Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

L. Berger
LabN Consulting, L.L.C.
C. Hopps
Deutsche Telekom
A. Lindem
Cisco Systems
D. Bogdanovic

March 13, 2017

YANG Network Instances
draft-ietf-rtgwg-ni-model-02

Abstract

This document defines a network instance module. This module along with the logical network element module can be used to manage the logical and virtual resource representations that may be present on a network device. Examples of common industry terms for logical resource representations are Logical Systems or Logical Routers. Examples of common industry terms for virtual resource representations are Virtual Routing and Forwarding (VRF) instances and Virtual Switch Instances (VSIs).

Table of Contents

1. Introduction

This document defines the second of two new modules that are defined to support the configuration and operation of network-devices that allow for the partitioning of resources from both, or either, management and networking perspectives. Both make use of the YANG functionality enabled by YANG Schema Mount [I-D.ietf-netmod-schema-mount].

Two forms of resource partitioning are supported:

The first form, which is defined in [I-D.ietf-rtgwg-lne-model], provides a logical partitioning of a network device where each partition is separately managed as essentially an independent network element which is 'hosted' by the base network device. These hosted network elements are referred to as logical network elements, or LNEs, and are supported by the logical-network-element module defined in [I-D.ietf-rtgwg-lne-model]. The module is used to identify LNEs and associate resources from the network-device with each LNE. LNEs themselves are represented in YANG as independent network devices; each accessed independently. Optionally, and when supported by the

implementation, they may also be accessed from the host system. Examples of vendor terminology for an LNE include logical system or logical router, and virtual switch, chassis, or fabric.

The second form, which is defined in this document, provides support what is commonly referred to as Virtual Routing and Forwarding (VRF) instances as well as Virtual Switch Instances (VSI), see [RFC4026]. In this form of resource partitioning multiple control plane and forwarding/bridging instances are provided by and managed via a single (physical or logical) network device. This form of resource partitioning is referred to as Network Instances and are supported by the network-instance module defined below. Configuration and operation of each network-instance is always via the network device and the network-instance module.

This document was motivated by, and derived from, [I-D.ietf-rtgwg-device-model].

1.1. Status of Work and Open Issues

The top open issues are:

1. This document will need to match the evolution and standardization of [I-D.openconfig-netmod-opstate] or [I-D.ietf-netmod-opstate-reqs] by the Netmod WG.

2. 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 logical 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 network instances (NIs). This breakdown is represented in Figure 1.

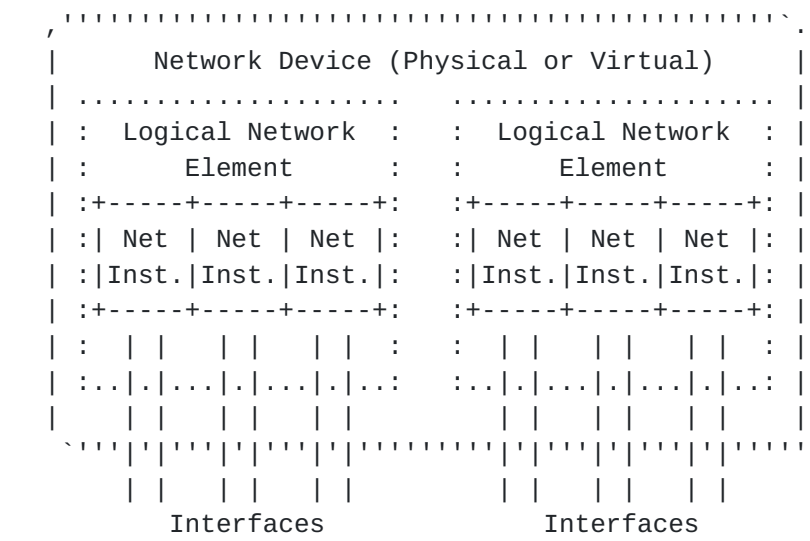

Figure 1: Module Element Relationships

A model for LNEs is described in [I-D.ietf-rtgwg-lne-model] and the model for network instances is covered in Section 3. For more information on how these models may be used within an overall device model structure, see [I-D.ietf-rtgwg-device-model].

The interface management model [RFC7223] is an existing model that is impacted by the definition of LNEs and network instances. This document and [I-D.ietf-rtgwg-lne-model] define augmentations to the interface module to support LNEs and NIs. Similar elements, although perhaps only for LNEs, may also need to be included as part of the definition of the future hardware and QoS modules.

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 management model is defined by [RFC7223].

The logical-network-element and network-instance modules augment the existing interface management model in two ways: The first, by the logical-network-element module, adds an identifier which is used on physical interface types to identify an associated LNE. The second, by the network-instance module, adds a name which is used on interface or sub-interface types to identify an associated network instance. Similarly, this name is also added for IPv4 and IPv6 types, as defined in [RFC7277].

The interface related augmentations are as follows:

```
module: ietf-logical-network-element
augment /if:interfaces/if:interface:
   +--rw bind-lne-name?   string

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

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

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

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-lne-name and bind-network-instance-name leaves provide the association between an interface and its associated LNE and NI (e.g., VRF or VSI).

3. 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 [RFC8022] and are shown below:

```
module: ietf-network-instance
   +--rw network-instances
      +--rw network-instance* [name]
         +--rw name                          string
         +--rw type?                         identityref
         +--rw enabled?                      boolean
         +--rw description?                  string
         +--rw network-instance-policy
         |  ...
         +--rw root?                      yang-schema-mount
         |  ...
augment /if:interfaces/if:interface:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv4:
   +--rw bind-network-instance-name?   string
augment /if:interfaces/if:interface/ip:ipv6:
   +--rw bind-network-instance-name?   string
```

A network instance is identified by a `name` string. This string is used both as an index within the network-instance module and to associate resources with a network instance as shown above in the interface augmentation. Type is used to indicate the type NI, such as L3-VRF, VPLS, L2-VSI, etc. Network instance policy and root are discussed in greater detail below.

3.1. Network Instance Policy

Network instance policies are used to control how NI information is represented at the device level, VRF routing policies, and VRF/VSI identifiers. Examples include BGP route targets (RTs) and route distinguishers (RDs), virtual network identifiers (VN-IDs), VPLS neighbors, etc. The structure is expected to be:

```
module: ietf-network-instance
   +--rw network-instances
      +--rw network-instance* [name]
         +--rw network-instance-policy
            (TBD)
```

3.2. Network Instance Management

Modules that may be used to represent network instance specific information will be available under `root`. As with LNEs, actual module availability is expected to be implementation dependent. The use-schema mechanism defined as part of the Schema Mount module [I-D.ietf-netmod-schema-mount] is expected to be the primary method used to identify supported modules. Resource related control and assignment is expected to be managed at the network-device level, not the network instance level, based on the `bind-network-instance-name` augmentation mentioned above. Mounted modules will access such information, as well as any other information contained within a module at the device root, by using the parent-reference mechanism defined in [I-D.ietf-netmod-schema-mount].

As an example, consider the case where a network instance with a `name` of "green" is defined on a network device. In this case the following logical structure might be made available:

```
   +--rw yanglib:modules-state           [RFC7895]
   +--rw if:interfaces                   [RFC7223]
   |  +--rw bind-network-instance-name="green" string
   +--rw network-instances
      +--rw network-instance* [name]
         +--rw name="green"    string
         +--rw type?                            identityref
         +--rw enabled=true                     boolean
         +--rw description="The Green VRF"      string
         +--rw network-instance-policy
         |  ... (RT=1000:1, RD=1.2.3.4)
         +--rw root?                            yang-schema-mount
```

with a corresponding logical structure in the schema-mount module:

```
module: ietf-yang-schema-mount
   +--ro schema-mounts
      :
      +--ro mount-point* [module name]
      |  +--ro module="ietf-network-instance"
      |  +--ro name="root"
      |  +--ro config=true
      |  +--ro (schema-ref)?
      |     +--:(use-schema)
      |        +--ro use-schema* [name]
      |           +--ro name="ni-vrf"
      :           :
      :
      +--ro schema* [name]
         +--ro name="ni-vrf"           string
         +--ro module*  [name revision]
         |  +--ro name="mm:network-services"
         :  :
         |  +--ro name="nn:oam-protocols"
         :  :
         |  +--ro name="oo:routing"
         :  :
         |  +--ro name="pp:mpls"
         :  :
         +--ro mount-point* [network-instance]
            :
```

All modules that represent control-plane and data-plane information may be present at the `root`, and be accessible via paths modified per [I-D.ietf-netmod-schema-mount]. The list of available modules is expected to be implementation dependent. As is the method used by an implementation to support NIs.

3.3. Network Instance Instantiation

Network instances may be controlled by clients using existing list operations. When list entries are created, a new instance is instantiated. The models mounted under a NI root is expected to be dependent on the server implementation. When a list entry is deleted, an existing network instance is destroyed. For more information see [RFC7950] Section 7.8.6.

4. Security Considerations

TBD

5. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-network-instance

Registrant Contact: The IESG.

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

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

```
name:         ietf-network-instance
namespace:    urn:ietf:params:xml:ns:yang:ietf-network-instance
prefix:       ni
reference:    RFC XXXX
```

6. Network Instance Model

The structure of the model defined in this document is described by the YANG module below.

```
<CODE BEGINS> file "ietf-network-instance@2017-03-13.yang"
module ietf-network-instance {

  yang-version 1.1;

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

  prefix ni;

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

  import ietf-ip {
    prefix ip;
  }

  import ietf-yang-schema-mount {
    prefix yangmnt;
  }
```

```
   // meta
     organization "IETF Routing Area Working Group (rtgwg)";

     contact
         "Routing Area Working Group - <rtgwg@ietf.org>";

     description
       "This module is used to support multiple network instances
        within a single physical or virtual device.  Network
        instances are commonly know as VRFs (virtual routing
        and forwarding) and VSIs (virtual switching instances).";

     revision "2017-03-13" {
       description
         "Initial revision.";
       reference "RFC TBD";
     }

     // extension statements

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

     // identity statements

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

      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";
      }

     // typedef statements

     // grouping statements
```

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

     }

     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 network-instance-policy {
      description
          "Network 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 network-instance-policy {
          description
            "Network Instance Policy -- details TBD,
            perhaps based on BESS model";
      }
    }

    // top level device definition statements
    container network-instances {
        description "Network 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
            "RFC 8022 - A YANG Data Model for Routing Management";
        list network-instance {
            key name;
            description "List of network-instances";
            leaf name {
                type string;
                description "device scoped
                             identifier for the network
                             instance";
            }
            leaf type {
                type identityref {
                    base network-instance-type;
                }
                description
```

```
                  "The network 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 network
                 instance is enabled.";
          }
          leaf description {
              type string;
              description
                "Description of the network instance
                and its intended purpose";
          }

          uses network-instance-policy;

          yangmnt:mount-point root {
              description
                "Root for models supported per
                 network instance.  This will
                 typically not be an inline type
                 mount point.";
          }
      }
    }

    // augment statements

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

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

    augment "/if:interfaces/if:interface/ip:ipv4" {
      description
```

```
          "Add a node for the identification of the logical
          network instance (which is within the interface's
          identified physical or virtual device) associated with
          the IP information configured on an interface";

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

      }
    }

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

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

      }
    }

    // rpc statements

    // notification statements

  }
  <CODE ENDS>
```

7. References

7.1. Normative References

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-04 (work in progress), March 2017.

[I-D.ietf-rtgwg-lne-model]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "YANG Logical Network Elements", draft-ietf-rtgwg-lne-model-01 (work in progress), October 2016.

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

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

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

[RFC7277] Bjorklund, M., "A YANG Data Model for IP Management", RFC 7277, DOI 10.17487/RFC7277, June 2014, <http://www.rfc-editor.org/info/rfc7277>.

7.2. Informative References

[I-D.ietf-netmod-opstate-reqs]
Watsen, K. and T. Nadeau, "Terminology and Requirements for Enhanced Handling of Operational State", draft-ietf-netmod-opstate-reqs-04 (work in progress), January 2016.

[I-D.ietf-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Organizational Models", draft-ietf-rtgwg-device-model-01 (work in progress), October 2016.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <http://www.rfc-editor.org/info/rfc4026>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <http://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <http://www.rfc-editor.org/info/rfc7950>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <http://www.rfc-editor.org/info/rfc8022>.

Appendix A. Acknowledgments

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 RFC text was produced using Marshall Rose's xml2rfc tool.

Appendix B. Contributors

Contributors' Addresses

TBD

Authors' Addresses

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Christan Hopps
Deutsche Telekom

Email: chopps@chopps.org

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Dean Bogdanovic

Email: ivandean@gmail.com