draft-ietf-rtgwg-ni-model-03
Impact on LxVPN device models

Lou Berger <lberger@labn.net>
Christian Hopps <chopps@chopps.org>
Acee Lindem <acee@cisco.com>
Dean Bogdanovic <ivandean@gmail.com>
Xufeng Liu <Xufeng_Liu@jabil.com>

Repo:  https://github.com/ietf-rtg-area-yang-arch-dt/meta-model
LNEs and NIs: Modeling Device Partitioning

LNE: Logical Network Element
- Separate management sub-domains
  - Sub-domains can be managed independently and by a top level manager (managed=true)
  - Commonly called logical system or router; or virtual switch, chassis, fabric, or device context
- Can be supported via multiple logical devices and VMs
  - Where only limited top level management of subdomains is supported

NI: Network Instance
- Separate routing / switching domains
  - Can represent of an RFC 4364 VRF or a Layer 2 Virtual Switch Instance (VSI) or a bridge/router (i.e., both)
  - General virtualized instance implying a separate L2, L3, or L2/L3 context.
    - For L3, this implies a unique IPv4/IPv6 address space.

Focus of this discussion
Status Summary

• Drafts use YANG Schema Mount to support virtual/logical partitioning of router and switch resources
  • draft-ietf-netmod-schema-mount-05
  • Each LNE/NI gets an independent data instance
  • With a YANG module root, and separate instances of YANG modules
  • Implementations decide what modules are included under a root
    • Modules included under mount point may be different from modules at device’s top level

• Both drafts have been updated and are ready for LC
  • Technical details were in flux due to Schema Mount open issues
  • Issues now resolved, expected LC without significant technical changes
Schema Mount Tree Representation

- **Schema Mount Additions**
  - `mp` for schema mount points
  - `/` for a mounted module
  - `@` for a node made available via a schema mount parent reference
  - Module (nodes/leaves/etc) marked `ro` when schema mount config leaf = false

- **Example**

  ```
  +--mp vrf-root?
      +--ro rt:routing-state/
          |   ...
      +--ro rt:routing/
          |   ...
      +--ro if:interfaces@
          |   ...
      +--ro if:interfaces-state@
          ...
  ```

- **See**
  draft-ietf-netmod-yang-tree-diagrams-01
NIs are used to model:

1. Information within an instance, i.e. CE context information
   - Using one of 3 *well known mount points*: VRFs, VSIs, VSI+VRFs
2. *Per instance*, related information in the core/PE instance
   - Using LxVPN *technology-specific augmentations*
     - L3VPN examples: BGP MPLS L3VPN over MPLS, over tunnels
     - L2VPN examples: VPLS, EVPN+MPLS, EVPN+VxLAN, …
LxVPN Support

- **NI Type**
  - For per VRF, PE/core information

- **Root Type**
  - For VRF/VSI information in the CE/Vxx context

---

**draft-ietf-rtgwg-ni-model-03:**

```plaintext
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?
```
NI Likely Impact on BESS

There are three types of LxVPN information to model:

1. **Core/PE + not instance specific**
   - Goes in augmentation of a module present at top level
     e.g., bgp, interfaces, or even top of network instances

2. **Core/PE + is associated with a named NI**
   - Goes into augmentation of:
     a) ni-types (preferred) or
     b) other module and associated with bind-ni-name (ala interfaces)

3. **CE-Context Information, per VRF/VSI**
   - Goes in augmentation of a module present under vrf/vsi-root
   - Do any any of these exist?

• Reminder: Implementations, not models, decide what gets mounted at top level and under each vrf/vsi-root
Thank you!
LxVPN Technology Specific Information

Two type of PE/Core information:

1. Per VRF/VSI instance information
   - May differs based on LxVPN technology
     - L2VPN – VPLS, VxLAN, EVPN, ...
     - L3VPN – MPLS, IP tunnels, ...
   - Supported via **ni-types** choice statement
     - Empty in base model
     - To be augmented with technology specific cases

2. Information shared across NI instances
   - Supported via augmentations to any top top-level module(s)
     - E.g., BGP or even top of NI model

```
++-rw network-instances
   ++-rw network-instance* [name]
      ++-rw name string
      ++-rw enabled? boolean
      ++-rw description? string
      ++-rw (ni-type)?
         |   ++-:(l3vpn) //augmentation
         |      ++-rw l3vpn:l3vpn
         |      |   ...   // config data
```

Type-specific augmentation

Composite Tree
Per VRF/VSI (CE Context) Information

- Supported via standard top level modules under a per-instance root mount point
  - Specific modules included under a mount point is an implementation choice
  - Modules are typically based on L2 or L3 type and not (PE) VPN technology

- Three types of Nis have been identified
  1. VRFs for L3VPNs
  2. VSIs for L2VPNs
  3. VSI+VRF for L2+L3VPNs (bridge/routers)

- Schema mount defines the schema (i.e., module list) on a per mount point name basis
  - So need named mount point per type

---

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?

//one root required per NI
VRF Mount Point Example: OSPF in VRF

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
      |        +--ro rt:routing-state/
      |          |   +--ro router-id?
      |          |   +--ro control-plane-protocols
      |          |     +--ro control-plane-protocol* |
      |          |          |     +--ro ospf:ospf/
      |          |        ...
      |   +--rw rt:routing/
      |          |   +--rw router-id?
      |          |   +--rw control-plane-protocols
      |          |     +--rw control-plane-protocol* |
      |          |          |     +--rw ospf:ospf/
      |          |          ...
    +--ro if:interfaces@
      |    ...
    +--ro if:interfaces-state@
      |    ...

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

- None really
- But LNEs are related to NIs as both are used to manage logical partitioning of device resources and sometimes confused
  - LNEs ~= VM/VNF
    - Sometimes called: logical system or router; virtual switch, chassis, or fabric
  - NI = VRF or VSI (Virtual Switch Instance)
module: ietf-logical-network-element
   +--rw logical-network-elements
      +--rw logical-network-element* [name]
         +--rw name string
         +--rw managed? boolean
         +--rw description? string
         +--mp root

augment /if:interfaces/if:interface:
   +--rw bind-lne-name?
      -> /logical-network-elements/logical-network-element/name

notifications:
   +---n bind-lne-name-failed
      +--ro name -> /if:interfaces/interface/name
      +--ro bind-lne-name -> /if:interfaces/interface/lne:bind-lne-name
      +--ro error-info? string

Covers cases of asynchronous interface NI bind failures
LNE: Module Example

module: ietf-logical-network-element
   +-rw logical-network-elements
      +-rw logical-network-element* [name]
      |    +-rw managed?       boolean
      |    +-rw name           string
      |    +-mp root
      |      ...
      |       ...
      +-rw sys:system/
      |    ...
      +-ro sys:system-state/
      |    ...
      +-ro rt:routing-state/
      |    +-ro router-id? quad
      |    +-ro control-plane-protocols
      |    |    +-ro control-plane-protocol* []
      |    |    |    +-ro ospf:ospf/
      |    |    |    |    +-ro instance* [af]
      |    |    |    |    |    ...
      |    +-rw rt:routing/
      |    ...
      +-rw if:interfaces/
      |    ...
      +-ro if:interfaces-state/
      ...

Managed=true