Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 20, 2018

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

X. Liu
Jabil
January 16, 2018

YANG Network Instances
draft-ietf-rtgwg-ni-model-06

Abstract

This document defines a network instance module. This module can be used to manage the virtual resource partitioning that may be present on a network device. Examples of common industry terms for virtual resource partitioning 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 leverage the YANG functionality enabled by YANG Schema Mount [I-D.ietf-netmod-schema-mount].

The first form of resource partitioning 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]. That 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. 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 for what is commonly referred to as Virtual Routing and Forwarding (VRF) instances as well as Virtual Switch Instances (VSI), see [RFC4026] and [RFC4664]. 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 a Network Instance and is 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.

One notable difference between the LNE model and the NI model is that the NI model provides a framework for VRF and VSI management. This document envisions the separate definition of VRF and VSI, i.e., L3 and L2 VPN, technology specific models. An example of such can be found in the emerging L3VPN model defined in [I-D.ietf-bess-l3vpn-yang] and the examples discussed below.

1.1. Terminology

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

Readers are expected to be familiar with terms and concepts of YANG [RFC7950] and YANG Schema Mount [I-D.ietf-netmod-schema-mount].

This document uses the graphical representation of data models defined in [I-D.ietf-netmod-yang-tree-diagrams].

1.2. Status of Work and Open Issues

The top open issues are:

1. Schema mount currently doesn't allow parent-reference filtering on the instance of the mount point, but rather just the schema. This means it is not possible to filter based on actual data, e.g., bind-network-instance-name="green". In the schema mount definition, the text and examples should be updated to cover this case.

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.

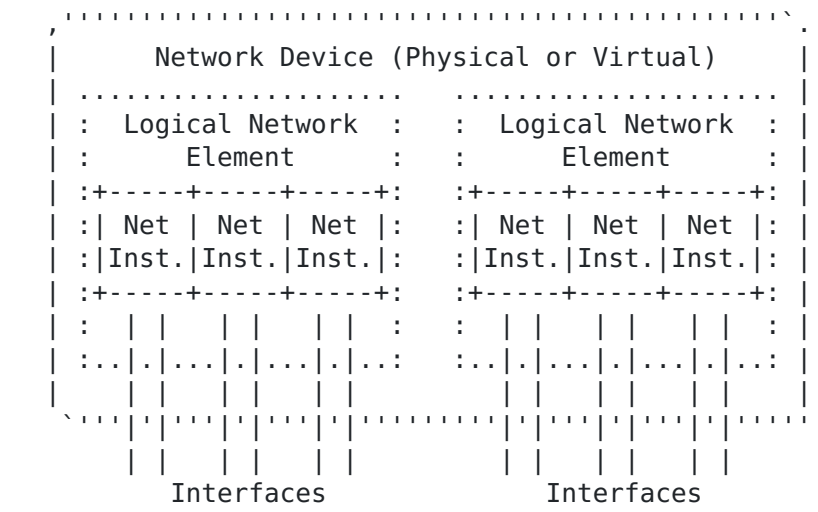

Figure 1: Module Element Relationships

A model for LNEs is described in [I-D.ietf-rtgwg-lne-model] and the model for NIs is covered in this document in Section 3.

The current interface management model [RFC7223] is impacted by the definition of LNEs and NIs. This document and [I-D.ietf-rtgwg-lne-model] define augmentations to the interface module to support LNEs and NIs.

The network instance model supports the configuration of VRFs and VSIs. Each instance is supported by information that relates to the device, for example the route target used when advertising VRF routes via the mechanisms defined in [RFC4364], and information that relates to the internal operation of the NI, for example for routing

protocols [RFC8022] and OSPF [I-D.ietf-ospf-yang]. This document defines the network-instance module that provides a basis for the management of both types of information.

NI information that relates to the device, including the assignment of interfaces to NIs, is defined as part of this document. The defined module also provides a placeholder for the definition of NI-technology specific information both at the device level and for NI internal operation. Information related to NI internal operation is supported via schema mount [I-D.ietf-netmod-schema-mount] and mounting appropriate modules under the mount point. Well known mount points are defined for L3VPN, L2VPN, and L2+L3VPN NI types.

3. Network Instances

The network instance container is used to represent virtual routing and forwarding instances (VRFs) and virtual switching instances (VSIs). 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 supports both core/provider and virtual instances. Core/provider instance information is accessible at the top level of the server, while virtual instance information is accessible under the root schema mount points.

The NI model can be represented using the tree format defined in [I-D.ietf-netmod-yang-tree-diagrams] as:

```
module: ietf-network-instance
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name           string
        +--rw enabled?       boolean
        +--rw description?   string
        +--rw (ni-type)?
        +--rw (root-type)
           +--:(vrf-root)
           |  +--mp vrf-root
           +--:(vsi-root)
           |  +--mp vsi-root
           +--:(vv-root)
              +--mp vv-root
augment /if:interfaces/if:interface:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name
augment /if:interfaces/if:interface/ip:ipv4:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name
augment /if:interfaces/if:interface/ip:ipv6:
  +--rw bind-ni-name?   -> /network-instances/network-instance/name

notifications:
  +---n bind-ni-name-failed
     +--ro name          -> /if:interfaces/interface/name
     +--ro interface
     |  +--ro bind-ni-name?
     |                  -> /if:interfaces/interface/ni:bind-ni-name
     +--ro ipv4
     |  +--ro bind-ni-name?
     |          -> /if:interfaces/interface/ip:ipv4/ni:bind-ni-name
     +--ro ipv6
     |  +--ro bind-ni-name?
     |          -> /if:interfaces/interface/ip:ipv6/ni:bind-ni-name
     +--ro error-info?   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. The ni-type and root-type choice statements are used to support different types of L2 and L3 VPN technologies. The bind-ni-name-failed notification is used in certain failure cases.

3.1. NI Types and Mount Points

The network-instance module is structured to facilitate the definition of information models for specific types of VRFs and VSIs using augmentations. For example, the information needed to support

VPLS, VxLAN and EVPN based L2VPNs are likely to be quite different. Example models under development that could be restructured to take advantage on NIs include, for L3VPNs [I-D.ietf-bess-l3vpn-yang] and for L2VPNs [I-D.ietf-bess-l2vpn-yang].

Documents defining new YANG models for the support of specific types of network instances should augment the network instance module. The basic structure that should be used for such augmentations include a case statement, with containers for configuration and state data and finally, when needed, a type specific mount point. Generally ni types, are expected to not need to define type specific mount points, but rather reuse one of the well known mount point, as defined in the next section. The following is an example type specific augmentation:

```
 augment "/ni:network-instances/ni:network-instance/ni:ni-type" {
   case l3vpn {
     container l3vpn {
         ...
     }
     container l3vpn-state {
         ...
     }
   }
 }
```

3.1.1. Well Known Mount Points

YANG Schema Mount, [I-D.ietf-netmod-schema-mount], identifies mount points by name within a module. This definition allows for the definition of mount points whose schema can be shared across ni-types. As discussed above, ni-types largely differ in the configuration information needed in the core/top level instance to support the NI, rather than in the information represented within an NI. This allows the use of shared mount points across certain NI types.

The expectation is that there are actually very few different schema that need to be defined to support NIs on an implementation. In particular, it is expected that the following three forms of NI schema are needed, and each can be defined with a well known mount point that can be reused by future modules defining ni-types.

The three well known mount points are:

vrf-root
: vrf-root is intended for use with L3VPN type ni-types.

vsi-root
vsi-root is intended for use with L2VPN type ni-types.

vv-root
vv-root is intended for use with ni-types that simultaneously support L2VPN bridging and L3VPN routing capabilities.

Future model definitions should use the above mount points whenever possible. When a well known mount point isn't appropriate, a model may define a type specific mount point via augmentation.

3.1.2. NI Type Example

The following is an example of an L3VPN VRF using a hypothetical augmentation to the networking instance schema defined in [I-D.ietf-bess-l3vpn-yang]. More detailed examples can be found in Appendix B.

```
module: ietf-network-instance
  +--rw network-instances
     +--rw network-instance* [name]
        +--rw name           string
        +--rw enabled?       boolean
        +--rw description?   string
        +--rw (ni-type)?
        |  +--:(l3vpn)
        |     +--rw l3vpn:l3vpn
        |     |  ... // config data
        |     +--ro l3vpn:l3vpn-state
        |     |  ... // state data
        +--rw (root-type)
           +--:(vrf-root)
              +--mp vrf-root
                 +--ro rt:routing-state/
                 |  +--ro router-id?                 yang:dotted-quad
                 |  +--ro control-plane-protocols
                 |     +--ro control-plane-protocol* [type name]
                 |        +--ro ospf:ospf/
                 |           +--ro instance* [af]
                 +--rw rt:routing/
                 |  +--rw router-id?                 yang:dotted-quad
                 |  +--rw control-plane-protocols
                 |     +--rw control-plane-protocol* [type name]
                 |     +--rw ospf:ospf/
                 |        +--rw instance* [af]
                 |           +--rw areas
                 |              +--rw area* [area-id]
                 |                 +--rw interfaces
                 |                    +--rw interface* [name]
                 |                       +--rw name if:interface-ref
                 |                       +--rw cost?   uint16
                 +--ro if:interfaces@
                 |  ...
                 +--ro if:interfaces-state@
                 |  ...
```

This shows YANG Routing Management [RFC8022] and YANG OSPF [I-D.ietf-ospf-yang] as mounted modules. The mounted modules can reference interface information via a parent-reference to the containers defined in [RFC7223].

3.2. NIs and Interfaces

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].

As shown below, the network-instance module augments the existing interface management model by adding 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 following is an example of envisioned usage. The interfaces container includes a number of commonly used components as examples:

```
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name                        string
   |     +--rw ip:ipv4!
   |     |  +--rw ip:enabled?                      boolean
   |     |  +--rw ip:forwarding?                   boolean
   |     |  +--rw ip:mtu?                          uint16
   |     |  +--rw ip:address* [ip]
   |     |  |  +--rw ip:ip               inet:ipv4-address-no-zone
   |     |  |  +--rw (ip:subnet)
   |     |  |     +--:(ip:prefix-length)
   |     |  |     |  +--rw ip:prefix-length?   uint8
   |     |  |     +--:(ip:netmask)
   |     |  |        +--rw ip:netmask?         yang:dotted-quad
   |     |  +--rw ip:neighbor* [ip]
   |     |  |  +--rw ip:ip                  inet:ipv4-address-no-zone
   |     |  |  +--rw ip:link-layer-address yang:phys-address
   |     |  +--rw ni:bind-network-instance-name?   string
   |     +--rw ni:bind-network-instance-name?   string
```

The [RFC7223] defined interface model is structured to include all interfaces in a flat list, without regard to virtual instances (e.g., VRFs) supported on the device. The bind-network-instance-name leaf provides the association between an interface and its associated NI (e.g., VRF or VSI). Note that as currently defined, to assign an interface to both an LNE and NI, the interface would first be assigned to the LNE using the mechanisms defined in [I-D.ietf-rtgwg-lne-model] and then within that LNE's interface module, the LNE's representation of that interface would be assigned to an NI.

3.3. Network Instance Management

Modules that may be used to represent network instance information will be available under the ni-type specific 'root' mount point. The use-schema mechanism defined as part of the Schema Mount module [I-D.ietf-netmod-schema-mount] MUST be used with the module defined in this document to identify accessible modules. A future version of this document could relax this requirement. Mounted modules in the non-inline case SHOULD be defined with access, via the appropriate schema mount parent-references [I-D.ietf-netmod-schema-mount], to device resources such as interfaces. An implementation MAY choose to restrict parent referenced information to information related to a specific instance, e.g., only allowing references to interfaces that have a "bind-network-instance-name" which is identical to the instance's "name".

All modules that represent control-plane and data-plane information may be present at the 'root' mount point, 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.

For example, the following could be used to define the data organization of the example NI shown in Section 3.1.2:

```
    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-network-instance",
          "label": "vrf-root",
          "use-schema": [
            {
              "name": "ni-schema",
              "parent-reference": [
                "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
              ]
            }
          ]
        }
      ],
      "schema": [
        {
          "name": "ni-schema",
          "module": [
            {
               "name": "ietf-routing",
               "revision": "2016-11-04",
               "namespace":
                 "urn:ietf:params:xml:ns:yang:ietf-routing",
               "conformance-type": "implement"
             },
            {
               "name": "ietf-ospf",
               "revision": "2017-03-12",
               "namespace":
                 "urn:ietf:params:xml:ns:yang:ietf-ospf",
               "conformance-type": "implement"
            }
          ]
        }
      ]
    }
```

Module data identified under "schema" will be instantiated under the mount point identified under "mount-point". These modules will be able to reference information for nodes belonging to top-level modules that are identified under "parent-reference". Parent referenced information is available to clients via their top level paths only, and not under the associated mount point.

To allow a client to understand the previously mentioned instance restrictions on parent referenced information, an implementation MAY

represent such restrictions in the "parent-reference" leaf-list. For example:

```
    "namespace": [
      {
        "prefix": "if",
        "uri": "urn:ietf:params:xml:ns:yang:ietf-interfaces"
      },
      {
        "prefix": "ni",
        "uri": "urn:ietf:params:xml:ns:yang:ietf-network-instance"
      }
    ],
    "mount-point": [
      {
        "parent-reference": [
          "/if:interfaces/if:interface
             [ni:bind-network-instance-name = current()/../ni:name]",
          "/if:interfaces-state/if:interface
             [if:name = /if:interfaces/if:interface
               [ni:bind-ni-name = current()/../ni:name]/if:name]",
          "/if:interfaces/if:interface/ip:ipv4
             [ni:bind-network-instance-name = current()/../ni:name]",
          "/if:interfaces-state/if:interface/ip:ipv4
             [if:name = /if:interfaces/if:interface/ip:ipv4
               [ni:bind-ni-name = current()/../ni:name]/if:name]",
          "/if:interfaces/if:interface/ip:ipv6
             [ni:bind-network-instance-name = current()/../ni:name]",
          "/if:interfaces-state/if:interface/ip:ipv6
             [if:name = /if:interfaces/if:interface/ip:ipv4
               [ni:bind-ni-name = current()/../ni:name]/if:name]",
        ]
      }
    ],
```

3.4. Network Instance Instantiation

Network instances may be controlled by clients using existing list operations. When a list entry is created, a new instance is instantiated. The models mounted under an NI root are 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.

Once instantiated, host network device resources can be associated with the new NI. As previously mentioned, this document augments ietf-interfaces with the bind-ni-name leaf to support such associations for interfaces. When a bind-ni-name is set to a valid

NI name, an implementation MUST take whatever steps are internally necessary to assign the interface to the NI or provide an error message (defined below) with an indication of why the assignment failed. It is possible for the assignment to fail while processing the set operation, or after asynchronous processing. Error notification in the latter case is supported via a notification.

4. Security Considerations

There are two different sets of security considerations to consider in the context of this document. One set is security related to information contained within mounted modules. The security considerations for mounted modules are not substantively changed based on the information being accessible within the context of an NI. For example, when considering the modules defined in [RFC8022], the security considerations identified in that document are equally applicable, whether those modules are accessed at a server's root or under an NI instance's root node.

The second area for consideration is information contained in the NI module itself. NI information represents network configuration and route distribution policy information. As such, the security of this information is important, but it is fundamentally no different than any other interface or routing configuration information that has already been covered in [RFC7223] and [RFC8022].

The vulnerable "config true" parameters and subtrees are the following:

/network-instances/network-instance: This list specifies the network instances and the related control plane protocols configured on a device.

/if:interfaces/if:interface/*/bind-network-instance-name: This leaf indicates the NI instance to which an interface is assigned.

Unauthorized access to any of these lists can adversely affect the routing subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations and other problems.

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@2017-12-04.yang"
module ietf-network-instance {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-instance";
  prefix ni;

  // import some basic types

  import ietf-interfaces {
    prefix if;
    reference "RFC 7223: A YANG Data Model for Interface
               Management";
  }
  import ietf-ip {
    prefix ip;
    reference "RFC 7277: A YANG Data Model for IP Management";
  }
  import ietf-yang-schema-mount {
    prefix yangmnt;
    reference "draft-ietf-netmod-schema-mount: YANG Schema Mount";
    // RFC Ed.: Please replace this draft name with the
    // corresponding RFC number
  }

  organization
    "IETF Routing Area (rtgwg) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>
```

```
        Author:   Lou Berger
                  <mailto:lberger@labn.net>
        Author:   Christan Hopps
                  <mailto:chopps@chopps.org>
        Author:   Acee Lindem
                  <mailto:acee@cisco.com>
        Author:   Dean Bogdanovic
                  <mailto:ivandean@gmail.com>";
    description
      "This module is used to support multiple network instances
       within a single physical or virtual device.  Network
       instances are commonly known as VRFs (virtual routing
       and forwarding) and VSIs (virtual switching instances).

       Copyright (c) 2017 IETF Trust and the persons
       identified as authors of the code.  All rights reserved.

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

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

    // RFC Ed.: replace XXXX with actual RFC number and remove
    // this note
    // RFC Ed.: please update TBD

    revision 2017-12-04 {
      description
        "Initial revision.";
      reference "RFC TBD";
    }

    // top level device definition statements

    container network-instances {
      description
        "Network instances each of which consists of a
         VRFs (virtual routing and forwarding) and/or
         VSIs (virtual switching instances).";
      reference "RFC 8022 - A YANG Data Model for Routing
                 Management";
      list network-instance {
        key "name";
```

```
          description
            "List of network-instances.";
          leaf name {
            type string;
            mandatory true;
            description
              "device scoped identifier for the network
               instance.";
          }
          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.";
          }
          choice ni-type {
            description
              "This node serves as an anchor point for different types
               of network instances.  Each 'case' is expected to
               differ in terms of the information needed in the
               parent/core to support the NI, and may differ in their
               mounted schema definition. When the mounted schema is
               not expected to be the same for a specific type of NI
               a mount point should be defined.";
          }
          choice root-type {
            mandatory true;
            description
              "Well known mount points.";
            container vrf-root {
              description
                "Container for mount point.";
              yangmnt:mount-point "vrf-root" {
                description
                  "Root for L3VPN type models. This will typically
                   not be an inline type mount point.";
              }
            }
            container vsi-root {
              description
                "Container for mount point.";
```

```
            yangmnt:mount-point "vsi-root" {
              description
                "Root for L2VPN type models. This will typically
                 not be an inline type mount point.";
            }
          }
          container vv-root {
            description
              "Container for mount point.";
            yangmnt:mount-point "vv-root" {
              description
                "Root models that support both L2VPN type bridging
                 and L3VPN type routing. 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 network
         instance associated with the information configured
         on a interface.

         Note that a standard error will be returned if the
         identified leafref isn't present.  If an interfaces cannot
         be assigned for any other reason, the operation SHALL fail
         with an error-tag of 'operation-failed' and an
         error-app-tag of 'ni-assignment-failed'.  A meaningful
         error-info that indicates the source of the assignment
         failure SHOULD also be provided.";
      leaf bind-ni-name {
        type leafref {
          path "/network-instances/network-instance/name";
        }
        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 network
         instance associated with the information configured
         on an IPv4 interface.
```

```
          Note that a standard error will be returned if the
          identified leafref isn't present.  If an interfaces cannot
          be assigned for any other reason, the operation SHALL fail
          with an error-tag of 'operation-failed' and an
          error-app-tag of 'ni-assignment-failed'.  A meaningful
          error-info that indicates the source of the assignment
          failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
          }
          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 network
           instance associated with the information configured
           on an IPv6 interface.

           Note that a standard error will be returned if the
           identified leafref isn't present.  If an interfaces cannot
           be assigned for any other reason, the operation SHALL fail
           with an error-tag of 'operation-failed' and an
           error-app-tag of 'ni-assignment-failed'.  A meaningful
           error-info that indicates the source of the assignment
           failure SHOULD also be provided.";
        leaf bind-ni-name {
          type leafref {
            path "/network-instances/network-instance/name";
          }
          description
            "Network Instance to which IPv6 interface is bound.";
        }
      }

      // notification statements

      notification bind-ni-name-failed {
        description
          "Indicates an error in the association of an interface to an
           NI. Only generated after success is initially returned when
           bind-ni-name is set.

           Note: some errors may need to be reported for multiple
           associations, e.g., a single error may need to be reported
           for an IPv4 and an IPv6 bind-ni-name.
```

```
          At least one container with a bind-ni-name leaf MUST be
          included in this notification.";
        leaf name {
          type leafref {
            path "/if:interfaces/if:interface/if:name";
          }
          mandatory true;
          description
            "Contains the interface name associated with the
            failure.";
        }
        container interface {
          description
            "Generic interface type.";
          leaf bind-ni-name {
            type leafref {
              path "/if:interfaces/if:interface/ni:bind-ni-name";
            }
            description
              "Contains the bind-ni-name associated with the
               failure.";
          }
        }
        container ipv4 {
          description
            "IPv4 interface type.";
          leaf bind-ni-name {
            type leafref {
              path "/if:interfaces/if:interface"
                + "/ip:ipv4/ni:bind-ni-name";
            }
            description
              "Contains the bind-ni-name associated with the
               failure.";
          }
        }
        container ipv6 {
          description
            "IPv6 interface type.";
          leaf bind-ni-name {
            type leafref {
              path "/if:interfaces/if:interface"
                + "/ip:ipv6/ni:bind-ni-name";
            }
            description
              "Contains the bind-ni-name associated with the
               failure.";
          }
```

```
      }
      leaf error-info {
        type string;
        description
          "Optionally, indicates the source of the assignment
           failure.";
      }
    }
  }
<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-08 (work in progress), October 2017.

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-04 (work in progress), December 2017.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://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, <https://www.rfc-editor.org/info/rfc6020>.

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

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

7.2. Informative References

[I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-07 (work in progress), October 2017.

[I-D.ietf-bess-l3vpn-yang]
Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS L3 VPNs", draft-ietf-bess-l3vpn-yang-02 (work in progress), October 2017.

[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-09 (work in progress), October 2017.

[I-D.ietf-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Logical Organization", draft-ietf-rtgwg-device-model-02 (work in progress), March 2017.

[I-D.ietf-rtgwg-lne-model]
Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Logical Network Elements", draft-ietf-rtgwg-lne-model-05 (work in progress), December 2017.

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

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <https://www.rfc-editor.org/info/rfc4664>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://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, <https://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. Useful review comments were also received by Martin Bjorklund and John Scudder.

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

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

Appendix B. Example NI usage

The following subsections provide example uses of NIs.

B.1. Configuration Data

The following shows an example where two customer specific network instances are configured:

```
{
  "ietf-network-instance:network-instances": {
    "network-instance": [
      {
        "name": "vrf-red",
        "vrf-root": {
          "ietf-routing:routing": {
            "router-id": "192.0.2.1",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "instance": [
                      {
                        "af": "ipv4",
                        "areas": {
                          "area": [
                            {
                              "area-id": "203.0.113.1",
                              "interfaces": {
                                "interface": [
```

```
                                {
                                  "name": "eth1",
                                  "cost": 10
                                }
                              ]
                            }
                          }
                        ]
                      }
                    }
                  ]
                }
              }
            }
          },
          {
            "name": "vrf-blue",
            "vrf-root": {
              "ietf-routing:routing": {
                "router-id": "192.0.2.2",
                "control-plane-protocols": {
                  "control-plane-protocol": [
                    {
                      "type": "ietf-routing:ospf",
                      "name": "1",
                      "ietf-ospf:ospf": {
                        "instance": [
                          {
                            "af": "ipv4",
                            "areas": {
                              "area": [
                                {
                                  "area-id": "203.0.113.1",
                                  "interfaces": {
                                    "interface": [
                                      {
                                        "name": "eth2",
                                        "cost": 10
                                      }
                                    ]
                                  }
                                }
                              ]
                            }
                          }
```

```
                    ]
                  }
                }
              ]
            }
          }
        }
      ]
    },

    "ietf-interfaces:interfaces": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.10",
                  "prefix-length": 24,
                }
              ]
            }
          },
          {
            "name": "eth1",
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            },
            "ni:bind-network-instance-name": "vrf-red"
          },
          {
            "name": "eth2",
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            },
            "ni:bind-network-instance-name": "vrf-blue"
```

```
          }
        ]
      }
    },

    "ietf-system:system": {
      "authentication": {
        "user": [
          {
            "name": "john",
            "password": "$0$password"
          }
        ]
      }
    }
  }
```

B.2. State Data

The following shows state data for the example above.

```
{
  "ietf-network-instance:network-instances": {
    "network-instance": [
      {
        "name": "vrf-red",
        "vrf-root": {
          "ietf-routing:routing-state": {
            "router-id": "192.0.2.1",
            "control-plane-protocols": {
              "control-plane-protocol": [
                {
                  "type": "ietf-routing:ospf",
                  "name": "1",
                  "ietf-ospf:ospf": {
                    "instance": [
                      {
                        "af": "ipv4",
                        "areas": {
                          "area": [
                            {
                              "area-id": "203.0.113.1",
                              "interfaces": {
                                "interface": [
                                  {
                                    "name": "eth1",
                                    "cost": 10
                                  }
```

```
                                ]
                              }
                            }
                          ]
                        }
                      }
                    ]
                  }
                }
              ]
            }
          }
        },
        {
          "name": "vrf-blue",
          "vrf-root": {
            "ietf-routing:routing-state": {
              "router-id": "192.0.2.2",
              "control-plane-protocols": {
                "control-plane-protocol": [
                  {
                    "type": "ietf-routing:ospf",
                    "name": "1",
                    "ietf-ospf:ospf": {
                      "instance": [
                        {
                          "af": "ipv4",
                          "areas": {
                            "area": [
                              {
                                "area-id": "203.0.113.1",
                                "interfaces": {
                                  "interface": [
                                    {
                                      "name": "eth2",
                                      "cost": 10
                                    }
                                  ]
                                }
                              }
                            ]
                          }
                        }
                      ]
                    }
                  }
                ]
```

```
              }
            }
          }
        }
      ]
    },

    "ietf-interfaces:interfaces-state": {
      "interfaces": {
        "interface": [
          {
            "name": "eth0",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C0",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.10",
                  "prefix-length": 24,
                }
              ]
            }
          },
          {
            "name": "eth1",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
            "phys-address": "00:01:02:A1:B1:C1",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          },
          {
            "name": "eth2",
            "type": "iana-if-type:ethernetCsmacd",
            "oper-status": "up",
```

```
            "phys-address": "00:01:02:A1:B1:C2",
            "statistics": {
               "discontinuity-time": "2017-06-26T12:34:56-05:00"
            },
            "ip:ipv4": {
              "address": [
                {
                  "ip": "192.0.2.11",
                  "prefix-length": 24,
                }
              ]
            }
          }
        ]
      }
    },

    "ietf-system:system-state": {
      "platform": {
        "os-name": "NetworkOS"
      }
    }

    "ietf-yang-library:modules-state": {
      "module-set-id": "123e4567-e89b-12d3-a456-426655440000",
      "module": [
        {
          "name": "iana-if-type",
          "revision": "2014-05-08",
          "namespace":
          "urn:ietf:params:xml:ns:yang:iana-if-type",
          "conformance-type": "import"
        },
        {
          "name": "ietf-inet-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-inet-types",
          "conformance-type": "import"
        },
        {
          "name": "ietf-interfaces",
          "revision": "2014-05-08",
          "feature": [
            "arbitrary-names",
            "pre-provisioning"
          ],
          "namespace":
```

```
            "urn:ietf:params:xml:ns:yang:ietf-interfaces",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-ip",
            "revision": "2014-06-16",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-ip",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-network-instance",
            "revision": "2017-03-13",
            "feature": [
              "bind-network-instance-name"
            ],
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-network-instance",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-ospf",
            "revision": "2017-03-12",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-ospf",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-routing",
            "revision": "2016-11-04",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-routing",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-system",
            "revision": "2014-08-06",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-system",
            "conformance-type": "implement"
          },
          {
            "name": "ietf-yang-library",
            "revision": "2016-06-21",
            "namespace":
            "urn:ietf:params:xml:ns:yang:ietf-yang-library",
            "conformance-type": "implement"
          },
          {
```

```
          "name": "ietf-yang-schema-mount",
          "revision": "2017-05-16",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount",
          "conformance-type": "implement"
        },
        {
          "name": "ietf-yang-types",
          "revision": "2013-07-15",
          "namespace":
          "urn:ietf:params:xml:ns:yang:ietf-yang-types",
          "conformance-type": "import"
        }
      ]
    },

    "ietf-yang-schema-mount:schema-mounts": {
      "mount-point": [
        {
          "module": "ietf-network-instance",
          "label": "vrf-root",
          "use-schema": [
            {
              "name": "ni-schema",
              "parent-reference": [
                "/*[namespace-uri() = 'urn:ietf:...:ietf-interfaces']"
              ]
            }
          ]
        }
      ],
      "schema": [
        {
          "name": "ni-schema",
          "module": [
            {
               "name": "ietf-routing",
               "revision": "2016-11-04",
               "namespace":
                 "urn:ietf:params:xml:ns:yang:ietf-routing",
               "conformance-type": "implement"
             },
            {
               "name": "ietf-ospf",
               "revision": "2017-03-12",
               "namespace":
                 "urn:ietf:params:xml:ns:yang:ietf-ospf",
               "conformance-type": "implement"
```

```
            }
          ]
        }
      ]
    }
  }
```

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

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com