Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 22, 2016

A. Lindem
Cisco
L. Berger, Ed.
LabN
D. Bogdanovic

C. Hopps
Deutsche Telekom

September 21, 2015

Network Device YANG Organizational Model
draft-rtgyangdt-rtgwg-device-model-01

Abstract

This document presents an approach for organizing YANG models in a comprehensive structure that defines how individual models may be composed to configure and operate network infrastructure and services. The structure is itself represented as a YANG model, with all of the related component models logically organized in a way that is operationally intuitive. This document is derived from work submitted to the IETF by members of the informal OpenConfig working group of network operators and is a product of the Routing Area YANG Architecture design team.

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 September 10, 2015.

Copyright Notice

Contents

1. Introduction

"Operational Structure and Organization of YANG Models" [OC-STRUCT], highlights the value of organizing individual, self-standing YANG [RFC6020] models into a more comprehensive structure. This document builds on that work and presents a derivative structure for use in representing the networking infrastructure aspects of physical and virtual devices. While [OC-STRUCT] and earlier versions of this document presented a single device-centric model root, this document no longer contains this element.

This document aims to provide an extensible structure that can be used to tie together other models. It allows for existing, emerging, and future models. The overall structure can be constructed using YANG augmentation and imports.

This document was motivated by, and derived from, [OC-STRUCT]. The requirements from that document have been combined with the requirements from "Consistent Modeling of Operational State Data in YANG", [OC-OPSTATE], into "NETMOD Operational State Requirements", [NETMOD-OPSTATE]. This document is aimed at the requirement related to a common model-structure, currently Requirement 7, and also aims to provide a modeling base for Operational State representation.

The approach taken in this (and the original) document is to organize the models describing various aspects of network infrastructure, focusing on devices, their subsystems, and relevant protocols operating at the link and network layers. The proposal does not consider a common model for higher level network services, nor does it specify details of how hardware-related data should be organized. We focus on the set of models that are commonly used by network operators, and suggest a corresponding organization.

A significant portion of the text and model contained in this document was taken from the -00 of [OC-STRUCT].

1.1. Status of Work and Open Issues

This version of the document and structure are a product of the Routing Area YANG Architecture design team and is very much a work in progress rather than a final proposal. Recent discussions have been quite focused on the single device root, /device, that was in the -00 version of this document and [OC-STRUCT]. Secondary discussions have focused on representation of logical network elements. Disagreement and open issues remain, even within the design team. Major open issues are as follows:

1. The structure related to L2VPNs, Ethernet services, and virtual switching instances has not yet received sufficient discussion and is likely to change.

2. Additional discussion and text is need to ensure that the interpretation of different policy containers is clear.

3. Configuration information related to network-instance interconnection (over a "core" network) is currently commingled with configuration related to operation within the instance.

4. The representation of operational state is currently missing. The model will be updated once the "opstate" requirements are addressed.

5. Interface logical network element id and logical networking instance name augmentations are currently defined within the context of network-device. It may clearer to define these each in their own modules.

6. There is a proposal on the table to support logical network elements using an enhanced version of [NETMOD-MOUNT]. The authors and design team are open to this proposal and are interested in discussing this option as specific details are available.

2. Module Overview

In this document, we consider network devices that support protocols and functions defined within the IETF Routing Area, e.g, routers, firewalls and hosts. Such devices may be physical or virtual, e.g., a classic router with custom hardware or one residing within a server-based virtual machine implementing a virtual network function (VNF). Each device may sub-divide their resources into logical network elements (LNEs) each of which provides a managed logical device. Examples of vendor terminology for an LNE include logical system or router, and virtual switch, chassis, or fabric. Each LNE may also support virtual routing and forwarding (VRF) and virtual switching instance (VSI) functions, which are referred to below as a networking instances (NIs). This breakdown is represented in Figure 1.

A model for LNEs is described in Section 2.2 and the sub-model for networking instances is covered in Section 2.2.2.

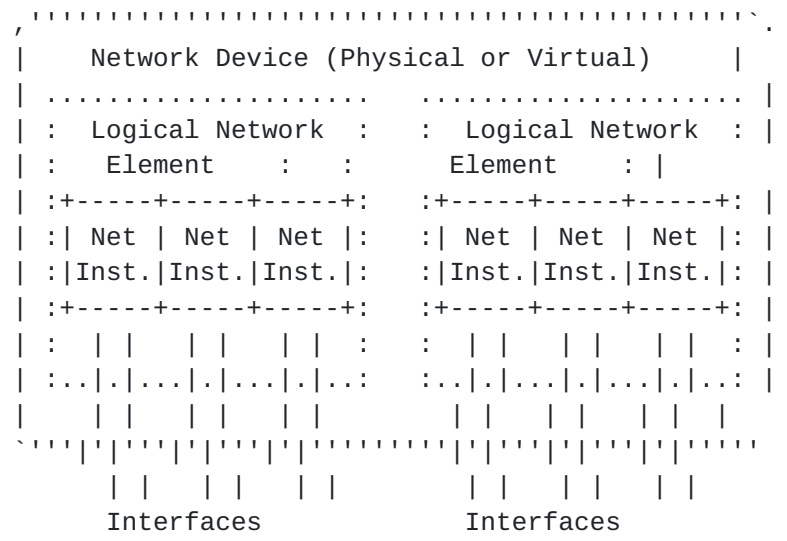

Figure 1: Module Element Relationships

The presented module can itself be thought of as a "meta-model" as it describes the relationships between individual models. We choose to represent it also as a simple YANG module consisting of models, lists and containers to serve as anchor points for the corresponding individual models.

The module does not follow the hierarchy of any particular implementation, and hence is vendor-neutral. Nevertheless, the structure should be familiar to network operators and also readily mapped to vendor implementations. The overall structure is:

```
module: network-device
+--rw info
|  +--rw device-type?   enumeration
+--rw hardware
+--rw qos
+--rw logical-network-elements
|    ...
augment /if:interfaces/if:interface:
      ...
```

The top level models generally represent resources that are associated with a device that can themselves be assigned to LNEs. Notably, the existing Interface Management model [RFC7223] is also included at the top level, although it is augmented to allow for LNEs. An info section is included for basic device information such as its type (e.g., physical or virtual), vendor, model, etc. The hardware section is a placeholder for device-specific configuration and operational state data. For example, a common structure for the hardware model might include chassis, line cards, and ports, but we leave this unspecified. The Quality of Service (qos) section is a placeholder for device-wide configuration and operational state data which relates to the treatment of traffic across the device.

2.1. Interface Model Components

Interfaces are a crucial part of any network device's configuration and operational state. They generally include a combination of raw physical interfaces, link-layer interfaces, addressing configuration, and logical interfaces that may not be tied to any physical interface. Several system services, and layer 2 and layer 3 protocols may also associate configuration or operational state data with different types of interfaces (these relationships are not shown for simplicity).

This document augments the existing Interface Management model [RFC7223] in two ways. The first is add an identifier which is used on physical interface types to identify an associated LNE. The second is to add a name which is used on sub-interface types to identify an associated networking instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [RFC7277].

The interface related definitions are as follows:

```
augment /if:interfaces/if:interface:
 +--rw bind-network-element-id?      uint8
augment /if:interfaces/if:interface:
 +--rw bind-networking-instance-name?   string
```

```
  augment /if:interfaces/if:interface/ip:ipv4:
   +--rw bind-networking-instance-name?   string
  augment /if:interfaces/if:interface/ip:ipv6:
   +--rw bind-networking-instance-name?   string
```

The following is an example of envisioned usage. The interfaces container includes a number of commonly used components as examples:

```
        +--rw interfaces
        |  +--rw interface* [name]
        |    +--rw name                    string
        |    +--rw bind-network-element-id?   uint8
        |    +--rw ethernet
        |    |  +--rw bind-networking-instance-name?   string
        |    |  +--rw aggregates
        |    |  +--rw rstp
        |    |  +--rw lldp
        |    |  +--rw ptp
        |    +--rw vlans
        |    +--rw tunnels
        |    +--rw ipv4
        |    |  +--rw bind-networking-instance-name?   string
        |    |  +--rw arp
        |    |  +--rw icmp
        |    |  +--rw vrrp
        |    |  +--rw dhcp-client
        |    +--rw ipv6
        |     +--rw bind-networking-instance-name?   string
        |     +--rw vrrp
        |     +--rw icmpv6
        |     +--rw nd
        |     +--rw dhcpv6-client
```

The bind-networking-instance-name leaf is an explicit and notable addition. The [RFC7223] defined interface model is structured to include all interfaces in a flat list, without regard to logical or virtual instances (e.g., VRFs) supported on the device. The bind-networking-instance-name leaf provides the association between an interface and its associated networking instance (e.g., VRF or VSI).

2.2. Logical Network Elements

Logical network elements represent the capability on some devices to partition resources into independent logical routers and/or switches. Device support for multiple logical network elements is implementation specific. Systems without such capabilities will have just a single container. In physical devices, some hardware features

are shared across partitions, but control plane (e.g., routing) protocol instances, tables, and configuration are managed separately. For example, in virtual routers or VNFs, this may correspond to establishing multiple logical instances using a single software installation. The model supports configuration of multiple instances on a single device by creating a list of logical network elements, each with their own configuration and operational state related to routing and switching protocols, as shown below:

```
  module: network-device
        +--rw logical-network-elements
        +--rw logical-network-element* [network-element-id]
              +--rw network-element-id                uint8
              +--rw network-element-name?             string
              +--rw default-networking-instance-name?   string
              +--rw system-management
              |  ...
              +--rw ietf-acl
              +--rw ietf-key-chain
              +--rw networking-instances
              |  ...
```

Network-element-id and network-element-name identify the logical network element.

Default-networking-instance-name identifies the networking instance to use for system management connectivity. Other instances may access system management function through appropriate inter-instance configuration.

2.2.1. System Management

The model supports the potentially independent system management functions and configuration per logical network element. This permits, for example, different users to manage either the whole device or just the associated logical network element. System management is supported by the system-management container which is expected to reuse definitions contained in [RFC7317] and is shown below:

```
  module: network-device
  +--rw logical-network-elements
        +--rw logical-network-element* [network-element-id]
        +--rw system-management
              +--rw device-view?                boolean
              +--rw system-management-global
              +--rw system-management-protocol* [type]
              +--rw type identityref
```

The device-view leaf is used to indicate if the system management functions associated with the logical network element are restricted to the logical network element or can manage the whole device. The leaf may have a fixed value. For example, some implementations may only support management on a device-wide basis. Additional information on the implications of this leaf can be found in Section 2.3.

System-management-global is used for configuration information and state that is independent of a particular management protocol. System-management-protocol is a list of management protocol specific elements. The type-specific sub-modules are expected to be defined.

The following is an example of envisioned usage:

```
  module: network-device
        +--rw logical-network-elements
              +--rw system-management
              +--rw device-view?                        boolean
              +--rw system-management-global
              |  +--rw statistics-collection
              |  ...
              +--rw system-management-protocol* [type]
              |  +--rw syslog
              |  +--rw dns
              |  +--rw ntp
              |  +--rw ssh
              |  +--rw tacacs
              |  +--rw snmp
              |  +--rw netconf
```

2.2.2. Network Instances

The network instance container is used to represent virtual routing and forwarding instances (VRFs) and virtual switching instances (VSIs), [RFC4026]. VRFs and VSIs are commonly used to isolate routing and switching domains, for example to create virtual private networks, each with their own active protocols and routing/switching policies. The model represents both core/provider and virtual instances. Network instances reuse and build on [RTG-CFG] and are shown below:

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw networking-instance-name     string
```

```
                        +--rw type?                          identityref
                        +--rw enabled?                       boolean
                        +--rw router-id?                     uint32
                        +--rw description?                   string
                        +--rw oam-protocols
                        |  ...
                        +--rw networking-instance-policy
                        |  ...
                        +--rw control-plane-protocols
                        |  ...
                        +--rw ribs
                        |  ...
                        +--rw mpls
                        |  ...
                        +--rw networking-services
                        ...
```

[Editor's note: L2/MAC forwarding table is TBD]

2.2.2.1. OAM Protocols

OAM protocols that may run within the context of a network instance are grouped. The type identifyref is used to identify the information and state that may relate to a specific OAM protocol. The defined structure is as follows:

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw oam-protocols
                    +--rw oam-protocol* [type]
                          +--rw type identityref
```

The following is an example of envisioned usage. Examples shown below include Bi-directional Forwarding Detection (BFD), Ethernet Connectivity Fault Management (CFM), and Two-Way Active Measurement Protocol (TWAMP):

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw oam-protocols
                    +--rw oam-protocol* [type]
                          +--rw bfd
                          +--rw cfm
                          +--rw twamp
```

2.2.2.2. Network Instance Policy

Network instance policies are used to control provider instances, VRF routing policies, and VRF/VSI identifiers. Examples include BGP route targets (RTs) and route distinguishers (RDs), if the instances is a core/provider instance, virtual network identifiers(VN-IDs), VPLS neighbors, etc. The structure is:

The following is an example of envisioned usage:

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw networking-instance-policy
                    (TBD)
```

2.2.2.3. Control Plane Protocols

Control plane protocols that may run within the context of a network instance are grouped. Each protocol is expected to have its own model, which is indicated by the type identityref. Protocol specific policy is included with the protocol rather than being combined in a separate generic policy grouping. The rationale behind this is that this ensures that only protocol relevant policies may be configured. A reusable or common approach may still be leveraged in creating these policy groupings, perhaps based on [RTG-POLICY]. The defined structure is as follows:

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw control-plane-protocols
                    +--rw control-plane-protocol* [type]
                          +--rw type identityref
                          +--rw policy
```

The following is an example of envisioned usage:

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw control-plane-protocols
                    +--rw control-plane-protocol* [type]
                          +--rw bgp
```

```
                              +--rw is-is
                              +--rw ospf
                              +--rw rsvp
                              +--rw segment-routing
                              +--rw ldp
                              +--rw pim
                              +--rw igmp
                              +--rw mld
                              +--rw static-routes
```

2.2.2.4. RIBs

Every routing instance manages one or more routing information bases (RIB). A RIB is a list of routes complemented with administrative data. RIBs reuse and build on [RTG-CFG] and are shown below:

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw ribs
                    +--rw rib* [name]
                          +--rw name        string
                          +--rw description?   string
                          +--rw policy
```

2.2.2.5. MPLS

MPLS data plane related information is grouped together. MPLS control plane protocols are included above. MPLS may reuse and build on [OC-MPLS] or other emerging models and is shown below:

```
  module: network-device
        +--rw logical-network-elements
              +--rw networking-instances
              +--rw networking-instance* [networking-instance-name]
                    +--rw mpls
                    +--rw global
                    +--rw label-switched-paths
                    +--rw constrained-path
                    +--rw igp-congruent
                    +--rw static
```

2.2.2.6. Networking Services

A device may provide services to other devices within the scope of a networking instance. The type identifyref is used to identify the service specific configuration and state information. The defined

structure is as follows:

```
module: network-device
      +--rw logical-network-elements
            +--rw networking-instances
            +--rw networking-instance* [networking-instance-name]
                  +--rw networking-services
                  +--rw networking-service* [type]
                   +--rw type identityref
```

The following is an example of envisioned usage: Examples shown below include a device-based Network Time Protocol (NTP) server, a Domain Name System (DNS) server, and a Dynamic Host Configuration Protocol (DHCP) server:

```
module: network-device
      +--rw logical-network-elements
            +--rw networking-instances
            +--rw networking-instance* [networking-instance-name]
                  +--rw networking-services
                  +--rw networking-service* [type]
                        +--rw ntp-server
                        +--rw dns-server
                        +--rw dhcp-server
```

2.3. Device View vs Logical Network Element (LNE) View Management

[Editor's note: an alternate approach based on a future enhanced version of [NETMOD-MOUNT] has been proposed and is being considered. This section would be replaced if such an alternate approach is followed.]

On some devices it is possible to limit control and management to a scoped set of system resources. As stated above in Section 2.2., the documented approach supports this capability using logical network elements and the system management device-view leaf.

When the device-view leaf is set to true, information accessible via a logical network element's system management functions represents the complete device. This applies to all system management functions, not just those represented in the YANG model. When viewing information represented in a YANG model, the device model will cover the full device and allow management across all logical network elements.

The case when a logical network element's system management functions do not have a device wide view is more complex. In this case, there

are two perspectives: one from functions that are operating within a context of a logical network element that has a device wide view (or more simply have a "device view"); and the other from functions that are operating within the context of a logical network element that has only a logical network element view (or more simply have an "LNE view").

From a management function operating with a device view, the limited logical network element's system management device-view leaf is simply set to false.

Management functions operating with an LNE view can only see information (e.g., resources, interfaces, configuration, operational state, etc.) associated with in the logical network element. When viewing information represented in a YANG model, a full device model (as defined in this document) is available from within the view, but it includes only those elements associated with the LNE. For information contained with the logical-network-element container entry, this is the same information as available in a device wide view. Information outside the logical-network-elements container is made available within an LNE view as is appropriated based on device wide configuration. For example, interfaces assigned to the logical network element can be managed from within the LNE view. Note: some information that can be modified from a device view may be read-only from within the LNE view.

Multiple implementation approaches are possible to provide LNE views, and these are outside the scope of this document.

3. Populating the structural model

The structural model in this document describes how individual YANG models may be used together to represent the configuration and operational state for all parts of a physical or virtual device. It does not, however, document the actual model in its entirety. In this section, we outline an option for creating the full model and also describe how it may be used.

3.1. Constructing the device model

One of the challenges in assembling existing YANG models is that they are generally written with the assumption that each model is at the root of the configuration or state tree. Combining models then results in a multi-rooted tree that does not follow any logical construction and makes it difficult to work with operationally. In some cases, models explicitly reference other models (e.g., via augmentation) to define a relationship, but this is the case for only a few existing models.

Some examples include the interfaces [RFC7223] and IP management [RFC7277] models, and proposed IS-IS [RTG-ISIS], OSPF [RTG-OSPF] and routing configuration [RTG-CFG] models.

3.2. "Pull" approach for model composition

To enable model composition, one possible approach is to avoid using root-level containers in individual component models. Instead, the top level container (and all other data definitions) can be enclosed in a YANG 'grouping' statement so that when the model is imported by another model, its location in the configuration tree can be controlled by the importing YANG module with the 'uses' statement. One advantage of this approach is that the importing module has the flexibility to readily use the data definitions where the author deems appropriate.

One obvious drawback is that individual models no longer contain any of their own data definitions and must be used by a higher-level model for their data nodes to become active. Some judgment as to which models are more suited for inclusion in higher level models is also necessary to decide when the corresponding YANG module should only contain groupings. Another potential drawback is that this approach does not define a common structure for models to fit together, limiting interoperability due to implementations using different structures. To address this, a top-level standard model structure could be defined and updated to import new models into the hierarchy as they are defined.

3.3. "Push" approach for model composition

An alternative approach is to develop a top level model which defines the overall structure of the models, similar to the structure described in Section 2. Individual models may augment the top level model with their data nodes in the appropriate locations. The drawback is the need for a pre-defined top level model structure. On the other hand, when this top level model is standardized, it can become the basis for a vendor-neutral way to manage devices, assuming that the component models are supported by a given implementation.

One question in both approaches is what the root of the top-level model should be. In this document we selected to base the model at a device because this layer should be common across many use cases and implementations. Starting at a higher layer (e.g., services) makes defining and agreeing on a common organization more challenging as discussed in Section 1.1.

Ideally, one could consider a hybrid construction mechanism that

supports both styles of model composition. For example, a YANG compiler or preprocessing directive could be used to indicate whether an individual model should assume it is at the root, or whether it is meant for inclusion in other higher-level models.

4. Security Considerations

The model structure described in this document does not define actual configuration and state data, hence it is not directly responsible for security risks.

However, each of the component models that provide the corresponding configuration and state data should be considered sensitive from a security standpoint since they generally manipulate aspects of network configurations. Each component model should be carefully evaluated to determine its security risks, along with mitigations to reduce such risks.

5. IANA Considerations

This YANG model currently uses a temporary ad-hoc namespace. If it is placed or redirected for the standards track, an appropriate namespace URI will be registered in the "IETF XML Registry" [RFC3688]. The YANG structure modules will be registered in the "YANG Module Names" registry [RFC6020].

6. YANG module

The model structure is described by the YANG module below.

6.1. Model structure

```
<CODE BEGINS> file "network-device.yang"
module network-device {

  yang-version "1";

  // namespace
  namespace "urn:ietf:params:xml:ns:yang:network-device";

  prefix "struct";

  // import some basic types
  import ietf-interfaces {
  prefix if;
  }
```

```
     import ietf-ip {
     prefix ip;
     }

     // meta
     organization "IETF RTG YANG Design Team Collaboration
                 with OpenConfig";

     contact
      "Routing Area YANG Architecture Design Team -
           <rtg-dt-yang-arch@ietf.org>";

     description
     "This module describes a model structure for YANG
     configuration and operational state data models. Its intent is
     to describe how individual device protocol and feature models
     fit together and interact.";

     revision "2015-09-06" {
     description
      "IETF Routing YANG Design Team Meta-Model";
     reference "TBD";
     }

     // extension statements

     // feature statements
     feature bind-network-element-id {
     description
      "Logical network element ID to which an interface is bound";
     }

     feature bind-networking-instance-name {
     description
      "Networking Instance to which an interface instance is bound";
     }

     // identity statements

     identity networking-instance {
      description
           "Base identity from which identities describing
           networking instance types are derived.";
     }

     identity oam-protocol-type {
      description
           "Base identity for derivation of OAM protocols";
```

```
    }

    identity networking-service-type {
     description
          "Base identity for derivation of networking services";
    }

    identity ethernet-protocol-type {
     description
          "Base identity for derivation of ethernet
          protocols";
    }

    identity ipv4-interface-protocol-type {
     description
          "Base identity for derivation of IPv4 interface
          protocols";
    }

    identity ipv6-interface-protocol-type {
     description
          "Base identity for derivation of IPv6 interface
          protocols";
    }

    identity mpls-protocol-type {
     description
          "Base identity for derivation of MPLS protocols";
    }

    identity control-plane-protocol-type {
     description
          "Base identity for derivation of control-plane protocols";
    }

    identity system-management-protocol-type {
     description
          "Base identity for derivation of system management
          protocols";
    }

    identity oam-service-type {
     description
          "Base identity for derivation of Operations,
          Administration, and Maintenance (OAM) services.";
    }

    identity aaa-service-type {
```

```
      description
           "Base identity for derivation of Authentication,
           Authorization, and Accounting (AAA) services.";
     }

     // typedef statements

     // grouping statements

     grouping interface-ip-common {
     description
      "interface-specific configuration for IP interfaces, IPv4 and
      IPv6";

     }

     grouping ethernet-protocols {
      description
           "Grouping for ethernet protocols configured
            on an interface";
      container ethernet-protocols {
           description
                 "Container for list of ethernet protocols configured
                 on an interface";
           list ethernet-protocol {
                 key "type";
                 description
                  "List of ethernet protocols configured
                       on an interface";
                 leaf type {
                  type identityref {
                       base ethernet-protocol-type;
                  }
                  mandatory true;
                  description
                       "Aggregates, RSTP, LLDP, PTP, etc.";
                 }
           }
      }
     }

     grouping ipv4-interface-protocols {
      container ipv4-interface-protocols {
           list ipv4-interface-protocol {
                 key "type";
                 leaf type {
                  type identityref {
                       base ipv4-interface-protocol-type;
```

```
                 }
                 mandatory true;
                 description
                      "ARP, ICMP, VRRP, DHCP Client, etc.";
                }
                description
                 "List of IPv4 protocols configured
                      on an interface";
          }
          description
                "Container for list of IPv4 protocols configured
                on an interface";
      }
      description
           "Grouping for IPv4 protocols configured on an interface";
     }

     grouping ipv6-interface-protocols {
      description
           "Grouping for IPv6 protocols configured on
           an interface.";
      container ipv6-interface-protocols {
           description
                "Container for list of IPv6 protocols configured
                on an interface.";
           list ipv6-interface-protocol {
                key "type";
                description
                 "List of IPv6 protocols configured
                      on an interface";
                leaf type {
                 type identityref {
                      base ipv6-interface-protocol-type;
                 }
                 mandatory true;
                 description
                      "ND, ICMPv6, VRRP, DHCPv6 Client, etc.";
                }
           }
      }
     }

     grouping router-id {
      description
           "This grouping provides router ID.";
      leaf router-id {
           type uint32;  // yang:dotted-quad
           description
            "A 32-bit number in the form of a dotted quad that is
```

```
                used by some routing protocols identifying a router.";
            reference
                "RFC 2328: OSPF Version 2.";
        }
      }

      grouping oam-protocols {
        container oam-protocols {
            list oam-protocol {
                key "type";
                leaf type {
                  type identityref {
                      base oam-protocol-type;
                  }
                  mandatory true;
                  description
                      "The Operations, Administration, and
                      Maintenance (OAM) protocol type, e.g., BFD,
                      TWAMP, CFM, etc.";
                }
                description
                  "List of OAM protocols configured for a
                      networking instance.";
            }
            description
                "Container for list of OAM protocols configured for a
                networking instance.";
        }
        description
            "Grouping for OAM protocols configured for a
            networking instance.";
      }

      grouping mpls {
      description
            "Grouping for MPLS and TE configuration configured for
            a networking-instance.";
      container mpls {
        description
            "Container for MPLS and TE configuration for a
            networking-instance.";
        container global {
            description "Global MPLS configuration";
        }
        list mpls-protocol {
            key "type";
            description
                "List of MPLS protocols configured for a
                networking instance.";
```

```
           leaf type {
                 type identityref {
                  base mpls-protocol-type;
                 }
                 mandatory true;
                 description
                  "MPLS and Traffic Engineering protocol type,
                       MPLS static, LDP, RSVP TE, etc.";
           }
      }
     }
     }

     grouping networking-instance-policy {
     description
           "Networking instance policies such as route
           distinguisher, route targets, VPLS ID and neighbor,
           Ethernet ID, etc. ";
     reference
           "RFC 4364 - BGP/MPLS Virtual Private Networks (VPNs)
           RFC 6074 - Provisioning, Auto-Discovery, and Signaling
                 in Layer 2 Virtual Private Networks (L2VPNs)
           RFC 7432 - BGP MPLS-Based Ethernet VPN";
     container networking-instance-policy {
           description "Networking Instance Policy -- details TBD";
     }
     }

     grouping control-plane-protocols {
      description
           "Grouping for control plane protocols configured for
           a networking-instance";
      container control-plane-protocols {
           description
                 "Container for control plane protocols configured for
                 a networking instance.";
           list control-plane-protocol {
                 key "type";
                 description
                  "List of control plane protocols configured for
                       a networking instance.";
                 leaf type {
                  type identityref {
                       base control-plane-protocol-type;
                  }
                  mandatory true;
                  description
                       "The control plane protocol type, e.g., BGP,
                       OSPF IS-IS, etc";
```

```
                }
                container policy {
                 description
                      "Protocol specific policy,
                      reusing [RTG-POLICY]";
                }
          }
      }
     }

     grouping ribs {
     description
      "Routing Information Bases (RIBs) supported by a
           networking-instance";
     container ribs {
           description
            "RIBs supported by a networking-instance";
           list rib {
            key "name";
            min-elements "1";
            description
                 "Each entry represents a RIB identified by the
                 'name' key. All routes in a RIB must belong to the
                 same address family.

                 For each routing instance, an implementation should
                 provide one system-controlled default RIB for each
                 supported address family.";
            leaf name {
                 type string;
                 description
                       "The name of the RIB.";
            }
            reference "draft-ietf-netmod-routing-cfg";
            leaf description {
                 type string;
                 description
                       "Description of the RIB";
            }
            // Note that there is no list of interfaces within
            container policy {
                 description "Policy specific to RIB";
            }
           }
     }
     }

     grouping networking-services {
      description
```

```
          "Grouping for networking-services configured for
          a networking-instance.";
       container networking-services {
          description
                "Container for lst of networking services configured
                for a networking instance.";
          list networking-service {
                key "type";
                description
                 "List of networking services configured for a
                      networking instance.";
                leaf type {
                 type identityref {
                      base networking-service-type;
                 }
                 mandatory true;
                 description
                      "The networking services type supported within
                      a networking instance, e.g., NTP server, DNS
                      server, DHCP server, etc.";
                }
          }
       }
      }

      grouping oam-services {
      description "containers for features related to operations,
                  administration, and maintenance (OAM).";
      container oam-services {
       description "Commonly use OAM functions on devices";
       list oam-service {
          key "type";
          description
                "List of OAM services configured for a
                logical network element.";
          leaf type {
                type identityref {
                 base oam-service-type;
                }
                mandatory true;
                 description
                      "The OAM services type supported within
                      a logical networking element, e.g., SNMP.";
          }
       }
      }
      }

      grouping system-services {
```

```
     description "Containers for system service models.";
     uses oam-services;
     }

     grouping system-aaa {
     description "AAA-related services";
     container aaa {
      description
           "Authentication, Authorization, and Accounting (AAA).";
      list aaa-service {
           key "type";
           description
                 "List of AAA services configured for a
                 logical network element.";
           leaf type {
                 type identityref {
                  base aaa-service-type;
                 }
                 mandatory true;
                  description
                       "The AAA services type supported within
                       a logical networking element, e.g., RADIUS.";
           }
      }
     }
     }

     grouping system-management {
      description "System management for device or logical network
                 element";
      container system-management {
           description "System management for device or logical
                       network element";
           leaf device-view {
                 type boolean;
                 default "true";
                 description "Flag indicating whether or not the
logical
                             network element is able to view and manage
                             the entire device";
           }
           container system-management-global {
                 description "System management - logical device
                             management with reuse of RFC 7317";
           }
           list system-management-protocol {
                 key "type";
                 leaf type {
                  type identityref {
```

```
                    base system-management-protocol-type;
                }
                mandatory true;
                description
                    "NTP, DNS, Syslog, ssh, TACAC+, NETCONF, etc.";
               }
               description "List of system management protocol
                           configured for a logical networking
                           element.";
          }
      }
     }

     grouping ietf-acl {
      description "Packet Access Control Lists (ACLs) as specified
                   in draft-ietf-netmod-acl-model";
      container ietf-acl {
          description "ACLs and packet forwarding rules";
      }
     }

     grouping ietf-key-chain {
      description "Key chains as specified in
                   draft-acee-rtgwg-yang-key-chain;";
      container ietf-key-chain {
          description "Key chains";
      }
     }

     // top level device definition statements
     container info {
     description
      "Base system information.
      This container is for base system information, including
      device type (e.g., physical or virtual), model, serial no.,
      location, etc.";

     leaf device-type {
      //TODO: consider changing to an identity if finer grained
      // device type classification is envisioned
      type enumeration {
          enum PHYSICAL {
          description "physical or hardware device";
          }
          enum VIRTUAL {
          description "virtual or software device";
          }
      }
```

```
      description
           "Type of the device, e.g., physical or virtual.  This node
           may be used to activate other containers in the model";
     }
     }

     container hardware {
     description
      "Hardware / vendor -specific data relevant to the platform.
      This container is an anchor point for platform-specific
      configuration and operational state data.  It may be further
      organized into chassis, line cards, ports, etc.  It is
      expected that vendor or platform-specific augmentations
      would be used to populate this part of the device model";
     }

     container qos {
     description "QoS features, for example policing, shaping, etc.";
     }

     container logical-network-elements {
     description "Network devices may support multiple logical
                  network instances";

     list logical-network-element {
      key network-element-id;
      description "List of logical network elements";
      leaf network-element-id {
           type uint8; // expect a small number of logical routers
           description "Device-wide unique identifier for the
                        logical network element";
      }
      leaf network-element-name {
           type string;
           description "Descriptive name for the logical network
                        element";
      }
      leaf default-networking-instance-name {
           type string;
           description "Specification of the networking instance to
                        use for management connectivity";
      }
      uses system-management;
      uses ietf-acl;
      uses ietf-key-chain;
      container networking-instances {
           description "Networking instances each of which have
                        an independent IP/IPv6 addressing space
```

```
                    and protocol instantiations. For layer 3,
                    this consistent with the routing-instance
                    definition in ietf-routing";
          reference "draft-ietf-netmod-routing-cfg";
          list networking-instance {
                key networking-instance-name;
                description "List of networking-instances";
                leaf networking-instance-name {
                 type string;
                 description "logical network element scoped
                                 identifier for the networking
                                 instance";
                }
                leaf type {
                 type identityref {
                      base networking-instance;
                 }
                 description
                      "The networking instance type -- details TBD
                      Likely types include core, L3-VRF, VPLS,
                      L2-cross-connect, L2-VSI, etc.";
                }
                leaf enabled {
                 type boolean;
                 default "true";
                 description
                      "Flag indicating whether or not the networking
                      instance is enabled.";
                }
                uses router-id {
                 description
                      "Router ID for networking instances";
                }
                leaf description {
                 type string;
                 description
                      "Description of the networking instance
                      and its intended purpose";
                }
                // Note that there is no list of interfaces within
                // the networking-instance
                uses oam-protocols;
                uses networking-instance-policy;
                uses control-plane-protocols;
                uses ribs;
                uses mpls;
                uses networking-services;
          }
     }
```

```
     }
     }

     // augment statements
     augment "/if:interfaces/if:interface" {
     description
           "Add a node for the identification of the logical network
           element associated with an interface. Applies to interfaces
           that can be assigned on a per logical network element basis.
           A <TBD> error is returned when the interface type cannot be
           assigned.";

     leaf bind-network-element-id {
      type uint8;
      description
           "Logical network element ID to which interface is bound";
     }
     }

     augment "/if:interfaces/if:interface" {
     description
           "Add a node for the identification of the logical networking
           instance (which is within the interface's identified logical
           network element) associated with the IP information
           configured on an interface";

     leaf bind-networking-instance-name {
      type string;
      description
           "Networking Instance to which an interface is bound";
     }
     }

     augment "/if:interfaces/if:interface/ip:ipv4" {
     description
           "Add a node for the identification of the logical networking
           instance (which is within the interface's identified logical
           network element) associated with the IP information
           configured on an interface";

     leaf bind-networking-instance-name {
      type string;
      description
           "Networking Instance to which IPv4 interface is bound";

     }
     }
```

```
     augment "/if:interfaces/if:interface/ip:ipv6" {
     description
           "Add a node for the identification of the logical networking
           instance (which is within the interface's identified logical
           network element) associated with the IP information
           configured on an interface";

     leaf bind-networking-instance-name {
      type string;
      description
           "Networking Instance to which IPv6 interface is bound";

     }
     }

     // rpc statements

     // notification statements

   }
   <CODE ENDS>
```

7. References

7.1. Normative references

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2014.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, June 2014.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, August 2014.

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

7.2. Informative references

[NETMOD-MOUNT] Clemm, A., Medved, J., Voit, E., "Mounting YANG-Defined Information from Remote Datastores",

work in progress, draft-clemm-netmod-mount (work in progress).

[NETMOD-OPSTATE] Watsen, K., Nadeau, T., "NETMOD Operational State Requirements", work in progress, draft-chairs-netmod-opstate-reqs (work in progress).

[OC-MPLS] George, J., Fang, L., Osborne, E., Shakir, R., "MPLS TE Model for Service Provider Networks", draft-openconfig-mpls-consolidated-model (work in progress).

[OC-OPSTATE] Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate (work in progress).

[OC-STRUCT] Shaikh, A., Shakir, R., D'Souza, K., Fang, L., "Operational Structure and Organization of YANG Models", draft-openconfig-netmod-model-structure (work in progress).

[RFC4026] Andersson, L., Madsen, T., "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, March 2005.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, June 2014.

[RTG-CFG] Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg (work in progress).

[RTG-POLICY] Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-shaikh-rtgwg-policy-model (work in progress).

[RTG-OSPF] Yeung, D., Qu, Y., Zhang, J., and D. Bogdanovic, "Yang Data Model for OSPF Protocol", draft-yeung-netmod-ospf (work in progress).

[RTG-ISIS] Litkowski, S., Yeung, D., Lindem, A., Zhang, J., and L. Lhotka, "YANG Data Model for ISIS protocol", draft-ietf-isis-yang-isis-cfg (work in progress).

Appendix A. Acknowledgments

This document is derived from draft-openconfig-netmod-model-structure-00. The Authors of that document who are not also authors of this document are listed as Contributors to this work.

The original stated: The authors are grateful for valuable contributions to this document and the associated models from: Deepak Bansal, Paul Borman, Chris Chase, Josh George, Marcus Hines, and Jim Uttaro.

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Lou Berger, Qin Wu, Rob Shakir, Stephane Litkowski, and Yan Gang.

The leafref approach was proposed by Mahesh Jethanandani.

Contributors

Anees Shaikh
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US
Email: aashaikh@google.com

Rob Shakir
BT
pp. C3L, BT Centre
81, Newgate Street
London EC1A 7AJ
UK
Email: rob.shakir@bt.com
URI: http://www.bt.com/

Kevin D'Souza
AT&T
200 S. Laurel Ave
Middletown, NJ
US
Email: kd6913@att.com

Luyuan Fang
Microsoft
205 108th Ave. NE, Suite 400
Bellevue, WA
US
Email: lufang@microsoft.com

Qin Wu
Email: bill.wu@huawei.com

Stephane Litkowski
Email: stephane.litkowski@orange.com

Yan Gang
Email: yangang@huawei.com

Authors' Addresses

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA
Email: acee@cisco.com

Lou Berger (editor)
LabN Consulting, L.L.C.

Email: lberger@labn.net

Dean Bogdanovic
Email: ivandean@gmail.com

Christian Hopps
Deutsche Telekom
Email: chopps@chopps.org