      }

      leaf out-broadcast-pkts {
        type yang:counter64;
        description
          "The total number of packets that higher-level protocols
           requested be transmitted and that were addressed to a
           broadcast address at this sub-layer, including those
           that were discarded or not sent.

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

      leaf out-multicast-pkts {
        type yang:counter64;
        description
          "The total number of packets that higher-level protocols
           requested be transmitted and that were addressed to a
           multicast address at this sub-layer, including those
           that were discarded or not sent.  For a MAC-layer
           protocol, this includes both Group and Functional
           addresses.

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

      leaf out-discards {
        type yang:counter32;
        description
          "The number of outbound packets that were chosen to be
           discarded even though no errors had been detected to
           prevent their being transmitted.  One possible reason
           for discarding such a packet could be to free up buffer
           space.
```

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

          leaf out-errors {
            type yang:counter32;
            description
              "For packet-oriented interfaces, the number of outbound
               packets that could not be transmitted because of errors.
               For character-oriented or fixed-length interfaces, the
               number of outbound transmission units that could not be
               transmitted because of errors.

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

      }
    }

    /*
     * Legacy typedefs
     */

    typedef interface-state-ref {
      type leafref {
        path "/if:interfaces-state/if:interface/if:name";
      }
      status deprecated;
      description
        "This type is used by data models that need to reference
         the operationally present interfaces.";
    }

    /*
     * Legacy operational state data nodes
     */

    container interfaces-state {
```

```
      config false;
      status deprecated;
      description
        "Data nodes for the operational state of interfaces.";

      list interface {
        key "name";
        status deprecated;

        description
          "The list of interfaces on the device.

           System-controlled interfaces created by the system are
           always present in this list, whether or not they are
           configured.";

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

             A server implementation MAY map this leaf to the ifName
             MIB object.  Such an implementation needs to use some
             mechanism to handle the differences in size and characters
             allowed between this leaf and ifName.  The definition of
             such a mechanism is outside the scope of this document.";
          reference
            "RFC 2863: The Interfaces Group MIB - ifName";
        }

        leaf type {
          type identityref {
            base interface-type;
          }
          mandatory true;
          status deprecated;
          description
            "The type of the interface.";
          reference
            "RFC 2863: The Interfaces Group MIB - ifType";
        }

        leaf admin-status {
          if-feature if-mib;
          type enumeration {
            enum up {
              value 1;
```

```
            description
              "Ready to pass packets.";
          }
          enum down {
            value 2;
            description
              "Not ready to pass packets and not in some test mode.";
          }
          enum testing {
            value 3;
            description
              "In some test mode.";
          }
        }
        mandatory true;
        status deprecated;
        description
          "The desired state of the interface.

           This leaf has the same read semantics as ifAdminStatus.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifAdminStatus";
      }

      leaf oper-status {
        type enumeration {
          enum up {
            value 1;
            description
              "Ready to pass packets.";
          }
          enum down {
            value 2;
            description
              "The interface does not pass any packets.";
          }
          enum testing {
            value 3;
            description
              "In some test mode.  No operational packets can
               be passed.";
          }
          enum unknown {
            value 4;
            description
              "Status cannot be determined for some reason.";
          }
          enum dormant {
```

```
            value 5;
            description
              "Waiting for some external event.";
          }
          enum not-present {
            value 6;
            description
              "Some component (typically hardware) is missing.";
          }
          enum lower-layer-down {
            value 7;
            description
              "Down due to state of lower-layer interface(s).";
          }
        }
        mandatory true;
        status deprecated;
        description
          "The current operational state of the interface.

           This leaf has the same semantics as ifOperStatus.";
        reference
          "RFC 2863: The Interfaces Group MIB - ifOperStatus";
      }

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

      leaf if-index {
        if-feature if-mib;
        type int32 {
          range "1..2147483647";
        }
        mandatory true;
        status deprecated;
        description
          "The ifIndex value for the ifEntry represented by this
           interface.";
```

```
          reference
            "RFC 2863: The Interfaces Group MIB - ifIndex";
        }

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

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

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

        leaf speed {
          type yang:gauge64;
          units "bits/second";
          status deprecated;
          description
              "An estimate of the interface's current bandwidth in bits
               per second.  For interfaces that do not vary in
               bandwidth or for those where no accurate estimation can
```

```
            be made, this node should contain the nominal bandwidth.
            For interfaces that have no concept of bandwidth, this
            node is not present.";
          reference
            "RFC 2863: The Interfaces Group MIB -
                       ifSpeed, ifHighSpeed";
        }

        container statistics {
          status deprecated;
          description
            "A collection of interface-related statistics objects.";

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

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

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

          leaf in-unicast-pkts {
            type yang:counter64;
            status deprecated;
            description
              "The number of packets, delivered by this sub-layer to a
               higher (sub-)layer, that were not addressed to a
               multicast or broadcast address at this sub-layer.
```

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

        leaf in-broadcast-pkts {
          type yang:counter64;
          status deprecated;
          description
            "The number of packets, delivered by this sub-layer to a
             higher (sub-)layer, that were addressed to a broadcast
             address at this sub-layer.

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

        leaf in-multicast-pkts {
          type yang:counter64;
          status deprecated;
          description
            "The number of packets, delivered by this sub-layer to a
             higher (sub-)layer, that were addressed to a multicast
             address at this sub-layer.  For a MAC-layer protocol,
             this includes both Group and Functional addresses.

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

        leaf in-discards {
          type yang:counter32;
          status deprecated;
```

```
        description
          "The number of inbound packets that were chosen to be
           discarded even though no errors had been detected to
           prevent their being deliverable to a higher-layer
           protocol.  One possible reason for discarding such a
           packet could be to free up buffer space.

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

      leaf in-errors {
        type yang:counter32;
        status deprecated;
        description
          "For packet-oriented interfaces, the number of inbound
           packets that contained errors preventing them from being
           deliverable to a higher-layer protocol.  For character-
           oriented or fixed-length interfaces, the number of
           inbound transmission units that contained errors
           preventing them from being deliverable to a higher-layer
           protocol.

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

      leaf in-unknown-protos {
        type yang:counter32;
        status deprecated;
        description
          "For packet-oriented interfaces, the number of packets
           received via the interface that were discarded because
           of an unknown or unsupported protocol.  For
           character-oriented or fixed-length interfaces that
           support protocol multiplexing, the number of
           transmission units received via the interface that were
           discarded because of an unknown or unsupported protocol.
           For any interface that does not support protocol
           multiplexing, this counter is not present.
```

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


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

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

      leaf out-unicast-pkts {
        type yang:counter64;
        status deprecated;
        description
          "The total number of packets that higher-level protocols
           requested be transmitted and that were not addressed
           to a multicast or broadcast address at this sub-layer,
           including those that were discarded or not sent.

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

      leaf out-broadcast-pkts {
        type yang:counter64;
        status deprecated;
```

```
        description
          "The total number of packets that higher-level protocols
           requested be transmitted and that were addressed to a
           broadcast address at this sub-layer, including those
           that were discarded or not sent.

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

      leaf out-multicast-pkts {
        type yang:counter64;
        status deprecated;
        description
          "The total number of packets that higher-level protocols
           requested be transmitted and that were addressed to a
           multicast address at this sub-layer, including those
           that were discarded or not sent.  For a MAC-layer
           protocol, this includes both Group and Functional
           addresses.

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

      leaf out-discards {
        type yang:counter32;
        status deprecated;
        description
          "The number of outbound packets that were chosen to be
           discarded even though no errors had been detected to
           prevent their being transmitted.  One possible reason
           for discarding such a packet could be to free up buffer
           space.
```

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

           leaf out-errors {
             type yang:counter32;
             status deprecated;
             description
               "For packet-oriented interfaces, the number of outbound
                packets that could not be transmitted because of errors.
                For character-oriented or fixed-length interfaces, the
                number of outbound transmission units that could not be
                transmitted because of errors.

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

<CODE ENDS>

6. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

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

Registrant Contact: The IESG.

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

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

```
  name:         ietf-interfaces
  namespace:    urn:ietf:params:xml:ns:yang:ietf-interfaces
  prefix:       if
  reference:    RFC 8343
```

7. Security Considerations

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

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

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

/interfaces/interface: This list specifies the configured interfaces on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

/interfaces/interface/enabled: This leaf controls whether or not an interface is enabled. Unauthorized access to this leaf could cause the device to ignore packets it should receive and process.

8. References

8.1. 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>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <https://www.rfc-editor.org/info/rfc2863>.

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

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

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

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

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

[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>.

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

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[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>.

8.2. Informative References

[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>.

Appendix A. Example: Ethernet Interface Module

This section gives a simple example of how an Ethernet interface module could be defined. It demonstrates how media-specific configuration parameters can be conditionally augmented to the generic interface list. It also shows how operational state parameters can be conditionally augmented to the operational interface list. The example is not intended as a complete module for Ethernet configuration.

```
module example-ethernet {
  namespace "http://example.com/ethernet";
  prefix "eth";

  import ietf-interfaces {
    prefix if;
  }
  import iana-if-type {
    prefix ianaift;
  }

  // configuration and state parameters for Ethernet interfaces
  augment "/if:interfaces/if:interface" {
    when "if:type = 'ianaift:ethernetCsmacd'";

    container ethernet {
      container transmission {
        choice transmission-params {
          case auto {
            leaf auto-negotiate {
              type empty;
            }
          }
          case manual {
            container manual {
              leaf duplex {
                type enumeration {
                  enum "half";
                  enum "full";
                }
              }
              leaf speed {
                type enumeration {
                  enum "10Mb";
                  enum "100Mb";
                  enum "1Gb";
                  enum "10Gb";
                }
```

```
                }
              }
            }
          }
          leaf duplex {
            type enumeration {
              enum "half";
              enum "full";
            }
            config false;
          }
        }
        // other Ethernet-specific params...
      }
    }
  }
```

Appendix B. Example: Ethernet Bonding Interface Module

This section gives an example of how interface layering can be defined. An Ethernet bonding interface that bonds several Ethernet interfaces into one logical interface is defined.

```
module example-ethernet-bonding {
  namespace "http://example.com/ethernet-bonding";
  prefix "bond";

  import ietf-interfaces {
    prefix if;
  }
  import iana-if-type {
    prefix ianaift;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ianaift:ieee8023adLag'";

    leaf-list slave-if {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]"
         + "/if:type = 'ianaift:ethernetCsmacd'" {
        description
          "The type of a slave interface must be 'ethernetCsmacd'.";
      }
    }
    leaf bonding-mode {
      type enumeration {
        enum round-robin;
```

```
            enum active-backup;
            enum broadcast;
          }
        }
        // other bonding config params, failover times, etc.
      }
    }
```

Appendix C. Example: VLAN Interface Module

This section gives an example of how a VLAN interface module can be defined.

```
module example-vlan {
  namespace "http://example.com/vlan";
  prefix "vlan";

  import ietf-interfaces {
    prefix if;
  }
  import iana-if-type {
    prefix ianaift;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ianaift:ethernetCsmacd' or
          if:type = 'ianaift:ieee8023adLag'";
    leaf vlan-tagging {
      type boolean;
      default false;
    }
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ianaift:l2vlan'";

    leaf base-interface {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]"
         + "/vlan:vlan-tagging = 'true'" {
        description
          "The base interface must have VLAN tagging enabled.";
      }
    }
    leaf vlan-id {
      type uint16 {
        range "1..4094";
      }
```

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

Appendix D. Example: NETCONF <get-config> Reply

This section gives an example of a reply to the NETCONF <get-config> request for the running configuration datastore for a device that implements the example data models above.

```
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="101">
  <data>
    <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
        xmlns:vlan="http://example.com/vlan">

      <interface>
        <name>eth0</name>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>false</enabled>
      </interface>

      <interface>
        <name>eth1</name>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <vlan:vlan-tagging>true</vlan:vlan-tagging>
      </interface>

      <interface>
        <name>eth1.10</name>
        <type>ianaift:l2vlan</type>
        <enabled>true</enabled>
        <vlan:base-interface>eth1</vlan:base-interface>
        <vlan:vlan-id>10</vlan:vlan-id>
      </interface>

      <interface>
        <name>lo1</name>
        <type>ianaift:softwareLoopback</type>
```

```
          <enabled>true</enabled>
        </interface>

      </interfaces>
    </data>
  </rpc-reply>
```

Appendix E. Example: NETCONF <get-data> Reply

This section gives an example of a reply to the NETCONF <get-data> request for the operational state datastore for a device that implements the example data models above.

This example uses the "origin" annotation, which is defined in the module "ietf-origin" [RFC8342].

```
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="101">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-datastores">
    <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
        xmlns:vlan="http://example.com/vlan"
        xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">

      <interface or:origin="or:intended">
        <name>eth0</name>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>false</enabled>
        <admin-status>down</admin-status>
        <oper-status>down</oper-status>
        <if-index>2</if-index>
        <phys-address>00:01:02:03:04:05</phys-address>
        <statistics>
          <discontinuity-time>
            2013-04-01T03:00:00+00:00
          </discontinuity-time>
          <!-- counters now shown here -->
        </statistics>
      </interface>

      <interface or:origin="or:intended">
        <name>eth1</name>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <admin-status>up</admin-status>
        <oper-status>up</oper-status>
```

```
      <if-index>7</if-index>
      <phys-address>00:01:02:03:04:06</phys-address>
      <higher-layer-if>eth1.10</higher-layer-if>
      <statistics>
        <discontinuity-time>
          2013-04-01T03:00:00+00:00
        </discontinuity-time>
        <!-- counters now shown here -->
      </statistics>
      <vlan:vlan-tagging>true</vlan:vlan-tagging>
    </interface>

    <interface or:origin="or:intended">
      <name>eth1.10</name>
      <type>ianaift:l2vlan</type>
      <enabled>true</enabled>
      <admin-status>up</admin-status>
      <oper-status>up</oper-status>
      <if-index>9</if-index>
      <lower-layer-if>eth1</lower-layer-if>
      <statistics>
        <discontinuity-time>
          2013-04-01T03:00:00+00:00
        </discontinuity-time>
        <!-- counters now shown here -->
      </statistics>
      <vlan:base-interface>eth1</vlan:base-interface>
      <vlan:vlan-id>10</vlan:vlan-id>
    </interface>

    <!-- This interface is not configured -->
    <interface or:origin="or:system">
      <name>eth2</name>
      <type>ianaift:ethernetCsmacd</type>
      <admin-status>down</admin-status>
      <oper-status>down</oper-status>
      <if-index>8</if-index>
      <phys-address>00:01:02:03:04:07</phys-address>
      <statistics>
        <discontinuity-time>
          2013-04-01T03:00:00+00:00
        </discontinuity-time>
        <!-- counters now shown here -->
      </statistics>
    </interface>

    <interface or:origin="or:intended">
      <name>lo1</name>
```

```
          <type>ianaift:softwareLoopback</type>
          <enabled>true</enabled>
          <admin-status>up</admin-status>
          <oper-status>up</oper-status>
          <if-index>1</if-index>
          <statistics>
            <discontinuity-time>
              2013-04-01T03:00:00+00:00
            </discontinuity-time>
            <!-- counters now shown here -->
          </statistics>
        </interface>

      </interfaces>
    </data>
  </rpc-reply>
```

Appendix F. Examples: Interface Naming Schemes

This section gives examples of some implementation strategies.

The examples make use of the example data model "example-vlan" (see Appendix C) to show how user-controlled interfaces can be configured.

F.1. Router with Restricted Interface Names

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

The device-specific names for these physical interfaces are "fastethernet-N/M" or "gigabitethernet-N/M".

The name of a VLAN interface is restricted to the form "<physical-interface-name>.<subinterface-number>".

It is assumed that the operator is aware of this naming scheme. The implementation auto-initializes the value for "type" based on the interface name.

The NETCONF server does not advertise the "arbitrary-names" feature in the <hello> message.

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

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

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

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

The client can configure a VLAN interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>fastethernet-1/0.10005</name>
  <type>ianaift:l2vlan</type>
  <vlan:base-interface>fastethernet-1/0</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
</interface>
```

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

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

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

F.2. Router with Arbitrary Interface Names

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

The device-specific names for these physical interfaces are "fastethernet-N/M" or "gigabitethernet-N/M".

The implementation does not restrict the user-controlled interface names. This allows an operator to more easily apply the interface configuration to a different interface. However, the additional level of indirection also makes it a bit more complex to map interface names found in other protocols to configuration entries.

The NETCONF server advertises the "arbitrary-names" feature in the <hello> message.

Physical interfaces are configured as in Appendix F.1.

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

```
<interface nc:operation="create">
  <name>acme-interface</name>
  <type>ianaift:l2vlan</type>
  <vlan:base-interface>fastethernet-1/0</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
</interface>
```

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

```
<interface nc:operation="merge">
  <name>acme-interface</name>
  <vlan:base-interface>fastethernet-1/1</vlan:base-interface>
</interface>
```

F.3. Ethernet Switch with Restricted Interface Names

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

The device-specific names for the physical interfaces are numbers that match the physical port number.

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

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

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

```
<interface>
  <name>6</name>
  <type>ianaift:ethernetCsmacd</type>
</interface>
```

F.4. Generic Host with Restricted Interface Names

In this example, a generic host has interfaces named by the kernel. The system identifies the physical interface by the name assigned by the operating system to the interface.

The name of a VLAN interface is restricted to the form "<physical-interface-name>:<vlan-number>".

The NETCONF server does not advertise the "arbitrary-names" feature in the <hello> message.

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

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

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

```
<interface>
  <name>eth8</name>
  <type>ianaift:ethernetCsmacd</type>
</interface>
```

The client can configure a VLAN interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>eth8:5</name>
  <type>ianaift:l2vlan</type>
  <vlan:base-interface>eth8</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
</interface>
```

F.5. Generic Host with Arbitrary Interface Names

In this example, a generic host has interfaces named by the kernel. The system identifies the physical interface by the name assigned by the operating system to the interface.

The implementation does not restrict the user-controlled interface names. This allows an operator to more easily apply the interface configuration to a different interface. However, the additional level of indirection also makes it a bit more complex to map interface names found in other protocols to configuration entries.

The NETCONF server advertises the "arbitrary-names" feature in the <hello> message.

Physical interfaces are configured as in Appendix F.4.

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

```
<interface nc:operation="create">
  <name>acme-interface</name>
  <type>ianaift:l2vlan</type>
  <vlan:base-interface>eth8</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
</interface>
```

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

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

Acknowledgments

The author wishes to thank Alexander Clemm, Per Hedeland, Ladislav Lhotka, and Juergen Schoenwaelder for their helpful comments.

Author's Address

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com