Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

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

C. Hopps
Deustche Telekom

July 6, 2015

Network Device YANG Organizational Model
draft-rtgyangdt-rtgwg-device-model-00

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 rooted at a device, 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 device-level model. This document builds on that work and presents a derivative structure for use in representing the networking infrastructure aspects of physical and virtual devices.

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.

Additional motivation for this work can be found in [OC-STRUCT].

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 that document.

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. 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.

2. Model Overview

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

As shown below, our model is rooted at a "device", which represents a network router, switch, or similar network element. The model is applicable to both physical, hardware-based devices, as well as software-based devices such as virtual network functions (VNFs). It 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.

```
  +--rw device
       +--rw info
       |  +--rw device-type?   enumeration
       +--rw hardware
       +--rw interfaces
       |  +--rw interface* [name]
       |     ...
       +--rw qos
       +--rw logical-network-elements
       |     ...
```

The key subsystems are represented at the top level of the device, including, system-wide configuration, interfaces, and routing instances. The info section can be used for basic device information such as its type (e.g., physical or virtual), vendor, and model. For physical devices, the hardware container is intended to be a placeholder for platform-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.

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).

The interface model is taken from [RFC7223] and includes possible technology-specific augmentations using example technologies defined in [RFC7277]. Also included are examples of envisioned models in order to provide future guidance.

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 the networking instance (e.g., VRF or virtual switch instance) that is configured to use it.

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:

```
  +--rw 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 and augment [RFC7317] and is shown below:

```
  +--rw device
       +--rw logical-network-elements
             +--rw system-management
             |  +--rw device-view?                    boolean
```

```
               |  +--rw syslog
               |  +--rw dns
               |  +--rw ntp
               |  +--rw statistics-collection
               |  +--rw ssh
               |  +--rw tacacs
               |  +--rw snmp
               |  +--rw netconf
```

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.

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:

```
  +--rw 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. Examples shown below include Bi-directional Forwarding Detection (BFD), Ethernet Connectivity Fault Management (CFM), and Two-Way Active Measurement Protocol (TWAMP):

```
  +--rw device
       +--rw logical-network-elements
             +--rw networking-instances
                +--rw networking-instance* [networking-instance-name]
                   +--rw oam-protocols
                   |  +--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:

```
  +--rw 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. Note that 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].

```
     +--rw device
          +--rw logical-network-elements
               +--rw networking-instances
                  +--rw networking-instance* [networking-instance-name]
                     +--rw control-plane-protocols
                     |  +--rw bgp
                     |  |  +--rw policy
                     |  +--rw is-is
                     |  |  +--rw policy
                     |  +--rw ospf
                     |  |  +--rw policy
                     |  +--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:

```
     +--rw 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:

```
     +--rw 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. 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:

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

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

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 a 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 contain only 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 "model-structure.yang"
module model-structure {

  yang-version "1";

  // namespace
  namespace "urn:ietf:params:xml:ns:yang:model-structure";

  prefix "struct";

  // import some basic types


  // meta
  organization "OpenConfig working group/IETF RTG YANG Design Team";

  contact
    "OpenConfig working group netopenconfig@googlegroups.com
     Routing Area YANG Architecture DT 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 "2015-07-09" {
```

```
      description
        "First Pass of IETF Routing YANG Design Team";
      reference "draft-rtgyangdt-rtgwg-device-model-00.txt";
    }

    // extension statements

    // feature statements

    // identity statements

    // typedef statements

    // grouping statements

    grouping info {
      description
        "Base system information";

      container info {
        description
          "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";
        }

      }
    }

    grouping hardware {
      description
        "hardware / vendor -specific data relevant to the platform";
```

```
       container hardware {
         description
           "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";
       }
     }

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

     }

     grouping interfaces {
       description "interface-related models";
       container interfaces {
         description "Various interface models";
         reference "RFC 7223 - A Yang Model for Interface Management";
         list interface {
           key "name";
           description "List of interfaces keyed by name";
           leaf name {
               type string;
               description "Interface name";
           }
           leaf bind-network-element-id {
             type uint8;
             description "Logical network element ID to which
                          interface is bound";
           }
           container ethernet {
             description "Ethernet interface config, e.g., 10, 40,
                          100GB Ethernet";
             leaf bind-networking-instance-name {
                 type string;
                 description "Networking Instance to which
                              the Ethernet instance is bound";
             }
             container aggregates {
                 description
                     "LAGs, LACP, etc. for Ethernet interfaces";
                 reference "IEEE 802.1ad, 802.1AX";
             }
             container rstp {
                 description "Rapid Spanning Tree Protocol";
```

```
                reference "IEEE 802.1D-2004";
              }
              container lldp {
                  description "link layer discovery protocol";
                  reference "IEEE 802.1AB";
              }
              container ptp {
                  description
                      "Precision Time Protocol (PTP) for time
                       synchronization services. PTP typically requires
                       per-interface configuration";
                  reference "IEEE 1588-2008";
              }
            }
            container vlans {
                description "VLANs, 802.1q, q-in-q, etc.";
                reference "IEEE 802.1Q";
            }
            container tunnels {
                description
                    "Logical tunnel interfaces incl. GRE, VxLAN,
                      L2TP etc.";
            }

            container ipv4 {
              description "IPv4 interface";
              reference "RFC 7277 - A Yang Model for IP Management";
              leaf bind-networking-instance-name {
                  type string;
                  description "Networking Instance to which
                               IPv4 interface is bound";
              }
              container arp {
                description "Address resolution protocol";
                reference "STD 37 - An Ethernet Address Resolution
                           Protocol";
              }
             container icmp {
               description "Internet Control Message Protocol";
               reference "RFC 792 - Internet Control Message Protocol";
             }
             container vrrp {
               description "virtual router redundancy protocol";
               reference
                 "RFC 5798 - Virtual Router Redundancy Protocol
                  (VRRP) Version 3 for IPv4 and IPv6";
             }
             container dhcp-client {
               description "Dynamic Host Configuration Protocol";
```

```
              reference
                "RFC 2131 - Dynamic Host Configuration Protocol";
            }
           }
           container ipv6 {
             description "IPv6 interface";
               reference "RFC 7277 - A Yang Model for IP Management";
               leaf bind-networking-instance-name {
                   type string;
                   description "Networking Instance to which
                                IPv4 interface is bound";
               }
            container vrrp {
                description "virtual router redundancy protocol";
                reference
                    "RFC 5798 - Virtual Router Redundancy Protocol
                     (VRRP) Version 3 for IPv4 and IPv6";
            }
            container icmpv6 {
              description "Internet Control Message Protocol";
              reference
                "RFC 2463 - Internet Control Message Protocol
                 (ICMPv6) for Internet Protocol Version 6 (IPv6)";
            }
            container nd {
              description "IPv6 Neighbor Discovery";
                reference "RFC 4861 - Neighbor Discovery for IP
                           version 6 (IPv6)";
            }
           container dhcpv6-client {
             description
                 "Dynamic Host Configuration Protocol Version 6";
               reference
                   "RFC 3315 - Dynamic Host Configuration Protocol
                    for IPv6 (DHCPv6)";
            }
           }
       }
     }
   }

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

   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 {
      description
         "Definitions for OAM protocols within a networking-instance";
      container oam-protocols {
          description
              "OAM protocols";
        container bfd {
            description "Bi-directional Forwarding Detection (BFD)
                      configuration";

        }
        container cfm {
            description
               "Ethernet connectivity fault management.  Also includes
                options that are associated with specific interfaces,
                such as maintenance endpoint domains.";
           reference "IEEE 802.1ag";
        }
        container twamp {
            description
                "Two-way active measurement protocol for measuring
                round-trip IP layer performance.";
            reference "RFC 5357 A Two-Way Active Measurement Protocol
                      (TWAMP)";
        }
      }
     }

     grouping mpls {
       description "MPLS and TE configuration";
       container mpls {
         description "MPLS and TE configuration";
         container global {
             description "global MPLS configuration";
         }
        container label-switched-paths {
          description "Models for different types of LSPs";
        }
```

```
        container constrained-path {
          description "traffic-engineered or constrained path LSPs";
        }
        container igp-congruent {
          description "LSPs that follow the IGP-computed path";
        }
        container static {
           description "Statically configured LSPs";
        }
      }
    }

    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 "control protocol models";

      container control-plane-protocols {
        description "Control plane protocols and features";

        container bgp {
            description "BGP-4 protocol configuration";
            reference "RFC 4271 - Border Gateway Protocol 4 (BGP-4)";
            container policy {
                description "Policy specific to BGP";
            }
        }
        container is-is {
            description "ISO IS-IS protocol configuration";
            reference "RFC 1195 - Use of OSI IS-IS for Routing in
                       TCP/IP and Dual Environments";
            container policy {
                description "Policy specific to IS-IS";
            }
        }
```

```
        container ospf {
            description "Open Shortest Path First (OSPF) protocol
                         configuration";
            reference "RFC 2328 - OSPFv2 Protocol
                       RFC 5340 - OSPF for IPv6";
            container policy {
                description "Policy specific to OSPF";
            }
        }
        container rsvp {
            description "Resource Reservation Protocol (RSVP)
                         protocol configuration";
            reference "RFC 2205 - Resource ReSerVation Protocol (RSVP)
                       RFC 3209 - RSVP-TE: Extensions to RSVP for LSP
                       Tunnels";
        }
        container segment-routing {
            description "Segment Routing configuration for networking
                         instance";
            reference "draft-likowski-spring-sr-yang";
        }
        container ldp {
            description "Label Distribution Protocol (LDP)
                         configuration";
            reference "RFC 5036 - LDP Specification";
        }
        container pim {
            description "Protocol Independent Multicast (PIM)
                         configuration";
            reference "RFC 4601 - Protocol Independent Multicast -
                       Sparse Mode (PIM-SM) Protocol Specification";
        }
        container igmp {
            description "Internet Group Management Protocol
                         configuration";
            reference "RFC 3376 - Internet Group Management Protocol,
                       Version 3";
        }
        container mld {
            description "Multicast Listener Discovery Protocol
                         configuration";
            reference "RFC 3810 - Multicast Listener Discovery
                       Version 2 (MLDv2 for IPv6)";
        }
        container static-routes {
            description "Static route configuration";
            reference "draft-ietf-netmod-routing-cfg";
        }
      }
```

```
    }

    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
        "Networking services provided by the device in the scope of
         the networking-instance";
       container networking-services {
         description "Networking services";
         container ntp-server {
           description "Network Time Protocol (NTP) Server";
           reference "RFC 5905 - Network Time Protocol Version 4:
                      Protocol and Algorithms Specification";
```

```
          }
          container dns-server {
            description "Domain Name System (DNS) Server";
            reference "RFC 1035 - DOMAIN NAMES - IMPLEMENTATION
                       AND SPECIFICATION";
          }
          container dhcp-server {
            description "Dynamic Host Configuration Protocol (DHCP)
                         Server";
            reference "RFC 2131 - Dynamic Host Configuration Protocol";
          }
        }
      }

      grouping system-management {
          description "System management for device or logical network
                       element";
          container system-management {
              description "System management - logical device
                           management with reuse of RFC 7317";
              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 syslog {
                  description "Syslog configuration";
              }
              container dns {
                  description "Domain Name Service (DNS) and resolver
                               configuration";
              }
              container ntp {
                  description "network time protocol configuration";
              }
              container statistics-collection {
                  description
                      "Mechanisms for data collection from devices,
                       including packet and flow-level sampling";
              }
              container ssh {
                  description "SSH server configuration";
              }
              container tacacs {
                  description "TACACS+ configuration";
              }
              container snmp {
```

```
                description "System Network Management Protocol
                             (SNMP) configuration";
            }
            container netconf {
                description "Network Configuration Protocol(NETCONF)";
                reference "RFC 6020 - YANG - A Data Modeling Language
                           for the Network Configuration Protocol
                           (NETCONF)";
            }
        }
    }

    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";
        }
    }

    grouping qos {
      description "QoS features";

      container qos {
        description "QoS, including policing, shaping, etc.";
      }
    }

    // data definition statements

    container device {
      description "Top-level anchor point for models. Device is a
                   generic L2/L3 network element";
      uses info;
      uses hardware;
      uses interfaces;
      uses qos;
      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

    // 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

[OC-MPLS] George, J., Fang, L., Osborne, E., Shakir, R., "MPLS TE Model for Service Provider Networks", draft-openconfig-mpls-consolidated-model-00 (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-00 (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-19 (work in progress), October 2014.

[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-01 (work in progress), July 2015.

[RTG-OSPF] Yeung, D., Qu, Y., Zhang, J., and D. Bogdanovic, "Yang Data Model for OSPF Protocol", draft-yeung-netmod-ospf-02 (work in progress), October 2014.

[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-04 (work in progress), October 2014.

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.

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
Deustche Telekom
Email: chopps@chopps.org