TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2022

T. Saad
V. Beeram
Juniper Networks
X. Liu
Volta Networks

July 12, 2021

A YANG Data Model for Network Resource Reservation Manager
draft-bestbar-teas-resmgr-yang-00

Abstract

This document defines a YANG data model for the network Resource Reservation Manager (RRM). The RRM can be deployed to manage set of network resources scoped to a node, a region of a network, a domain of the network, or globally for all resources in a network.

This model covers data for configuration, operational state, remote procedural calls pertaining to links managed by the RRM.

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 https://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 January 13, 2022.

Table of Contents

1. Introduction

YANG [RFC6020] and [RFC7950] is a data modeling language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG data models can be used as the basis of implementation for other interfaces, such as gRPC, CLI and other programmatic APIs.

This document describes YANG data model for the Resource Reservation Manager (RRM). The RRM can be deployed to manage set of network resources scoped to a node, a region of a network, a domain of the network, or globally for all resources in a network.

The RRM can acquire topological elements and their attributes from the devices using routing protocols or another suitable interface to the network devices. An aggregate view of the dynamic resource reservation state on links managed by the RRM can be downloaded to the device. The device can then disseminate the dynamic link state to the network using known means (e.g. link state protocols). The headend or Path Computation Engine (PCE) can update their topologies with current network state and use it to make further for path computations.

It is possible to deploy multiple instances of RRM to service different parts of the network. For example, a per-domain RRM may be deployed to service requests within a domain. A per-node RRM instance may be deployed to manage resources specific to a node.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC6241] and are used in this specification:

- client
- configuration data
- state data

This document also makes use of the following terminology introduced in the YANG Data Modeling Language [RFC7950]:

- augment
- data model
- data node

2.1. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in Table 1.

Prefix	YANG module	Reference
inet	ietf-inet-types	[RFC6991]
te-types	ietf-te-types	[RFC8776]
te-packet-types	ietf-te-packet-types	[RFC8776]
topo-filt	ietf-topology-filter	{{!I-D.bestbar-teas-yang-topology-filter
rt	ietf-routing	[RFC8349]
rrm	ietf-resmgr	this document

Table 1: Prefixes and corresponding YANG modules

2.2. Model Tree Diagrams

The tree diagrams extracted from the module(s) defined in this document are given in subsequent sections as per the syntax defined in [RFC8340].

3. Design Considerations

The following other design considerations are taken into account with respect data organization:

- In general, minimal elements in the model are designated as "mandatory" to allow freedom to vendors to adapt the data model to their specific product implementation.

- For optional data nodes, default values are specified when multi-vendor implementations can agree on the default behavior.

- The Network Management Datastore Architecture (NMDA) [RFC8342] addresses modeling state data for ephemeral objects. This document adopts the NMDA model for configuration and state data representation as per IETF guidelines for new IETF YANG models.

4. Network Resource Reservation Manager YANG Model

The network RRM YANG module ('ietf-resmgr') is meant to manage resource reservation on a set of resources of a network.

This includes admitting and releasing paths on specific links and nodes managed by the RRM.

4.1. Module Structure

The 'ietf-resmgr' structured hierarchically. The set of network resources managed by the RRM are organized by domain and node membership.

domains:

A YANG container that includes the list of domain resources managed by this RRM.

nodes:

A YANG container that includes the list of node resources under a specific domain that are managed by this RRM.

links:

A YANG container that includes the list of link resources under a specific node in a domain that are managed by this RRM.

path-admit:

A Remote Procedure Call (RPC) to request path admission of a specific path on a set of network resources managed by this RRM.

topology-update:

An RPC to request a addition or removal of a network element whose resources are managed by this RRM.

4.2. Tree Diagram

Figure 1 shows the tree diagram of the generic TE YANG model defined in modules 'ietf-resmgr.yang'.

```
module: ietf-resmgr
  +--rw resmgr
     +--rw external-rrms
     |  +--rw external-rrm* [external-rrm-id]
     |     +--rw external-rrm-id      inet:ip-address
     |     +--rw external-rrm-role?   enumeration
     |     +--rw topology-filter
     |        +--rw filter?       leafref
     |        +--rw filter-set?   leafref
```

```
        +--rw domains
           +--rw domain* [domain-id]
              +--rw domain-id    uint32
              +--rw nodes
                 +--rw node* [node-id]
                    +--rw node-id    inet:ip-address
                    +--rw links
                       +--rw link* [local-id remote-id]
                          +--rw local-id             inet:ip-address
                          +--rw remote-id            inet:ip-address
                          +--rw local-domain-id?     uint32
                          +--rw remote-domain-id?    uint32
                          +--rw total-bw?            uint64
                          +--rw max-reservable-bw?   uint64
                          +--rw max-link-bw?         uint64
                          +--rw link-name?           string
                          +--ro available-bw* [priority]
                          |  +--ro priority    uint8
                          |  +--ro val?        uint64
                          +--rw admission-method?    identityref
                          +--rw external-rrm
                          |  +--rw resmgr-server-address?
                          |          inet:ip-address
                          +--rw paths
                             +--rw path*
                                     [client-id tunnel-id
                                      path-instance-id multipath-id
                                      source destination]
                                +--rw client-id
                                |       string
                                +--rw source
                                |       inet:ip-address
                                +--rw destination
                                |       inet:ip-address
                                +--rw context?
                                |       string
                                +--rw tunnel-id
                                |       uint32
                                +--rw path-instance-id
                                |       uint32
                                +--rw multipath-id
                                |       uint32
                                +--rw admission-timestamp?
                                |       uint64
                                +--rw admission-bw?
                                |       uint64
                                +--rw admission-priority?
                                |       uint8
```

```
                              +--rw admission-reservation-style?
                                      identityref

     rpcs:
       +---x path-admit
       |  +---w input
       |  |  +---w action?      enumeration
       |  |  +---w path-info
       |  |     +---w client-id?            string
       |  |     +---w source?               inet:ip-address
       |  |     +---w destination?          inet:ip-address
       |  |     +---w context?              string
       |  |     +---w tunnel-id?            uint32
       |  |     +---w path-instance-id?     uint32
       |  |     +---w multipath-id?         uint32
       |  |     +---w admission-priority?   uint8
       |  |     +---w nodes
       |  |        +---w node* [node-id]
       |  |           +---w node-id      inet:ip-address
       |  |           +---w node-name?   string
       |  |           +---w links
       |  |              +---w link* [local-id remote-id]
       |  |                 +---w local-id            inet:ip-address
       |  |                 +---w remote-id           inet:ip-address
       |  |                 +---w local-domain-id?    uint32
       |  |                 +---w remote-domain-id?   uint32
       |  |                 +---w admission-bw?       uint64
       |  +--ro output
       |     +--ro result?   enumeration
       +---x topology-update
          +---w input
             +---w topology-element-type?   enumeration
             +---w action?                  enumeration
             +---w topology-elemnt-info
                +---w (element-type)
                   +--:(ne-link)
                   |  +---w local-id?           inet:ip-address
                   |  +---w remote-id?          inet:ip-address
                   |  +---w local-domain-id?    uint32
                   |  +---w remote-domain-id?   uint32
                   +--:(ne-node)
                      +---w node-id?           inet:ip-address
```

Figure 1: The RRM data model YANG tree diagram

4.3. YANG Module

The RRM YANG module 'ietf-resmgr' imports the following modules:

- ietf-yang-types and ietf-inet-types defined in [RFC6991]
- ietf-te-types defined in [RFC8776]
- ietf-routing defined in [RFC8349]
- ietf-topology-filter defined in [I-D.bestbar-teas-yang-topology-filter]

```
<CODE BEGINS> file "ietf-resmgr@2021-07-01.yang"
module ietf-resmgr {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-resmgr";
  prefix rrm;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC6991: Common YANG Data Types";
  }
  import ietf-topology-filter {
    prefix topo-filt;
    reference
      "I-D.bestbar-teas-yang-topology-filter";
  }
  import ietf-routing {
    prefix rt;
    reference
      "RFC8349: A YANG Data Model for Routing Management";
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
     Working Group.";
  contact
    "WG Web:   <http://tools.ietf.org/wg/teas/>
     WG List:  <mailto:teas@ietf.org>

     Editor:   Tarek Saad
               <mailto:tsaad@juniper.net>

     Editor:   Vishnu Pavan Beeram
               <mailto:vbeeram@juniper.net>";
  description
```

```
      "YANG data module for configuration, state, and RPCs of
       a Resource Reservation Manager.
       The model fully conforms to the Network Management
       Datastore Architecture (NMDA).

       Copyright (c) 2019 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
       (https://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.: update the date below with the date of RFC publication
    // and remove this note.

    revision 2021-07-01 {
      description
        "Initial revision";
      reference
        "RFC XXXX: A YANG data model for the Resource Reservation
         Manager.";
    }

    identity path-admission-method {
      description
        "Base identity for path admission method.";
    }

    identity path-admission-local {
      base path-admission-method;
      description
        "Indicates path admission is managed local RRM.";
    }

    identity path-admission-external {
      base path-admission-method;
      description
        "Indicates path admission is managed by external RRM.";
    }

    identity path-reservation-style {
```

```
    description
      "Base identity for reservation style.";
  }

  identity path-reservation-fixed-filter {
    base path-reservation-style;
    description
      "Fixed-Filter (FF) Style.";
    reference
      "RFC2205";
  }

  identity path-reservation-shared-explicit {
    base path-reservation-style;
    description
      "Shared Explicit (SE) Style.";
    reference
      "RFC2205";
  }

  grouping path-key {
    description
      "Grouping for leafs that identify a specific path.";
    leaf client-id {
      type string;
      description
        "A client identifier";
    }
    leaf source {
      type inet:ip-address;
      description
        "The path source address.";
    }
    leaf destination {
      type inet:ip-address;
      description
        "The path destination address.";
    }
    leaf context {
      type string;
      description
        "The path context set by the tunnel manager. For
         example, this can be the SR Candidate Path name";
    }
    leaf tunnel-id {
      type uint32;
      description
        "The tunnel ID that is shared for multiple path-instances
```

```
            belonging to the tunnel.";
        }
        leaf path-instance-id {
          type uint32;
          description
            "The path instance identifier. Multiple path instances may
             be instantiated for the same tunnel.";
        }
        leaf multipath-id {
          type uint32;
          description
            "An identifier that uniquely distinguishes the path within
             a set of multiple paths for a path instance.";
        }
      }

      grouping link-key {
        description
          "A grouping for a link key descriptor";
        leaf local-id {
          type inet:ip-address;
          description
            "Link local identifier.";
        }
        leaf remote-id {
          type inet:ip-address;
          description
            "Link remote identifier.";
        }
        leaf local-domain-id {
          type uint32;
          description
            "The local domain identifier.";
        }
        leaf remote-domain-id {
          type uint32;
          description
            "The remote domain identifier.";
        }
      }

      grouping node-key {
        description
          "Node properties.";
        leaf node-id {
          type inet:ip-address;
          description
            "The node identifier.";
```

```
      }
    }

    container resmgr {
      description
        "A container that holds all RRM information.";

      container external-rrms {
        description
          "A container for the list of external RRMs.";

        list external-rrm {
          key "external-rrm-id";
          description
            "An entry in the list of external RRMs.";

          leaf external-rrm-id {
            type inet:ip-address;
            description
              "The IP address of the external RRM managing network
               resources.";
          }
          leaf external-rrm-role {
            type enumeration {
              enum redundancy-active {
                description
                  "External RRM in active role.";
              }
              enum redundancy-stanby {
                description
                  "External RRM in standby role.";
              }
            }
            description
              "The redundancy role of the external RRM managing the
               network resources.";
          }
          container topology-filter {
            description
              "A container for the set of topology filters that
               describe network resources managed by the RRM.";
            leaf filter {
              type leafref {
                path "/rt:routing/topo-filt:topology-filters/"
                     + "topo-filt:topology-filter/topo-filt:name";
              }
              description
                "A filter that describes the set of network resources
```

```
                managed by the RRM.";
            }
            leaf filter-set {
              type leafref {
                path "/rt:routing/topo-filt:topology-filter-sets/"
                     + "topo-filt:topology-filter-set/topo-filt:name";
              }
              description
                "A filter set that describes the network resources
                 managed by the RRM.";
            }
          }
        }
      }

      container domains {
        description
          "A container for the list of managed domains.";
        list domain {
          key "domain-id";
          description
            "Represents a domain in the network.";
          leaf domain-id {
            type uint32;
            description
              "The domain identifier.";
          }
          container nodes {
            description
              "A container for the list of managed nodes.";
            list node {
              key "node-id";
              description
                "Represents a node entry in a domain.";
              uses node-key;
              // Node attributes
              container links {
                description
                  "A container for the list of managed links.";
                list link {
                  key "local-id remote-id";
                  description
                    "A resource reservation managed link entry.";
                  uses link-key;
                  // Static Link attributes
                  leaf total-bw {
                    type uint64;
                    description
```

```
        "Link total bandwidth (capacity) of this link.";
      }
      leaf max-reservable-bw {
        type uint64;
        description
          "The maximum reservable bandwidth of this link.";
      }
      leaf max-link-bw {
        type uint64;
        description
          "The maximum bandwidth of this link.";
      }
      leaf link-name {
        type string;
        description
          "The symbolic name of this link (e.g. FQDN).";
      }
      list available-bw {
        key "priority";
        config false;
        description
          "A list of available bandwidth (by priority).";
        leaf priority {
          type uint8;
          description
            "The reservation priority.";
        }
        leaf val {
          type uint64;
          description
            "Available bandwidth value at specific
             priority.";
        }
      }
      leaf admission-method {
        type identityref {
          base path-admission-method;
        }
        default "path-admission-local";
        description
          "The path admission method. By default, it is
           locally managed by the RRM.";
      }
      container external-rrm {
        when "derived-from-or-self(../admission-method, "
           + "'path-admission-external')" {
          description
            "The external RRM where the path admission is
```

```
          managed.";
      }
      description
        "The container that holds information about
         RRM external server managing path admission.";
      leaf resmgr-server-address {
        type inet:ip-address;
        description
          "The IP address of the RRM server externally
           managing link resources.";
      }
    }
    // Admitted paths
    container paths {
      description
        "A container for the list of admitted paths on a
         link.";
      list path {
        key "client-id tunnel-id path-instance-id"
          + " multipath-id source destination";
        description
          "A list of paths admitted on a link.";
        uses path-key;
        leaf admission-timestamp {
          type uint64;
          description
            "The admission timestamp.";
        }
        leaf admission-bw {
          type uint64;
          description
            "The admitted bandwidth on this link.";
        }
        leaf admission-priority {
          type uint8;
          description
            "The admission priority for this path.";
        }
        leaf admission-reservation-style {
          type identityref {
            base path-reservation-style;
          }
          default "path-reservation-shared-explicit";
          description
            "The path admssion bandwidth reservation
             style.";
        }
      }
```

```
                  }
                }
              }
            }
          }
        }
      }

      rpc path-admit {
        description
          "Input arguments for the RPC to admit/release a path on a
           specific set of resource links.";
        input {
          leaf action {
            type enumeration {
              enum add {
                description
                  "Operation add.";
              }
              enum delete {
                description
                  "Operation delete.";
              }
            }
            description
              "Admit/release RPC.";
          }
          container path-info {
            description
              "A container that includes information about the admitted
               path.";
            uses path-key;
            leaf admission-priority {
              type uint8;
              description
                "The admission priority for this path.";
            }
            container nodes {
              description
                "A container for the list of nodes that the path is being
                 admitted on.";
              list node {
                key "node-id";
                description
                  "A node that holds resources for the admitted path.";
                uses node-key;
                leaf node-name {
```

```
                type string;
                description
                  "The symbolic name of this node (e.g. FQDN).";
              }
              // Node attributes
              container links {
                description
                  "A container for the list of links used by the
                   admitted path.";
                list link {
                  key "local-id remote-id";
                  description
                    "A link that is used by the admitted path.";
                  uses link-key;
                  leaf admission-bw {
                    type uint64;
                    description
                      "The admitted bandwidth on this link.";
                  }
                }
              }
            }
          }
        }
        output {
          leaf result {
            type enumeration {
              enum unknown {
                description
                  "The RPC result is unknown.";
              }
              enum successful {
                description
                  "The RPC result is successful.";
              }
              enum rejected {
                description
                  "The RPC result is rejected.";
              }
              enum in-progress {
                description
                  "The RPC result is in-progress.";
              }
            }
            description
              "Result of admission RPC.";
          }
```

```
      }
    }

    rpc topology-update {
      description
        "Input arguments for the RPC to update the topological
         elements managed by the Resource Reservation Manager.";
      input {
        leaf topology-element-type {
          type enumeration {
            enum link {
              description
                "Topology element link type.";
            }
            enum node {
              description
                "Topology element node type.";
            }
          }
          description
            "Type of topology element.";
        }
        leaf action {
          type enumeration {
            enum add {
              description
                "Operation add.";
            }
            enum delete {
              description
                "Operation delete.";
            }
          }
          description
            "Add/delete topology element.";
        }
        container topology-elemnt-info {
          description
            "A container for the network element information.";
          choice element-type {
            mandatory true;
            description
              "The network element type.";
            case ne-link {
              uses link-key;
            }
            case ne-node {
              uses node-key;
```

```
            }
          }
        }
      }
    }
  }
<CODE ENDS>
```

Figure 2: The network RRM YANG module

5. IANA Considerations

This document registers the following URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made.

```
URI: urn:ietf:params:xml:ns:yang:ietf-resmgr
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

```
Name:       ietf-resmgr
Namespace:  urn:ietf:params:xml:ns:yang:ietf-resmgr
Prefix:     rrm
Reference:  RFCXXXX
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config)

to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

"/resmgr/topology-filters": This container and any of its encompassing data nodes defines the filter for the network resources managed by this RRM. Unauthorized access to this list could cause the RRM to ignore some network resources and could cause preemptions and disruptions in the network.

"/resmgr/domains": This container and any of its encompassing data nodes represent the set of network resources managed by this RRM. Unauthorized access to this list could cause the RRM to preempt existing path and causing disruptions to existing services in the network.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

"path-admit": using this RPC, an attacker can attempt to deplete certain network resources managed by this RRM. Also, it is possible for an attacker to preempt existing admitted paths on a set of resources by sending higher priority requests on the same set of network resources. This may affect paths that can be carrying live traffic, and hence may result in interruptions to services carried over the network.

"topology-update": using this RPC, an attacker can attempt to delete certain network resources that are already managed by this RRM. This may result in preemption of existing paths admitted on those network resources and result in interruptions to services carried over the network.

The security considerations spelled out in the YANG 1.1 specification [RFC7950] apply for this document as well.

7. Normative References

[I-D.bestbar-teas-yang-topology-filter]
Beeram, V. P. and T. Saad, "YANG Data Model for Topology Filter", draft-bestbar-teas-yang-topology-filter-00 (work in progress), July 2021.

[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>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[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>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8776] Saad, T., Gandhi, R., Liu, X., Beeram, V., and I. Bryskin, "Common YANG Data Types for Traffic Engineering", RFC 8776, DOI 10.17487/RFC8776, June 2020, <https://www.rfc-editor.org/info/rfc8776>.

Authors' Addresses

Tarek Saad
Juniper Networks

Email: tsaad@juniper.net

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com