Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 11, 2011

M. Bjorklund
Tail-f Systems

December 8, 2010

# A YANG Data Model for Interface Configuration
## draft-bjorklund-netmod-interfaces-cfg-00

Abstract

This document defines a YANG data model for the configuration of network interfaces. It is expected that interface type specific configuration data models augment the generic interfaces data model defined in this document.

Status of this Memo

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

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

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

This Internet-Draft will expire on June 11, 2011.

Table of Contents

## 1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of network interfaces. It is expected that interface type specific configuration data models augment the generic interfaces data model defined in this document.

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

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119].

## 2. Objectives

This section describes some of the design objectives for the model presented in Section 4.

- It is recognized that existing implementations will have to map the interface data model defined in this memo to their proprietary native data model. The new data model should be simple to facilitate such mappings.

- The data model should be suitable for new implementations to use as-is, without requiring a mapping to a different native model.

- The data model must be extensible for different specific interface types, including vendor-specific types.

- References to interfaces should be as simple as possible, preferably by using a single leafref.

- The mapping to ifIndex [RFC2863] used by SNMP to identify interfaces must be clear.

- The model must support interface layering, both simple layering where one interface is layered on top of exactly one other interface, and more complex scenarios where one interface is aggregated over N other interfaces, or when N interfaces are multiplexed over one other interface.

- The data model should support the pre-provisioning of interface configuration, i.e, it should be possible to configure an interface whose physical interface hardware is not present on the device. It is recommended that devices that supports dynamic addition and removal of physical interfaces also support pre-provisioning.

## 3. Interfaces Data Model

### 3.1. The interface List

The data model for interface configuration presented in this document uses a flat list of interfaces. Each interface in the list is identified by its name. Furthermore, each interface has a mandatory "type" leaf, and a "location" leaf. The combination of "type" and "location" is unique within the interface list.

The "type" is a YANG identity which must be derived from the base identity "interface-type". By using an identity instead of an enumeration, the definition of interface types is decentralized. Other standard or vendor-specific data models can define their own interface types without having to update a central data model.

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

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

```
import interfaces {
    prefix "if";
}

augment "/if:interfaces/if:interface" {
    when "if:type = 'ethernet'";
    container ethernet {
        leaf duplex {
            ...
        }
    }
}
```

The "location" leaf is a string. It is optional in the data model, but if the type represents a physical interface, it is mandatory. The format of this string is device- and type-dependent. The device uses the location string to identify the physical or logical entity that the configuration applies to. For example, if a device has a single array of 8 ethernet ports, the location can be one of the strings "1" to "8". As another example, if a device has N cards of M ports, the location can be on the form "n/m", such as "1/0".

How a client can learn which types and locations are present on a certain device is outside the scope of this document.

### 3.2. Interface References

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

### 3.3. Interface Layering

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

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

```
  identity eth-bonding {
      base if:interface-type;
  }

  augment "/if:interfaces/if:interface" {
      when "if:type = eth-bonding";

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

## 4. Interfaces YANG module

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

```
<CODE BEGINS> file "ietf-interfaces@2010-12-08.yang"

module ietf-interfaces {

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

  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: David Kessens
               <mailto:david.kessens@nsn.com>

     WG Chair: Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

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

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

     Copyright (c) 2010 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

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

```
    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    revision 2010-12-08 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for Interface Configuration";
    }

    /* 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.";
    }

    /* Features */

    feature snmp-if-mib {
      description
        "This feature indicates that the server implements IF-MIB,
         accessible over SNMP.";
    }

    /* Identities */

    identity interface-type {
      description
        "The base identity from which media-specific interface
         identities are derived.";
    }

    /* Data nodes */

    container interfaces {
      description
        "Interface parameters.";

      list interface {
        key "name";
        unique "type location";

        description
          "The list of configurable interfaces on the device.";
```

```
        leaf name {
          type string {
            length "1..255";
          }
          description
            "An arbitrary name for the interface.

             A device MAY restrict the allowed values for this leaf,
             possibly depending on the type and location.";
        }

        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.";
        }

        leaf location {
          type string;
          description
            "The device-specific location of the interface of a
             particular type.  The format of the location string
             depends on the interface type and the device.

             Media-specific modules must specify if the location
             is needed for the given type.

             For example, if a device has a single array of 8 ethernet
             ports, the location can be one of '1' to '8'.  As another
             example, if a device has N cards of M ports, the location
             can be on the form 'n/m'.

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

        leaf admin-status {
          type enumeration {
```

```
            enum "up";
            enum "down";
          }
          default "up";
          description
            "The desired state of the interface.

             This leaf contains the configured, desired state of the
             interface.  Systems that implement the IF-MIB use the
             value of this leaf to set IF-MIB.ifAdminStatus after an
             ifEntry has been initialized, as described in RFC 2863.";
          // FIXME: Can we say that changing ifAdminStatus does NOT
          //        change this object?  If not, is the opposite
          //        always true, i.e. that changing ifAdminStatus
          //        results in a change of this object (in running)?
          //        Or should we be silent?
          reference
            "RFC 2863: The Interfaces Group MIB - ifAdminStatus";
        }

        leaf-list if-index {
          if-feature snmp-if-mib;
          type int32 {
            range "1..2147483647";
          }
          config false;
          description
            "The list of ifIndex values for all ifEntries that are
             represented by this interface.  If there is a one-to-one
             mapping between the interface and entries in the ifTable,
             this leaf-list will have a single value.

             Media-specific modules must specify how the type is
             mapped to entries in the ifTable.";
          reference
            "RFC 2863: The Interfaces Group MIB - ifIndex";
        }

        leaf mtu {
          type uint32;
          description
            "The size, in octets, of the largest packet that the
             interface can send and receive.  This node might not be
             valid for all interface types.

             Media-specific modules must specify any restrictions on
             the mtu for their interface type.";
        }
```

```
      }
    }
  }
```

<CODE ENDS>

## 5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

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

Registrant Contact: The NETMOD WG of the IETF.

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 XXXX
```

## 6. Security Considerations

TBD.

## 7. Acknowledgments

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

## 8. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

## Appendix A. Example: Ethernet Interface Module

This section gives a simple example of how an Ethernet interface module could be defined. It demonstrates how a media-specific type can be derived from the base identity "interface-type", and how media-specific configuration parameters can be conditionally augmented to the generic interface list. It is not intended as a complete module for ethernet configuration.

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

     import ietf-interfaces {
       prefix if;
     }

     identity ethernet {
       base if:interface-type;
     }

     augment "/if:interfaces/if:interface" {
       when "if:type = 'eth:ethernet'";
       container ethernet {
         must "../if:location" {
           description
             "An ethernet interface must specify the physical location
              of the ethernet hardware.";
         }
         choice transmission-params {
           case auto {
             leaf auto-negotiate {
               type empty;
             }
           }
           case manual {
             leaf duplex {
               type enumeration {
                 enum "half";
                 enum "full";
               }
             }
             leaf speed {
               type enumeration {
                 enum "10Mb";
                 enum "100Mb";
                 enum "1Gb";
                 enum "10Gb";
               }
             }
           }
         }
         // 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 type is defined, which bonds several ethernet interfaces into one logical interface.

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

  import ietf-interfaces {
    prefix if;
  }
  import ex-ethernet {
    prefix eth;
  }

  identity eth-bonding {
    base if:interface-type;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'bond:eth-bonding'";
    leaf-list slave-if {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]"
         + "/if:type = 'eth:ethernet'" {
        description
          "The type of a slave interface must be ethernet.";
      }
    }
    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 vlan interface module can be defined.

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

  import ietf-interfaces {
    prefix if;
  }
  import ex-ethernet {
    prefix eth;
  }
  import ex-ethernet-bonding {
    prefix bond;
  }

  identity vlan {
    base if:interface-type;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'eth:ethernet' or
          if:type = 'bond:eth-bonding'";
    // Should we list all types that support vlan tagging here, or
    // should we just remove the when, and state in text that not
    // all interfaces support this?
    leaf vlan-tagging {
      type boolean;
      default false;
    }
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'vlan:vlan'";
    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";
    }
  }
}
```

## Appendix D. Example: IP Module

This section gives an example how an IP module can be defined.

```
module ex-ip {

  namespace "http://example.com/ip";
  prefix "ip";

  import ietf-interfaces {
    prefix if;
  }

  import ietf-inet-types {
    prefix inet;
  }

  augment "/if:interfaces/if:interface" {
    container ip {
      list address {
        key "ip";
        leaf ip {
          type inet:ip-address;
        }
        leaf prefix-length {
          type uint16;
          // range depends on type of address
        }
      }
    }
  }
}
```

Author's Address

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com