

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 25, 2016

L. Berger
LabN Consulting, L.L.C.
C. Hopps
Deutsche Telekom
A. Lindem
Cisco Systems
D. Bogdanovic
June 23, 2016

Network Instance Model
draft-ietf-rtgwg-ni-model-00

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).

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 December 25, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

- [1. Introduction](#) [2](#)
- [1.1. Status of Work and Open Issues](#) [3](#)
- [2. Overview](#) [3](#)
- [3. Network Instances](#) [6](#)
- [3.1. Network Instance Policy](#) [6](#)
- [3.2. Network Instance Management](#) [7](#)
- [3.3. Network Instance Instantiation](#) [8](#)
- [4. Security Considerations](#) [8](#)
- [5. IANA Considerations](#) [8](#)
- [6. Network Instance Model](#) [8](#)
- [7. References](#) [13](#)
- [7.1. Normative References](#) [13](#)
- [7.2. Informative References](#) [14](#)
- [Appendix A. Acknowledgments](#) [14](#)
- [Appendix B. Contributors](#) [15](#)
- [Authors' Addresses](#) [15](#)

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 emerging YANG functionality supported by YANG Schema Mount [[I-D.ietf-netmod-schema-mount](#)]. This document is expected to use whatever Schema Mount solution is agreed upon by the Netmod Working Group.

Two forms of resource partitioning are supported:

The first form, which is defined in [[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 [[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, [[RTG-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.

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 [I-D.ietf-netmod-routing-cfg] 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?               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 yang library module [[I-D.ietf-netconf-yang-library](#)] 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.

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 structure might be made available:

```

+-rw yanglib:modules-state          [I-D.ietf-netconf-yang-library]
+-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?              schema-mount
      +-rw yanglib:modules-state [I-D.ietf-netconf-yang-library]
      +-rw if:intefaces          [RFC7223]
      +-rw mm:network-services
      +-rw nn:oam-protocols
      +-rw oo:routing
      +-rw pp:mpls

```

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

TBD -- need to resolve if instantiation is based on new list entry creation per the pending Schema Mount solution definition.

4. Security Considerations

LNE portion is TBD

NI portion is TBD

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. 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@2016-06-23.yang"
module ietf-network-instance {

    yang-version "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;
    }

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

    contact
        "Routing Area Working Group - <rtwgw@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 "2016-06-23" {
  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";
    }
  }

  // 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 "draft-ietf-netmod-routing-cfg";
    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;
    leaf root {
        type schema-mount;
        description "Root for models supported per
            network instance";
    }
}
}

// 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-01](#) (work in progress), April 2016.
- [LNE-MODEL]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "Logical Network Element Model", [draft-ietf-rtgwg-lne-model-00.txt](#) (work in progress), June 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-netconf-yang-library]

Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", [draft-ietf-netconf-yang-library-06](#) (work in progress), April 2016.

[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-netmod-routing-cfg]

Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", [draft-ietf-netmod-routing-cfg-21](#) (work in progress), March 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>>.

[RTG-DEVICE-MODEL]

Lindem, A., Ed., Berger, L., Ed., Bogdanovic, D., and C. Hopps, "Network Device YANG Organizational Models", [draft-rtgyangdt-rtgwg-device-model-04.txt](#) (work in progress), May 2016.

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

