Routing Area Yang Architecture Design Team Update

Members: Acee Lindem, Anees Shaikh, Christian Hopps,
Dean Bogdanovic, Ebban Aries, Lou Berger,
Qin Wu, Rob Shakir, Xufeng Liu, Yingzhen Qu

Wiki: http://trac.tools.ietf.org/area/rtg/trac/wiki/RtgYangArchDT

Repo: https://github.com/ietf-rtg-area-yang-arch-dt/

Agenda

- DT status
- Update on Routing Types
- Discussion:
  Handling Transition to Revised Data Stores
- LNE/NI Examples (if time permits)

DT current “work” topics

1. Meta Model
2. OpStateⁱ/revised data stores
   YANG Relationship of Config and Operational State (and intended)
3. Conventions
4. Model Classifications

   Individual Draft in NetMod
5. Requested topics
   - RegEx, expired RTG Area drafts

Status: RTGWG drafts

- Started with so called Meta-Model, Now:
  - 2 Standards Track Models
    - Logical Network Element (draft-ietf-rtgwg-lne-model)
    - Network Instance (draft-ietf-rtgwg-ni-model)
    - Both use / gated schema mount
      - Only a minor update this time, waiting on NetMod
      - Expect to improve examples and narrative in next version
  - 1 Informational Meta Model
    - Network Device YANG Organizational Model (draft-rtgyangdt-rtgwg-device-model)
    - If module-tags accepted in a WG, will realign to use tags to provide logical organization
- Conventions:
  - Routing Area Common YANG Data Types (draft-ietf-rtgwg-routing-types)

Schema Mount: Gating Issues

- Covered in draft
  - https://tools.ietf.org/html/draft-ietf-netmod-schema-mount-04#appendix-B

1. Referencing Mount Points Using Schema Node Identifiers
   - Mount point identified by path vs node name
2. Defining the "mount-point" Extension in a Separate Module
   - Simplifies support for inline only implementation
   - Also suggests removing use of mount-point for non-inline case – What does WG think?
3. Parent References
   - Currently limited to absolute paths – Is this acceptable?
4. RPC Operations and Notifications in Mounted Modules
   - A remaining corner case to be worked out…
5. Tree Representation
   - A remaining TBD (along with Security considerations)
6. Design-Time Mounts
   - Authors are looking for an possible initial use case – Any takers?

Status: OpState

- Tracking NetMod DT
  - See A Revised Conceptual Model for YANG Datastores
    - draft-nmdsdt-netmod-revised-datastores
- Note:
  - RFC8022 (draft-ietf-netmod-routing-cfg) published
    - Follows RFC7223 –config/-state convention
  - RFC6087bis about to be published
    - Section 5.23 provides related guidance,
      but not a simple directive
- Transition conventions
  - More on this later…

Status: Conventions

- **Routing Area Common YANG Data Types**
  draft-rtgyangdt-rtgwg-routing-types
- Covers types expected to be generally useful to YANG modules developed in the routing area
  - Looking to wrap this document up & LC
- Repo:
  - https://github.com/ietf-rtg-area-yang-arch-dt/conventions-features

New Draft: YANG Module Tags

See NetMod for details

- draft-rtgyangdt-netmod-module-tags-00
  - Draft is targeted at NetMod as is core YANG functionality
- Objective is to provide user controllable per-module meta-data to help classify and organize modules
  - User controllable
    - Default values set during module definition, or implementation, or by user
  - Standardized or not
    - Uses well known required prefixes
    - "ietf:", "vendor:" or "local:“

Routing Types Update

draft-ietf-rtgwg-routing-types-02

Xufeng Liu, Yingzhen Qu, Acee Lindem,
Christian Hopps, Lou Berger

Goals and Usage

- Same or similar YANG types are used in the routing area YANG models
  - Defined multiple times in individual models
  - Definitions not Consistent
- These common types are collected and defined in this model
  - In a sharable module
  - Should be imported in routing area models
  - Ensures consistent definitions for common routing types
  - Analogous to ietf-yang-types but types specific to routing area
- Several drafts have started to used this module:
  - OSPF,TE, MPLS base, L2VPN, EVPN, LDP, PIM

Changes From Last Meeting

- Renamed the following types for consistency
  - Changed multicast-source-ipv4-addr-type to ipv4-multicast-source-address
  - Changed multicast-source-ipv6-addr-type to ipv6-multicast-source-address
  - Changed ieee-bandwidth to bandwidth-ieee-float32
- New Types and Groupings

New Types and Groupings

New Types

- route-target-type
- ipv4-multicast-group-address
- ipv6-multicast-group-address
- ip-multicast-group-address
- generalized-label
- mpls-label-special-purpose
- mpls-label-general-use
- mpls-label
- mpls-label-stack

New Groupings

- mpls-label-stack
- vpn-route-targets

Next Steps

- WG last call?
  - Most useful if it progresses ahead of the first wave of importing routing models

Transitioning to Revised Data Stores

Xufeng Liu

Goals

Identify a common approach for the Routing Area

- Work with current NETCONF datastores.
- Work with revised datastores without rewriting.
- Clear migration paths.
- Minimum impacts to implementers and operators during migration.

- TE-Tunnel module will be used an example
  - From draft-ietf-teas-yang-te-03

TE Tunnel Model with Revised Datastore

- Simplified TE tunnel model.
- Use the proposed style working with revised datastore.
- Can possibly be used as one reference structure.

```
+--rw te!
   +--rw tunnels
      +--rw tunnel* [name]
         +--rw name                  leafref
         +--rw protection-type?      identityref     // Configuration and
operational
         +--rw reoptimize-timer?     uint16
         +--rw set-bandwidth?        bandwidth-kbps
         +--ro state
         |  +--ro oper-status?       identityref     // Derived operational
state
         |  +--ro protection-statue? identityref
         +--ro statistics
            +--ro octets?            yang:counter64  // Statistics
            +--ro errors?            yang:counter32
```

Option 1: Split Top-level Trees

- Works with current datastore.
- Has a clear migration path.
- Migration requires restructuring and new publications:
  - Deprecating portion of the existing model.
  - Adding new portion to the existing model.

TE Tunnel Model with Split Top-level Trees

```
+--rw te!
|  +--rw tunnels
|     +--rw tunnel* [name]
|        +--rw name                 leafref
|        +--rw protection-type?   identityref       // Configuration
|        +--rw reoptimize-timer?  uint16
|        +--rw set-bandwidth?     bandwidth-kbps
+--ro te-state!
   +--ro tunnels
      +--ro tunnel* [name]
         +--ro name                 leafref
         +--ro protection-type?   identityref       // Operational
         +--ro reoptimize-timer?  uint16
         +--ro set-bandwidth?     bandwidth-kbps
         +--ro oper-status?       identityref       // Derived operational
state
         +--ro protection-statue? identityref
         +--ro statistics
            +--ro octets?         yang:counter64    // Statistics
            +--ro errors?         yang:counter32
```

Migration to Revised Datastore

```
+--rw te!
|  +--rw tunnels
|     +--rw tunnel* [name]
|        +--rw name                 leafref
|        +--rw protection-type?     identityref          // Configuration
|        +--rw reoptimize-timer?    uint16
|        +--rw set-bandwidth?       bandwidth-kbps
|        +--ro oper-status?         identityref          // Derived operational
state
|        +--ro protection-statue?   identityref
|        +--ro statistics
|           +--ro octets?            yang:counter64       // Statistics
|           +--ro errors?            yang:counter32
+--ro te-state!
   +--ro tunnels
      +--ro tunnel* [name]
         +--ro name                 leafref
         +--ro protection-type?     identityref          // Operational
         +--ro reoptimize-timer?    uint16
         +--ro set-bandwidth?       bandwidth-kbps
         +--ro oper-status?         identityref          // Derived operational
state
         +--ro protection-statue?   identityref
         +--ro statistics
            +--ro octets?           yang:counter64       // Statistics
```

Add

Deprecate

Option 2: Split Containers

- Works with current datastore.
- Has a clear migration path.
- Migration does not requires restructuring:
    - Deprecating portion of the existing model.
- All configurable leaves are under container “config”, even for their operational states.

TE Tunnel Model with Split Containers

```
+--rw te!
   +--rw tunnels
      +--rw tunnel* [name]
         +--rw name                  leafref
         +--rw config
         |  +--rw protection-type?   identityref     // Configuration
         |  +--rw reoptimize-timer?  uint16
         |  +--rw set-bandwidth?     bandwidth-kbps
         +--ro state
         |  +--ro protection-type?   identityref     // Operational
         |  +--ro reoptimize-timer?  uint16
         |  +--ro set-bandwidth?     bandwidth-kbps
         |  +--ro oper-status?       identityref     // Derived operational
state
         |  +--ro protection-statue? identityref
         +--ro statistics
            +--ro octets?            yang:counter64  // Statistics
            +--ro errors?            yang:counter32
```

Migration to Revised Datastore

```
+--rw te!
   +--rw tunnels
      +--rw tunnel* [name]
         +--rw name                   leafref
         +--rw config
         |  +--rw protection-type?    identityref     // Configuration
         |  +--rw reoptimize-timer?   uint16
         |  +--rw set-bandwidth?      bandwidth-kbps
         +--ro state
         |  +--ro protection-type?    identityref     // Operational
         |  +--ro reoptimize-timer?   uint16
         |  +--ro set-bandwidth?      bandwidth-kbps
         |  +--ro oper-status?        identityref     // Derived operational
state
         |  +--ro protection-statue? identityref
         +--ro statistics
            +--ro octets?             yang:counter64  // Statistics
            +--ro errors?             yang:counter32
```

Style is different from revised datastore proposal

Deprecate

Option 3: Additional State Containers

- Works with current datastore.
- Has a clear migration path.
- Migration does not requires restructuring:
  - Deprecating portion of the existing model.
- All configurable leaves are directly under list item (not under container “config”)

TE Tunnel Model with Additional State Containers

```
+--rw te!
   +--rw tunnels
      +--rw tunnel* [name]
         +--rw name                  leafref
         +--rw protection-type?      identityref     // Configuration
         +--rw reoptimize-timer?     uint16
         +--rw set-bandwidth?        bandwidth-kbps
         +--ro state
         |  +--ro protection-type?   identityref     // Operational
         |  +--ro reoptimize-timer?  uint16
         |  +--ro set-bandwidth?     bandwidth-kbps
         |  +--ro oper-status?       identityref     // Derived operational
state
         |  +--ro protection-statue? identityref
         +--ro statistics
            +--ro octets?            yang:counter64  // Statistics
            +--ro errors?            yang:counter32
```

Migration to Revised Datastore

```
+--rw te!
   +--rw tunnels
      +--rw tunnel* [name]
         +--rw name                  leafref
         +--rw protection-type?      identityref     // Configuration
         +--rw reoptimize-timer?     uint16
         +--rw set-bandwidth?        bandwidth-kbps
         +--ro state
         |  +--ro protection-type?   identityref     // Operational
         |  +--ro reoptimize-timer?  uint16
         |  +--ro set-bandwidth?     bandwidth-kbps
         |  +--ro oper-status?       identityref     // Derived operational
state
         |  +--ro protection-statue? identityref
         +--ro statistics
            +--ro octets?            yang:counter64  // Statistics
            +--ro errors?            yang:counter32
```

Deprecate

Option 4: No State Containers

- Benefits
  - No changes or later deprecation needed for migration.
  - Benefit from better organization now.
  - Gain benefits of multiple datastores later.
- Drawbacks
  - Will not work for some models. In particular:
    - Models with system created state that aligns with config state.
    - Models where the a state value is often unaligned with it's config value.

TE Tunnel Model with No State Containers

```
+--rw te!
   +--rw tunnels
      +--rw tunnel* [name]
         +--rw name                  leafref
         +--rw protection-type?      identityref     // Configuration
         +--rw reoptimize-timer?     Uint16          // Not support oper leaf with the same
name
         +--rw set-bandwidth?        bandwidth-kbps
         +--ro oper-status?          identityref     // Derived operational state
         +--ro protection-statue?    identityref
         +--ro statistics
            +--ro octets?            yang:counter64  // Statistics
            +--ro errors?            yang:counter32
```

Migration to Revised Datastore

```
+--rw te!
   +--rw tunnels
      +--rw tunnel* [name]
         +--rw name                  leafref
         +--rw protection-type?      identityref     // Configuration and
operational
         +--rw reoptimize-timer?     uint16
         +--rw set-bandwidth?        bandwidth-kbps
         +--ro oper-status?          identityref     // Derived operational
state
         +--ro protection-statue?    identityref
         +--ro statistics
            +--ro octets?            yang:counter64  // Statistics
            +--ro errors?            yang:counter32
```

Gain benefits of additional states

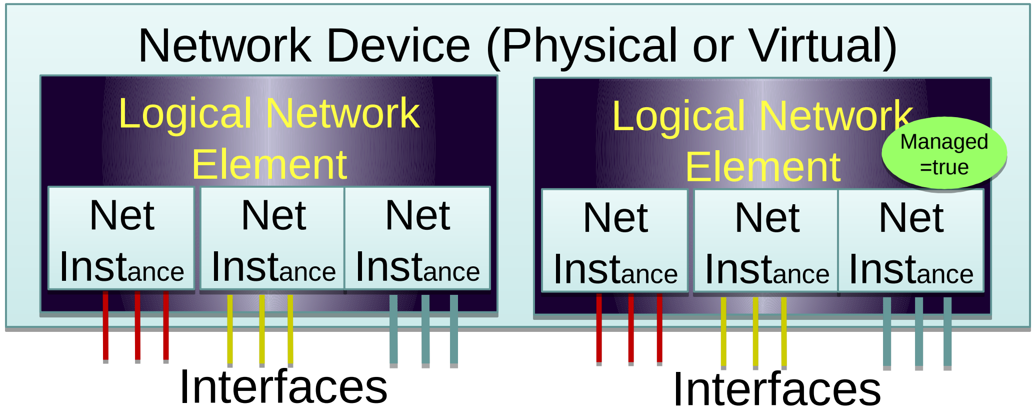
Summary

- Revised datastores are coming, but aren’t here yet
- Multiple ways we can move models forward without waiting
  - Would be best to have a common convention, at least within the area
- Options
  1. Top-level split
  2. Split containers
  3. Additional state containers
  4. No state containers
  5. Something else

LNI/NI Examples

(time permitting)

Reminder



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

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.

Logical Network Element Example

- Implementation time schema.
- Non-shared ietf-interfaces schema.
- ietf-routing is mounted under LNE (Logical Network Element Example).
- ietf-ospf augments ietf-routing.

LNE Schema View

```
module: ietf-logical-network-element
   +--rw logical-network-elements
      +--rw logical-network-element* [name]
         +--rw name                 string
         +--rw root?                yangmnt:mount-
point
            // module: ietf-library
            // module: ietf-routing
            // module: ietf-interfaces
            +--rw interfaces
            |  +--rw interface* [name]
            |     +--rw name                      string
            +--ro interfaces-state
               +--ro interface* [name]
               +--ro name                         string
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name
string
   |     +--rw lne:bind-lne-name?
string
   +--ro interfaces-state
      +--ro interface* [name]
```

mounted

Mounted*

augment

```
// module: ietf-routing
+--ro routing-state
|  +--ro router-id?                  yang:dotted-
quad
|  +--ro control-plane-protocols
|  |  +--ro control-plane-protocol* [type name]
         // module: ietf-ospf
         +--ro ospf
            +--ro instance* [af]
+--rw routing
   +--rw router-id?                  yang:dotted-
quad
   +--rw control-plane-protocols
   |  +--rw control-plane-protocol* [type name]
         // module: ietf-ospf
   |     +--rw ospf:ospf
   |        +--rw ospf:instance* [af]
   |           +--rw ospf:areas
   |           |  +--rw ospf:area* [area-id]
   |           |     +--rw ospf:interfaces
   |           |        +--rw ospf:interface*
[name]
   |           |           +--rw ospf:name

                                    if:interface-
ref
   |           |           +--rw ospf:cost?
uint16
```

augment

augment

LNE Data View

```
module: ietf-logical-network-element
   +--rw logical-network-elements
      +--rw logical-network-element
[“name”:“ne1”]
         +--rw root?          yangmnt:mount-
point
            // module: ietf-library
            // module: ietf-routing
            // module: ietf-interfaces
            +--rw interfaces
            |  +--rw interface [ ]
            +--ro interfaces-state
               +--ro interface [“name”:“eth1”]
               +--ro “oper-status”: “up”

module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface [“name”: “eth0”]
   |  +--rw interface [“name”: “eth1”]
   |     +--rw “lne:bind-lne-name”: “ne1”
   +--ro interfaces-state
         +--ro “oper-status”: “up”
```

```
// module: ietf-routing
+--ro routing-state
|  +--ro “router-id” : “1.1.1.1”
|  +--ro control-plane-protocols
|  |  +--ro control-plane-protocol

[“type”:“ospf”,“name”:“1”]
         // module: ietf-ospf
         +--ro ospf
            +--ro instance [“af”:“ipv4”]
+--rw routing
   +--rw “router-id” : “1.1.1.1”
   +--rw control-plane-protocols
   |  +--rw control-plane-protocol

[“type”:“ospf”,“name”:“1”]
         // module: ietf-ospf
   |     +--rw ospf:ospf
   |        +--rw ospf:instance [“af”:“ipv4”]
   |           +--rw ospf:areas
   |           |  +--rw ospf:area [“area-
id”:“2.2.2.2”]
   |           |     +--rw ospf:interfaces
   |           |        +--rw ospf:interface

[“name”:“eth1”]
   |           |           +--rw “ospf:cost” : 10
```

Mapped interface may or may not have the same name – based on host implementation, not config

LNE Implementation: Static Mounting Data

```
"ietf-yang-schema-mount:schema-mounts": {
  "mount-point": [
    {
      "module": "ietf-logical-network-
element",
      "name": "root",
      "use-schema": [
        {
          "name": "lne-schema"
        }
      ]
    }
  ],
```

Requires: Managed=true

Managed=false MUST use inline

```
"schema": [
    {
      "name": “lne-schema",
      "module": [
        {
           "name": "ietf-routing",
           "revision": "2016-11-04“,
           "namespace":
             "urn:ietf:params:xml:ns:yang:ietf
        -routing",
           "conformance-type": "implement"

         },
         {
            "name": "ietf-interfaces",
            "revision": "2014-05-08",
            "namespace":

"urn:ietf:params:xml:ns:yang:ietf
                   -interfaces",
            "conformance-type": "implement“
          }
        ]
     }
```

Unnecessary duplicates

LNE Implementation: Static YANG Library Data

```
"ietf-yang-library:modules-state": {
  "module-set-id": "14e2ab5dc325f6d86f743e8d3ade233f1a61a899",
  "module": [
    { "name": "iana-if-type", "revision": "2014-05-08",
      "namespace": "urn:ietf:params:xml:ns:yang:iana-if-type", "conformance-type": "implement"
},
    { "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",
      "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-key-chain", "revision": "2017-02-16",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-key-chain", "conformance-type":
"implement" },
    { "name": "ietf-logical-network-element", "revision": "2016-10-21", "feature": [ "bind-
lne-name" ],
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-logical-network-element", "conformance-
type": "implement" },
    { "name": "ietf-ospf", "revision": "2017-03-12",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-ospf", "con
```

Only in mounted library instance*

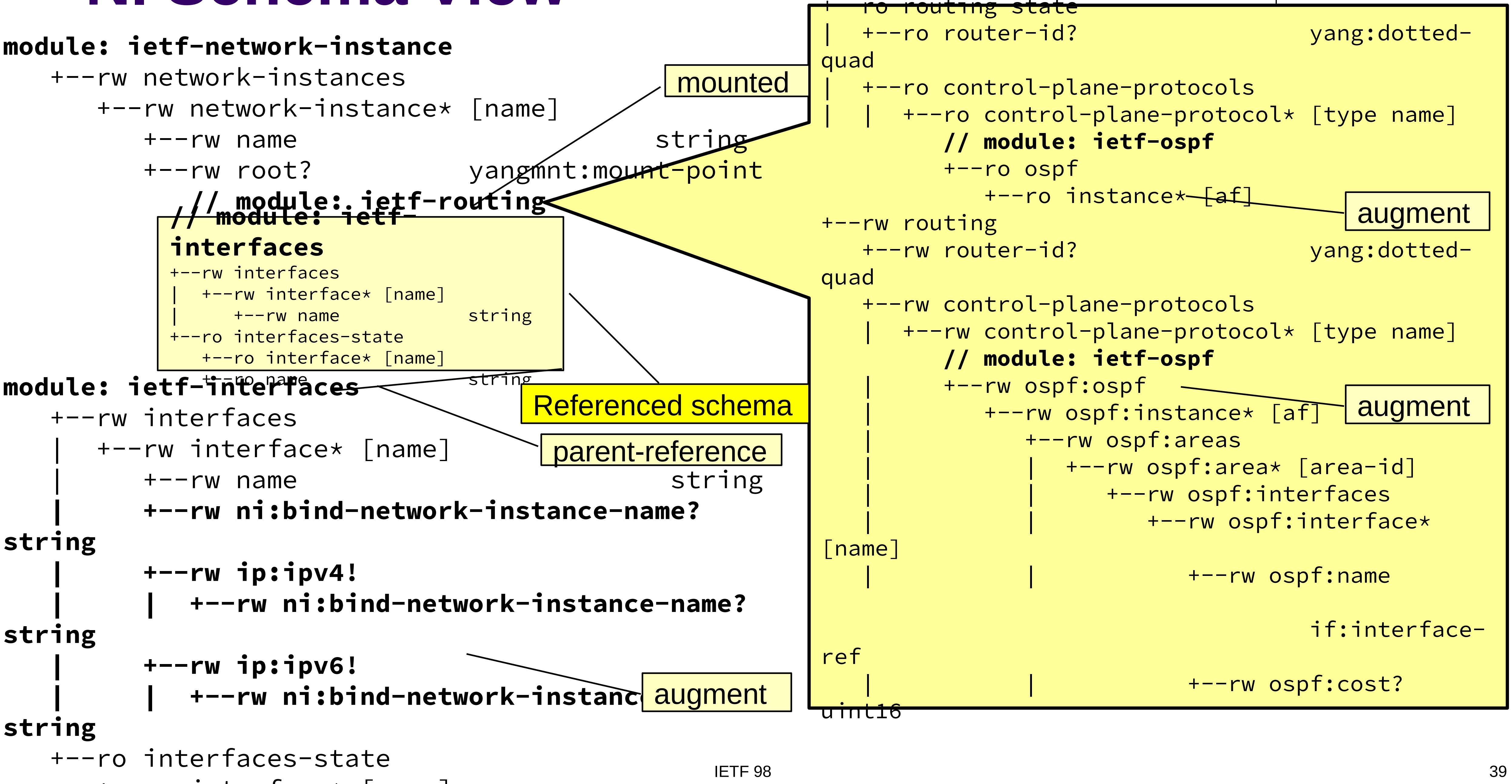
LNE Implementation:
Static YANG Library Data

```
    { "name": "ietf-routing-types", "revision": "2017-02-27",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing-types", "conformance-type":
"import" },
    { "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": "2017-03-06",
      "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"
}
  ]
}
```

Network Instance Example

- Implementation time schema.
- Shared ietf-interfaces schema.
- ietf-routing is mounted under NI (Network Instance).
- ietf-ospf augments ietf-routing.

NI Schema View



```
module: ietf-network-instance
   +--rw network-instances
      +--rw network-instance* [name]
         +--rw name                      string
         +--rw root?           yangmnt:mount-point
            // module: ietf-routing
```

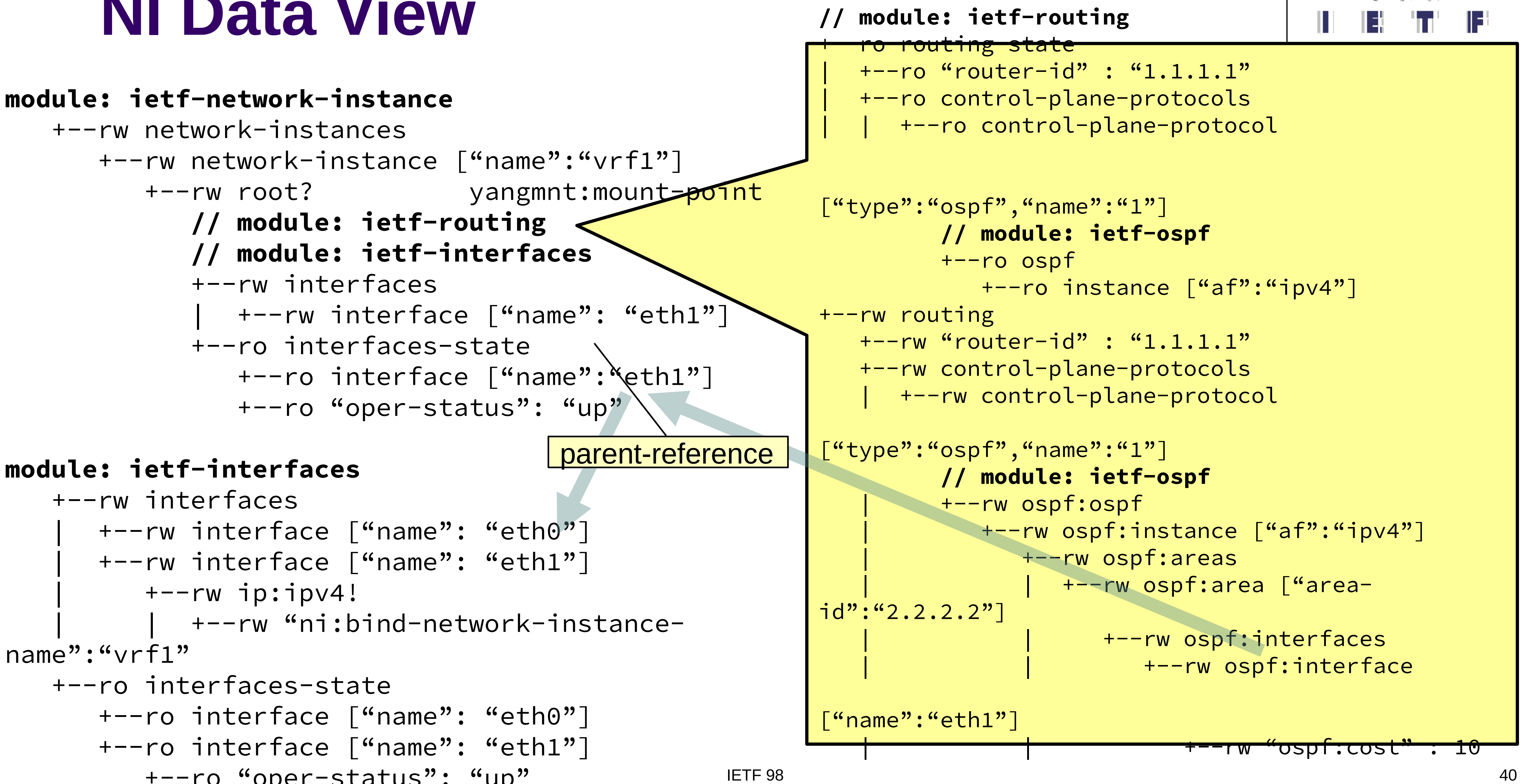
```
// module: ietf-
interfaces
+--rw interfaces
|  +--rw interface* [name]
|     +--rw name               string
+--ro interfaces-state
   +--ro interface* [name]
      +--ro name               string
```

```
module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name                         string
   |     +--rw ni:bind-network-instance-name?
string
   |     +--rw ip:ipv4!
   |     |  +--rw ni:bind-network-instance-name?
string
   |     +--rw ip:ipv6!
   |     |  +--rw ni:bind-network-instanc
string
   +--ro interfaces-state
```

```
// module: ietf-routing
+--ro routing-state
|  +--ro router-id?                  yang:dotted-
quad
|  +--ro control-plane-protocols
|  |  +--ro control-plane-protocol* [type name]
         // module: ietf-ospf
         +--ro ospf
            +--ro instance* [af]
+--rw routing
   +--rw router-id?                  yang:dotted-
quad
   +--rw control-plane-protocols
   |  +--rw control-plane-protocol* [type name]
         // module: ietf-ospf
   |     +--rw ospf:ospf
   |        +--rw ospf:instance* [af]
   |           +--rw ospf:areas
   |           |  +--rw ospf:area* [area-id]
   |           |     +--rw ospf:interfaces
   |           |        +--rw ospf:interface*
[name]
   |           |           +--rw ospf:name

                                     if:interface-
ref
   |           |           +--rw ospf:cost?
uint16
```

NI Data View



```
module: ietf-network-instance
   +--rw network-instances
      +--rw network-instance [“name”:“vrf1”]
         +--rw root?          yangmnt:mount-point
            // module: ietf-routing
            // module: ietf-interfaces
            +--rw interfaces
            |  +--rw interface [“name”: “eth1”]
            +--ro interfaces-state
               +--ro interface [“name”:“eth1”]
               +--ro “oper-status”: “up”

module: ietf-interfaces
   +--rw interfaces
   |  +--rw interface [“name”: “eth0”]
   |  +--rw interface [“name”: “eth1”]
   |     +--rw ip:ipv4!
   |     |  +--rw “ni:bind-network-instance-
name”:“vrf1”
   +--ro interfaces-state
      +--ro interface [“name”: “eth0”]
      +--ro interface [“name”: “eth1”]
         +--ro “oper-status”: “up”
```

parent-reference

```
// module: ietf-routing
+--ro routing-state
|  +--ro “router-id” : “1.1.1.1”
|  +--ro control-plane-protocols
|  |  +--ro control-plane-protocol

[“type”:“ospf”,“name”:“1”]
         // module: ietf-ospf
         +--ro ospf
            +--ro instance [“af”:“ipv4”]
+--rw routing
   +--rw “router-id” : “1.1.1.1”
   +--rw control-plane-protocols
   |  +--rw control-plane-protocol

[“type”:“ospf”,“name”:“1”]
         // module: ietf-ospf
   |     +--rw ospf:ospf
   |        +--rw ospf:instance [“af”:“ipv4”]
   |           +--rw ospf:areas
   |           |  +--rw ospf:area [“area-
id”:“2.2.2.2”]
   |           |     +--rw ospf:interfaces
   |           |        +--rw ospf:interface

[“name”:“eth1”]
   |           |           +--rw “ospf:cost” : 10
```

NI Implementation: Static Mounting Data

```
"ietf-yang-schema-mount:schema-mounts": {
  "mount-point": [
    {
      "module": "ietf-network-instance",
      "name": "root",
      "use-schema": [
        {
          "name": "routing-schema“,
          "parent-reference“: [
             "ietf-interfaces"
          ]
        }
      ]
    }
  ],
```

per schema based ?

```
"schema": [
    {
      "name": "routing-schema",
      "module": [
        "name": "ietf-routing",,
        "revision": "2016-11-04“,
        "namespace":
       "urn:ietf:params:xml:ns:yang:ietf-
routing",
         "conformance-type": "implement"

      ]
    }
  ]
}
```

NI Implementation: Static YANG Library Data

- Unlike LNE, must include all modules

```
"ietf-yang-library:modules-state": {
  "module-set-id": "14e2ab5dc325f6d86f743e8d3ade233f1a61a899",
  "module": [
    { "name": "iana-if-type", "revision": "2014-05-08",
      "namespace": "urn:ietf:params:xml:ns:yang:iana-if-type", "conformance-type": "implement" },
    { "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",
      "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-key-chain", "revision": "2017-02-16",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-key-chain", "conformance-type": "implement" },
    { "name": "ietf-network-instance", "revision": "2016-10-21", "feature": [ "bind-ni-name" ],
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-logical-network-element", "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" },
```

YANG Module Tags

More details

Initial Set of Tags

Tag	Description
ietf:area:art	Applications and Real-Time Area module.
ietf:area:gen	General Area module.
ietf:area:int	Internet Area module.
ietf:area:ops	Operations and Management Area module.
ietf:area:rtg	Routing Area module.
ietf:area:sec	Security Area module.
ietf:area:tsv	Transport Area module.
ietf:entity	A module for an entity (*).
ietf:service	A module for a service (*).
ietf:hardware	A module for hardware.
ietf:software	A module for software.
ietf:protocol	A module representing a protocol.
ietf:protocol:system-management	A module representing a system management protocol.
ietf:protocol:network-service	A module representing a network service protocol.
ietf:protocol:routing	A module representing a control plane routing protocol.
ietf:protocol:signaling	A module representing a control plane signaling protocol.
ietf:protocol:oam	A module representing a Operations, Administration, and Maintenance protocol.
ietf:protocol:lmp	A module representing a link management protocol.
ietf:protocol:routing:igp	An IGP protocol module.
ietf:protocol:routing:egp	An EGP protocol module.

Tag User Control

- Two proposed approaches:

1. Add tags list to YANG Library via augmentation

```
module: ietf-library-tags
 augment /yanglib:modules-state/yanglib:module:
   +--ro tags*   string
```

2. Add support to modules directly
   - Augmentation for existing, inclusion in the future
     - Using module-tags grouping
   - RPCs from control {add, remove, reset}