Network Working Group
Internet-Draft
Intended status: Experimental
Expires: June 18, 2015

A. Clemm
J. Medved
R. Varga
T. Tkacik
Cisco
N. Bahadur
Bracket Computing
H. Ananthakrishnan
Packet Design
December 15, 2014

A Data Model for Network Topologies
draft-clemm-i2rs-yang-network-topo-02.txt

Abstract

This document defines a YANG data model for network and service topologies.

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 June 18, 2015.

Table of Contents

1. Introduction

This document introduces an abstract (base) network YANG [RFC6020] [RFC6021] model that can be augmented to cover both network inventories and network/service topologies. Moreover, the document also introduces an abstract (basic) topology model that augments the basic network model and can in turn be augmented to describe many different network and service topologies. Applications can operate on any inventory or topology at a generic level, where specifics of particular inventory/topology types are not required; applications can also operate with intentory-specific data or or data specific to a particular topology level when those specifics come into play. Examples of specific topology types include Layer 2 topology, Layer 3 topologies such as Unicast IGP, IS-IS [RFC1195] and OSPF [RFC2328],

traffic engineering (TE) data [RFC3209], or any of the variety of transport and service topologies. Information specific to such a particular network topology types will be captured in separate, technology-specific models. The basic data models introduced in this document is generic in nature and can be applied to many network topologies and inventories.

The abstract (base) network YANG module introduced in this document contains a list of abstract network nodes and defines the concept of network hierarchy (network stack). The abstract network node can be augmented in inventory and topology models with inventory and topology specific attributes. Network hierarchy (stack) allows any given network to have one or more "supporting networks". The relationship of the base network model, the inventory models and the topology models is shown in the following figure (dotted lines in the figure denote possible augmentations to models defined in this document).

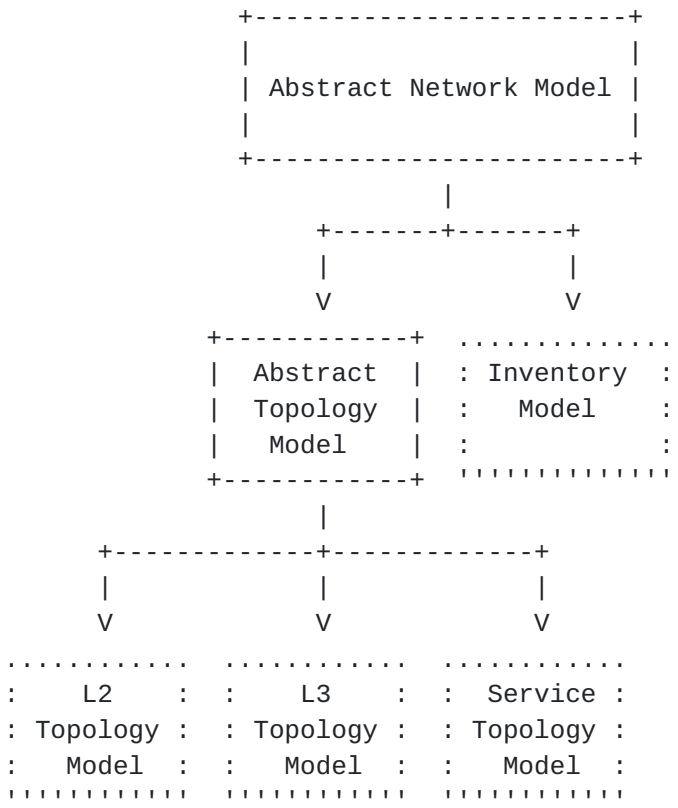

Figure 1: The network models structure

The network-topology YANG module introduced in this document defines a generic topology model at its most general level of abstraction. The module defines a topology graph and components from which it is composed: nodes, edges and termination points. Nodes represent graph

vertices and links represent graph edges. Nodes contain termination points that anchor the links. A network can contain multiple topologies, for example topologies at different layers and overlay topologies. The model therefore allows to capture relationships between topologies, as well as dependencies between nodes and termination points across topologies. An example of a topology stack is shown in the following figure.

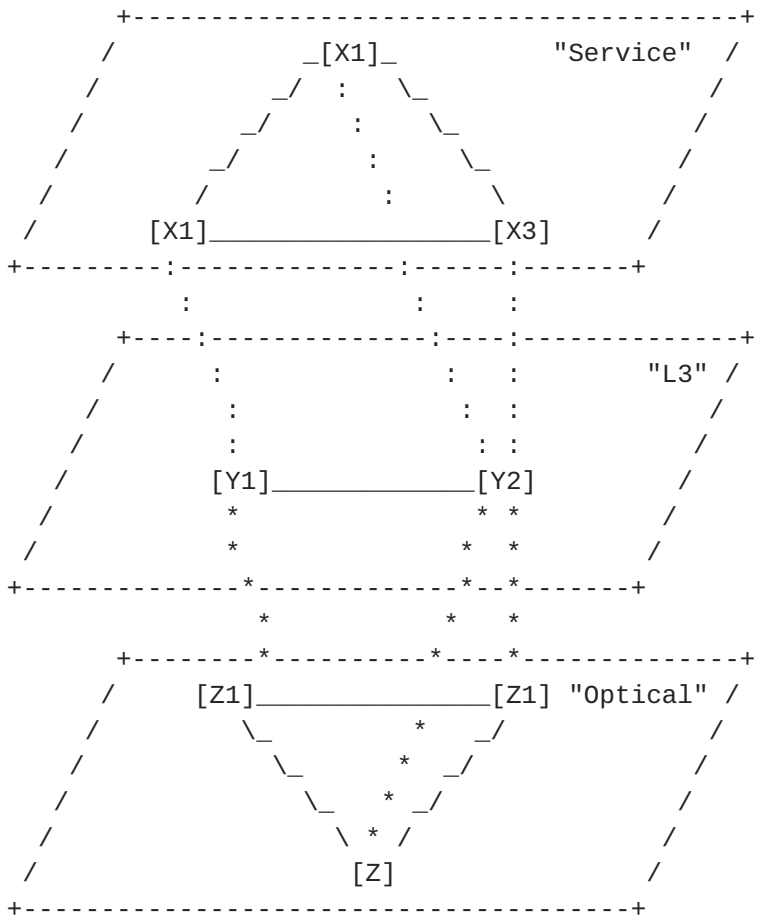

Figure 2: Topology hierarchy (stack) example

The figure shows three topology levels. At top, the "Service" topology shows relationships between service entities, such as service functions in a service chain. The "L3" topology shows network elements at Layer 3 (IP) and the "Optical" topology shows network elements at Layer 1. Service functions in the "Service" topology are mapped onto network elements in the "L3" topology, which in turn are mapped onto network elements in the "Optical" topology. The figure shows two Service Functions - X1 and X2 - mapping onto a single L3 network element; this could happen, for example, if two service functions reside in the same VM (or server) and share the same set of network interfaces. The figure shows a single "L3"

network element mapped onto multiple "Optical" network elements. This could happen, for example, if a single IP router attaches to multiple ROADMs in the optical domain.

There are multiple applications for such a data model. For example, within the context of I2RS, nodes within the network can use the data model to capture their understanding of the overall network topology and expose it to a network controller. A network controller can then use the instantiated topology data to compare and reconcile its own view of the network topology with that of the network elements that it controls. Alternatively, nodes within the network could propagate this understanding to compare and reconcile this understanding either among themselves or with help of a controller. Beyond the network element and the immediate context of I2RS itself, a network controller might even use the data model to represent its view of the topology that it controls and expose it to applications north of itself. Further use cases that the data model can be applied to are described in [topology-use-cases].

2. Definitions and Acronyms

Datastore: A conceptual store of instantiated management information, with individual data items represented by data nodes which are arranged in hierarchical manner.

Data subtree: An instantiated data node and the data nodes that are hierarchically contained within it.

HTTP: Hyper-Text Transfer Protocol

IGP: Interior Gateway Protocol

IS-IS: Intermediate System to Intermediate System protocol

NETCONF: Network Configuration Protocol

OSPF: Open Shortest Path First, a link state routing protocol

URI: Uniform Resource Identifier

ReST: Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

YANG: A data definition language for NETCONF

3. Model Structure Details

The abstract (base) network model is defined in the network.yang module. Its structure is shown in the following figure. Brackets enclose list keys, "rw" means configuration data, "ro" means operational state data, and "?" designates optional nodes.

```
module: network
   +--rw network* [network-id]
      +--rw network-id            network-id
      +--ro server-provided?      boolean
      +--rw network-types
      +--rw supporting-network* [network-ref]
      |  +--rw network-ref    leafref
      +--rw node* [node-id]
         +--rw node-id            node-id
         +--rw supporting-node* [network-ref node-ref]
            +--rw network-ref    leafref
            +--rw node-ref       leafref
```

Figure 3: The structure of the abstract (base) network model

The abstract (base) network topology model is defined by augmenting the network model defined in the network.yang module with link data defined in the network-topology.yang module. Effectively, both the network.yang module and the network-topology.yang module are used to define the abstract (base) network topology. The network-topology.yang module augments 'node' with 'termination points' and 'network' with 'links'. The structure of the network topology module is shown in the following figure. Brackets enclose list keys, "rw" means configuration data, "ro" means operational state data, and "?" designates optional nodes.

```
module: network
   +--rw network* [network-id]
      +--rw network-id            network-id
      +--ro server-provided?      boolean
      +--rw network-types
      +--rw supporting-network* [network-ref]
      |  +--rw network-ref    leafref
      +--rw node* [node-id]
      |  +--rw node-id                  node-id
      |  +--rw supporting-node* [network-ref node-ref]
      |  |  +--rw network-ref    leafref
      |  |  +--rw node-ref       leafref
      |  +--rw lnk:termination-point* [tp-id]
      |     +--rw lnk:tp-id                           tp-id
      |     +--rw lnk:supporting-termination-point*
                             [network-ref node-ref tp-ref]
      |        +--rw lnk:network-ref    leafref
      |        +--rw lnk:node-ref       leafref
      |        +--rw lnk:tp-ref         leafref
      +--rw lnk:link* [link-id]
         +--rw lnk:link-id            link-id
         +--rw lnk:source
         |  +--rw lnk:source-node    leafref
         |  +--rw lnk:source-tp?     leafref
         +--rw lnk:destination
         |  +--rw lnk:dest-node    leafref
         |  +--rw lnk:dest-tp?     leafref
         +--rw lnk:supporting-link* [network-ref link-ref]
            +--rw lnk:network-ref    leafref
            +--rw lnk:link-ref       leafref
```

Figure 4: The structure of the abstract (base) network topology model

3.1. Main building blocks

A network can contain multiple topologies. Each topology is captured in its own list entry, distinguished via a topology-id. This is captured by list "topology", contained underneath the root container for this module, "network-topology".

A topology has a certain type, such as L2, L3, OSPF or IS-IS. A topology can even have multiple types simultaneously. The type, or types, are captured underneath the container "topology-types". This container serves as container for data nodes that represent specific topology types. In this module it serves merely as an augmentation target; topology-specific modules will later introduce new data nodes to represent new topology types below this target, i.e. insert them below "topology-types" by ways of yang augmentation.

Topology types SHOULD always be represented using presence containers, not leafs of empty type. This allows to represent hierarchies of topology subtypes within the instance information. For example, an instance of an OSPF topology (which, at the same time, is a layer 3 unicast IGP topology) would contain underneath "topology-types" another container "l3-unicast-igp-topology", which in turn would contain a container "ospf-topology".

A topology can in turn be part of a hierarchy of topologies, building on top of other topologies. Any such topologies are captured in the list "underlay-topology".

Furthermore, a topology contains nodes and links, each captured in their own list.

A node has a node-id that distinguishes the node from other nodes in the list. In addition, a node has a list of termination points that are used to terminate links. An example of a termination point might be a physical or logical port or, more generally, an interface. Also, a node can map onto one or more other nodes in an underlay topology. This is captured in the list "supporting-node".

A link is identified by a link-id that uniquely identifies the link within a given topology. Links are point-to-point and unidirectional. Accordingly, a link contains a source and a destination. Both source and destination reference a corresponding node, as well as a termination point on that node. Similar to a node, a link can map onto one or more links in an underlay topology. This is captured in the list "supporting-link".

3.2. Discussion and selected design decisions

Rather than maintaining lists in separate containers, the model is kept relatively flat in terms of its containment structure. Therefore, path specifiers used to refer to specific nodes, be it in management operations or in specifications of constraints, can remain relatively compact. Of course, this means there is no separate structure in instance information that separates elements of different lists from one another. Such structure is semantically not required, although it might enhance human readability in some cases.

To minimize assumptions of what a topology might actually represent, mappings between topologies, nodes, links, and termination points are kept strictly generic. For example, no assumptions are made whether a termination point actually refers to an interface, or whether a node refers to a specific "system" or device; the model at this generic level makes no provisions for that. Any greater specifics about mappings between upper and lower layers can be captured in

augmenting modules. For example, if a termination point maps to an interface, an augmenting module can augment the termination point with a leaf that references the corresponding interface [if-config]. If a node maps to a particular device or network element, an augmenting module can augment node with a leaf that references the network element.

The model makes use of groupings, instead of simply defining data nodes "in-line". This allows to more easily include the corresponding data nodes in notifications, which then do not need to respecify each data node that is to be included. The tradeoff for this is that it makes the specification of constraints more complex, because constraints involving data nodes outside the grouping need to be specified in conjunction with a "uses" statement where the grouping is applied. This also means that constraints and XPath-statements need to specified in such a way that the navigate "down" first and select entire sets of nodes, as opposed to being able to simply specify them against individual data nodes.

The topology model includes links that are point-to-point and unidirectional. It does not directly support multipoint and bidirectional links. While this may appear as a limitation, it does keep the model simple, generic, and allows it to very easily be subjected applications that make use of graph algorithms. Bi-directional connections can be represented through pairs of unidirectional links. Multipoint networks can be represented through pseudo-nodes (similar to IS-IS, for example). By introducing hierarchies of nodes, with nodes at one level mapping onto a set of other nodes at another level, and the introducing new links for nodes at that level, topologies with connections representing non-point-to-point communication patterns can be represented.

Links are terminated by a single termination point, not sets of termination points. Connections involving multihoming or link aggregation schemes need to be represented using multiple point-to-point links, then defining a link at a higher layer that is supported by those individual links.

In a hierarchy of topologies, there are nodes mapping to nodes, links mapping to links, and termination points mapping to termination points. Some of this information is redundant. Specifically, with the link-to-links mapping known, and the termination points of each link known, maintaining separate termination point mapping information is not needed but can be derived via transitive closure. The model does provide for the option to include this information explicitly, but does not allow for it to be configured to avoid the potential to introduce (and having to validate) corresponding integrity issues.

A topology's topology types are represented using a container which contains a data node for each of its topology types. A topology can encompass several types of topology simultaneously, hence a container is used instead of a case construct, with each topology type in turn represented by a dedicated presence container itself. The reason for not simply using an empty leaf, or even simpler, do away even with the topology container and just use a leaf-list of topology-type instead, is to be able to represent "class hierarchies" of topology types, with one topology type refining the other. Topology-type specific containers are to be defined in the topology-specific modules, augmenting the topology-types container.

3.3. Open issues and items for further discussion

YANG requires data needs to be designated as either configuration or operational data, but not both, yet it is important to have all topology information, including vertical cross-topology dependencies, captured in one coherent model. In most cases topology information is discovered about a network; the topology is considered a property of the network that is reflected in the model. That said, it is conceivable that certain types of topology need to also be configurable by an application.

There are several alternatives in which this can be addressed. The alternative chosen in this draft does not restrict topology information as read-only, but includes a flag that indicates for each topology whether it should be considered as read-only or configurable by applications.

An alternative would be to designate topology list elements as read only. The read-only topology list includes each topology; it is the complete reference. In parallel a second topology list is introduced. This list serves the purpose of being able to configure topologies which are then mirrored in the read-only list. The configurable topology list adheres to the same structure and uses the same groupings as its read-only counterpart. As most data is defined in those groupings, the amount of additional definitions required will be limited. A configurable topology will thus be represented twice: once in the read-only list of all topologies, a second time in a configuration sandbox.

4. YANG Modules

4.1. Defining the Abstract Network: network.yang

```
<CODE BEGINS> file "network.yang"
module network {
  yang-version 1;
```

```
     namespace "urn:TBD:params:xml:ns:yang:nodes";
     prefix nd;

     import ietf-inet-types { prefix inet; }

     organization "TBD";
     contact
       "WILL-BE-DEFINED-LATER";
     description
       "This module defines a common base model for a collection
        of nodes in a network. Node definitions s are further used
        in network topologies and inventories.";

     revision 2014-12-11 {
       description
         "Initial revision.";
       reference "draft-clemm-i2rs-yang-network-topo-01";
     }

     typedef node-id {
       type inet:uri;
     }

     typedef network-id {
       type inet:uri;
     }

     grouping network-ref {
       leaf network-ref {
         type leafref {
           path "/network/topology-id";
         }
       }
     }

     grouping node-ref {
       uses network-ref;
       leaf node-ref {
         type leafref {
           path "/network[network-id=current()" +
                "/../network-ref]/node/node-id";
         }
       }
     }

     list network {
       key "network-id";
```

```
      leaf network-id {
        type network-id;
      }

      leaf server-provided {
        type boolean;
        config false;
      }

      container network-types {
      }

      list supporting-network {
        key "network-ref";
        leaf network-ref {
          type leafref {
            path "/network/network-id";
          }
        }
      }

      list node {
        key "node-id";
        leaf node-id {
          type node-id;
        }
        list supporting-node {
          key "network-ref node-ref";
          leaf network-ref {
            type leafref {
              path "../../../supporting-network/network-ref";
            }
          }
          leaf node-ref {
            type leafref {
            // path "/network[network-id=current()" +
            // "/../network-ref]/node/node-id";
            path "/network/node/node-id";
            }
          }
        }
      }
    }

  }
  <CODE ENDS>
```

4.2. Creating Abstract Network Topology: network-topology.yang

```
<CODE BEGINS> file "network-topology.yang"
module network-topology {
  yang-version 1;
  namespace "urn:TBD:params:xml:ns:yang:links";
  prefix lnk;

  import ietf-inet-types { prefix inet; }
  import nodes { prefix nd; }

  organization "TBD";
  contact
    "WILL-BE-DEFINED-LATER";
  description
    "This module defines a common base model for a collection of links
     connecting nodes.";

  revision 2014-12-11 {
    description
      "Initial revision.";
    reference "draft-clemm-i2rs-yang-network-topo-01";
  }

  typedef link-id {
    type inet:uri;
    description
      "An identifier for a link in a topology.
       The identifier may be opaque.
       The identifier SHOULD be chosen such that the same link in a
       real network topology will always be identified through the
       same identifier, even if the model is instantiated in separate
       datastores. An implementation MAY choose to capture semantics
       in the identifier, for example to indicate the type of link
       and/or the type of topology that the link is a part of.";
  }

  typedef tp-id {
    type inet:uri;
    description
      "An identifier for termination points on a node.
       The identifier may be opaque.
       The identifier SHOULD be chosen such that the same TP in a
       real network topology will always be identified through the
       same identifier, even if the model is instantiated in separate
       datastores. An implementation MAY choose to capture semantics
       in the identifier, for example to indicate the type of TP
       and/or the type of node and topology that the TP is a part
```

```
         of.";
    }

    grouping link-ref {
      description
        "A type for an absolute reference a link instance.
         (This type should not be used for relative references.
         In such a case, a relative path should be used instead.)";
      uses nd:network-ref;
      leaf link-ref {
        type leafref {
          path "/nd:network[nd:network-id=current()" +
               "/../nd:network-ref]/link/link-id";
        }
      }
    }

    grouping tp-ref {
      description
        "A type for an absolute reference to a termination point.
         (This type should not be used for relative references.
         In such a case, a relative path should be used instead.)";
      uses nd:node-ref;
      leaf tp-ref {
        type leafref {
          path "/nd:network[nd:network-id=current()" +
               "/../nd:network-ref]/nd:node[nd:node-id=current()" +
               "/../nd:node-ref]/termination-point/tp-id";
        }
      }
    }

    augment "/nd:network" {
      list link {
        key "link-id";
        leaf link-id {
          type link-id;
          description
            "The identifier of a link in the topology.
             A link is specific to a topology to which it belongs.";
        }
        description
          "A Network Link connects a by Local (Source) node and
          a Remote (Destination) Network Nodes via a set of the
          nodes' termination points.
          As it is possible to have several links between the same
          source and destination nodes, and as a link could
          potentially be re-homed between termination points, to
```

```
          ensure that we would always know to distinguish between
          links, every link is identified by a dedicated link
          identifier.
          Note that a link models a point-to-point link, not a
          multipoint link.
          Layering dependencies on links in underlay topologies are
          not represented as the layering information of nodes and of
          termination points is sufficient.";
        container source {
          description
            "This container holds the logical source of a particular
            link.";
          leaf source-node {
            type leafref {
              path "../../../nd:node/nd:node-id";
            }
            mandatory true;
            description
              "Source node identifier, must be in same topology.";
          }
          leaf source-tp {
            type leafref {
              path "../../../nd:node[nd:node-id=current()/.." +
                   "/source-node]/termination-point/tp-id";
            }
            description
              "Termination point within source node that terminates
              the link.";
          }
        }
        container destination {
          description
            "This container holds the logical destination of a
            particular link.";
          leaf dest-node {
            type leafref {
              path "../../../nd:node/nd:node-id";
            }
            mandatory true;
            description
              "Destination node identifier, must be in the same
              network.";
          }
          leaf dest-tp {
            type leafref {
              path "../../../nd:node[nd:node-id=current()/.." +
                   "/dest-node]/termination-point/tp-id";
            }
```

```
            description
              "Termination point within destination node that
              terminates the link.";
          }
        }
        list supporting-link {
          key "network-ref link-ref";
          description
            "Identifies the link, or links, that this link
            is dependent on.";
          leaf network-ref {
            type leafref {
              path "../../../nd:supporting-network/nd:network-ref";
            }
            description
              "This leaf identifies in which underlay topology
              supporting link is present.";
          }
          leaf link-ref {
            type leafref {
              path "/nd:network[nd:network-id=" +
                   "current()/../network-ref]/link/link-id";
            }
            description
              "This leaf identifies a link which is forms a part
              of this link's underlay. Reference loops, where
              a link identifies itself as its underlay, either
              directly or transitively, are not allowed.";
          }
        }
      }
    }

    augment "/nd:network/nd:node" {
      list termination-point {
        key "tp-id";
        description
          "A termination point can terminate a link.
          Depending on the type of topology, a termination point
          could, for example, refer to a port or an interface.";
        leaf tp-id {
          type tp-id;
          description
            "Termination point identifier.";
        }
        list supporting-termination-point {
          key "network-ref node-ref tp-ref";
          description
```

```
            "The leaf list identifies any termination points that
            the termination point is dependent on, or maps onto.
            Those termination points will themselves be contained
            in a supporting node.
            This dependency information can be inferred from
            the dependencies between links.  For this reason,
            this item is not separately configurable.  Hence no
            corresponding constraint needs to be articulated.
            The corresponding information is simply provided by the
            implementing system.";
          leaf network-ref {
            type leafref {
              path "../../../nd:supporting-node/nd:network-ref";
            }
            description
              "This leaf identifies in which topology the
              supporting termination point is present.";
          }
          leaf node-ref {
            type leafref {
              path "../../../nd:supporting-node/nd:node-ref";
            }
            description
              "This leaf identifies in which node the supporting
               termination point is present.";
          }
          leaf tp-ref {
            type leafref {
              path "/nd:network[nd:network-id=" +
                   "current()/../network-ref]/nd:node[nd:node-id=" +
                   "current()/../node-ref]/termination-point" +
                   "/tp-id";
            }
            description
              "Reference to the underlay node, must be in a
              different topology";
          }
        }
      }
    }
  }
```

<CODE ENDS>

5. Security Considerations

The transport protocol used for sending the topology data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

6. Contributors

The model presented in this paper was contributed to by more people than can be listed on the author list. Additional contributors include:

- Ken Gray, Cisco Systems
- Tom Nadeau, Brocade
- Aleksandr Zhdankin, Cisco

7. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Martin Bjorklund, Ladislav Lhotka, Andy Bierman, Carlos Pignataro, Juergen Schoenwaelder, Alia Atlas and Benoit Claise.

8. References

8.1. Normative References

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, December 1990.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, April 1998.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

8.2. Informative References

[if-config]
Bjorklund, M., "A YANG Data Model for Interface Management", I-D draft-ietf-netmod-interfaces-cfg-16, July 2013.

[restconf]
Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "RESTCONF Protocol", I-D draft-bierman-netconf-restconf-04, February 2014.

[topology-use-cases]
Medved, J., Previdi, S., Lopez, V., and S. Amante, "Topology API Use Cases", I-D draft-amante-i2rs-topology-use-cases-01, October 2013.

[yang-json]
Lhotka, L., "Modeling JSON Text with YANG", I-D draft-lhotka-etmod-yang-json-02, September 2013.

Authors' Addresses

Alexander Clemm
Cisco

EMail: alex@cisco.com

Jan Medved
Cisco

EMail: jmedved@cisco.com

Robert Varga
Cisco

EMail: rovarga@cisco.com

Tony Tkacik
Cisco

EMail: ttkacik@cisco.com

Nitin Bahadur
Bracket Computing

EMail: nitin_bahadur@yahoo.com

Hariharan Ananthakrishnan
Packet Design

EMail: hanantha@juniper.net