

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 2, 2018

I. Bryskin
Huawei Technologies
X. Liu
Jabil
March 1, 2018

SF Aware TE Topology YANG Model
draft-bryskin-teas-sf-aware-topo-model-01

Abstract

This document describes a YANG data model for TE network topologies that are network service and function aware.

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 September 2, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
1.2. Tree Diagrams	3
1.3. Prefixes in Data Node Names	3
2. Modeling Considerations	3
3. Model Structure	4
4. YANG Modules	6
5. IANA Considerations	13
6. Security Considerations	14
7. References	14
7.1. Normative References	14
7.2. Informative References	15
Appendix A. Companion YANG Model for Non-NMDA Compliant Implementations	16
A.1. SF Aware TE Topology State Module	16
Appendix B. Data Examples	18
B.1. A Topology with Multiple Connected Network Functions	18
B.2. A Topology with an Encapsulated Network Service	23
Authors' Addresses	27

1. Introduction

Today a network offers to its clients far more services than just connectivity across the network. Large variety of physical, logical and/or virtual service functions, network functions and transport functions (collectively named in this document as SFs) could be allocated for and assigned to a client. As described in [I-D.bryskin-teas-use-cases-sf-aware-topo-model], there are some important use cases, in which the network needs to represent to the client SFs at the client's disposal as topological elements in relation to other elements of a topology (i.e. nodes, links, link and tunnel termination points) used by the network to describe itself to the client. Not only would such information allow for the client to auto-discover the network's SFs available for the services provisioned for the client, it would also allow for the client selecting the SFs, dual-optimizing the selection on the SF location on the network and connectivity means (e.g. TE tunnels) to inter-connect the SFs. Consequently this would give to both the network and the client powerful means for the service function chain (SFC [RFC7498] [RFC7665]) negotiation to achieve most efficient and cost effective (from the network point of view) and most optimal yet satisfying all necessary constraints of SFCs (from the client's point of view).

This document defines a YANG data model that allows service functions to be represented along with TE topology elements.

1.1. Terminology

The keywords "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].

The following terms are defined in [RFC7950] and are not redefined here:

- o augment
- o data model
- o data node

1.2. Tree Diagrams

A simplified graphical representation of the data model is presented in this document, by using the tree format defined in [I-D.ietf-netmod-yang-tree-diagrams].

1.3. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
inet	ietf-inet-types	[RFC6991]
nw	ietf-network	[I-D.ietf-i2rs-yang-network-topo]
nt	ietf-network-topology	[I-D.ietf-i2rs-yang-network-topo]
tet	ietf-te-topology	[I-D.ietf-teas-yang-te-topo]

Table 1: Prefixes and Corresponding YANG Modules

2. Modeling Considerations

The model introduced in this document is an augmentation of the TE Topology model defined in [I-D.ietf-teas-yang-te-topo]. SFs are modeled as child elements of a TE node similarly to how Link Termination Points (LTPs) and Tunnel Termination Points (TTPs) are

modeled in the TE Topology model. The SFs are defined as opaque objects identified via topology unique service-function-id's. Each SF has one or more Connection Points (CPs) identified via SF-unique sf-connection-point-id's, over which the SF could be connected to other SFs resided on the same TE node, as well as to other elements of the TE node, in particular, to the node's LTPs and/or TTPs. An interested client may use service-function-id's to look up the SFs in TOSCA or YANG data store(s) defined by [ETSI-NFV-MAN] to retrieve the details of the SFs, for example, to understand the SF's mutual substitutability.

The TE Topology model introduces a concept of Connectivity Matrix (CM), and uses the CM to describe which and at what costs a TE node's LTPs could be inter-connected internally across the TE node. The model defined in this document heavily uses the same concept to describe the SF connectivity via introducing 3 additional CMs:

1. SF2SF CM. This CM describes which pairs of SFs could be locally inter-connected, and, if yes, in which direction, via which CPs and at what costs. In other words, the SF2SF CM describes how SFs residing on the same TE node could be inter-connected into local from the TE node's perspective SFCs;
2. SF2LTP CM. This CM describes how, in which direction and at what costs the TE node's SFs could be connected to the TE node's LTPs and hence to SFs residing on neighboring TE nodes that are connected to LTPs at the remote ends of corresponding TE links;
3. SF2TTP CM. This CM describes how, in which direction and at what costs the TE node's SFs could be connected to the TE node's TTPs and hence to SFs residing on other TE nodes on the topology that could be inter-connected with the TE node in question via TE tunnels terminated by the corresponding TTPs.

In addition to SF2SF CM, the local SF chaining could be described with the help of ETSI models Virtual Links (VLs) [ETSI-NFV-MAN]. This option is especially useful when the costs of the local chaining are negligible as compared to ones of the end-to-end SFCs said local SFCs are part of.

3. Model Structure

```
module: ietf-te-topology-sf
  augment /nw:networks/nw:network/nw:network-types/tet:te-topology:
    +--rw sf!
  augment /nw:networks/nw:network/nw:node/tet:te
  /tet:te-node-attributes:
    +--rw service-function
```

```

    +--rw connectivity-matrices
    |   +--rw connectivity-matrix* [id]
    |   |   +--rw id                uint32
    |   |   +--rw from
    |   |   |   +--rw service-function-id?    string
    |   |   |   +--rw sf-connection-point-id? string
    |   |   +--rw to
    |   |   |   +--rw service-function-id?    string
    |   |   |   +--rw sf-connection-point-id? string
    |   |   +--rw enabled?                boolean
    |   |   +--rw direction?              connectivity-direction
    |   |   +--rw virtual-link-id?        string
    +--rw link-terminations
    |   +--rw link-termination* [id]
    |   |   +--rw id                uint32
    |   |   +--rw from
    |   |   |   +--rw tp-ref?    -> ../../../../../../..
    |   |   |   /nt:termination-point/tp-id
    |   |   +--rw to
    |   |   |   +--rw service-function-id?    string
    |   |   |   +--rw sf-connection-point-id? string
    |   |   +--rw enabled?                boolean
    |   |   +--rw direction?              connectivity-direction
    augment /nw:networks/nw:network/nw:node/tet:te
    /tet:information-source-entry:
    +--ro service-function
    |   +--ro connectivity-matrices
    |   |   +--ro connectivity-matrix* [id]
    |   |   |   +--ro id                uint32
    |   |   |   +--ro from
    |   |   |   |   +--ro service-function-id?    string
    |   |   |   |   +--ro sf-connection-point-id? string
    |   |   |   +--ro to
    |   |   |   |   +--ro service-function-id?    string
    |   |   |   |   +--ro sf-connection-point-id? string
    |   |   |   +--ro enabled?                boolean
    |   |   |   +--ro direction?              connectivity-direction
    |   |   |   +--ro virtual-link-id?        string
    +--ro link-terminations
    |   +--ro link-termination* [id]
    |   |   +--ro id                uint32
    |   |   +--ro from
    |   |   +--ro to
    |   |   |   +--ro service-function-id?    string
    |   |   |   +--ro sf-connection-point-id? string
    |   |   +--ro enabled?                boolean
    |   |   +--ro direction?              connectivity-direction
    augment /nw:networks/nw:network/nw:node/tet:te

```

```
/tet:tunnel-termination-point:
  +--rw service-function
    +--rw tunnel-terminations
      +--rw tunnel-termination* [id]
        +--rw id                               uint32
        +--rw service-function-id?            string
        +--rw sf-connection-point-id?         string
        +--rw enabled?                         boolean
        +--rw direction?                      connectivity-direction
```

4. YANG Modules

```
<CODE BEGINS> file "ietf-te-topology-sf@2018-02-27.yang"
module ietf-te-topology-sf {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology-sf";

  prefix "tet-sf";

  import ietf-network {
    prefix "nw";
  }

  import ietf-network-topology {
    prefix "nt";
  }

  import ietf-te-topology {
    prefix "tet";
  }

  organization
    "Traffic Engineering Architecture and Signaling (TEAS)
     Working Group";

  contact
    "WG Web:    <http://tools.ietf.org/wg/teas/>
    WG List:    <mailto:teas@ietf.org>

    Editors:    Igor Bryskin
                 <mailto:Igor.Bryskin@huawei.com>

                 Xufeng Liu
                 <mailto:Xufeng_Liu@jabil.com>";

  description
```

"Network service and function aware aware TE topology model.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

```
revision 2018-02-27 {
  description "Initial revision";
  reference "TBD";
}

/*
 * Typedefs
 */
typedef connectivity-direction {
  type enumeration {
    enum "to" {
      description
        "The direction is uni-directional, towards the 'to'
        entity direction.";
    }
    enum "from" {
      description
        "The direction is uni-directional, from the 'to'
        entity direction.";
    }
    enum "bidir" {
      description
        "The direction is bi-directional.";
    }
  }
}
description
  "A type used to indicates whether a connectivity is
  uni-directional, or bi-directional. If the relation is
  uni-directional, the value of this type indicates the
  direction.";
} // connectivity-direction

/*
 * Groupings
 */
grouping service-function-node-augmentation {
```

```
description
  "Augmenting a TE node to be network service and function
  aware.";
container service-function {
  description
    "Containing attributes related to network services and
    network functions";
  container connectivity-matrices {
    description
      "Connectivity relations between network services/functions
      on a TE node, which can be either abstract or physical.";
    reference
      "ETSI GS NFV-MAN 01: Network Functions Virtualisation
      (NFV); Management and Orchestration.
      RFC7665: Service Function Chaining (SFC) Architecture.";
    list connectivity-matrix {
      key "id";
      description
        "Represents the connectivity relations between network
        services/functions on a TE node.";
      leaf id {
        type uint32;
        description "Identifies the connectivity-matrix entry.";
      }
    }
    container from {
      description
        "Reference to the source network service or
        network function.";
      leaf service-function-id {
        type string;
        description
          "Reference to a network service or a network
          function.";
      }
    }
    leaf sf-connection-point-id {
      type string;
      description
        "Reference to a connection point on a network
        service or a network function.";
    }
  } // from
  container to {
    description
      "Reference to the destination network service or
      network function.";
    leaf service-function-id {
      type string;
```



```
        description
          "Reference to a network service or a network
           function.";
      }
      leaf sf-connection-point-id {
        type string;
        description
          "Reference to a connection point on a network
           service or a network function.";
      }
    } // to
    leaf enabled {
      type boolean;
      description
        "'true' if this connectivity entry is enabled.";
    }
    leaf direction {
      type connectivity-direction;
      description
        "Indicates whether this connectivity is
         uni-directional, or bi-directional. If the
         relation is uni-directional, the value of
         this leaf indicates the direction.";
    }
    leaf virtual-link-id {
      type string;
      description
        "Reference to a virtual link that models this
         connectivity relation in the network function
         model.";
    }
  } // connectivity-matrix
} // connectivity-matrices

container link-terminations {
  description
    "Connectivity relations between network services/functions
     and link termination points on a TE node, which can be
     either abstract or physical.";
  reference
    "ETSI GS NFV-MAN 01: Network Functions Virtualisation
     (NFV); Management and Orchestration.
     RFC7665: Service Function Chaining (SFC) Architecture.";
  list link-termination {
    key "id";
    description
      "Each entry of the list represents the connectivity
       relation between a network service/function and
```

```
        a link termination point on a TE node.";
    leaf id {
        type uint32;
        description "Identifies the termination entry.";
    }

    container from {
        description
            "Reference to the link termination point.";
    } // from
    container to {
        description
            "Reference to the network service or network
            function.";
        leaf service-function-id {
            type string;
            description
                "Reference to a network service or a network
                function.";
        }
        leaf sf-connection-point-id {
            type string;
            description
                "Reference to a connection point on a network
                service or a network function.";
        }
    } // to
    leaf enabled {
        type boolean;
        description
            "'true' if this connectivity entry is enabled.";
    }
    leaf direction {
        type connectivity-direction;
        description
            "Indicates whether this connectivity is
            uni-directional, or bi-directional. If the
            relation is uni-directional, the value of
            this leaf indicates the direction.";
    }
    } // link-termination
}
}
} // service-function-node-augmentation

grouping service-function-ttp-augmentation {
    description
        "Augmenting a tunnel termination point to be network service
```

```
    aware.";
  container service-function {
    description
      "Containing attributes related to network services and
      network functions";
    container tunnel-terminations {
      description
        "Connectivity relations between network services/functions
        and tunnel termination points on a TE node, which can be
        either abstract or physical.";
      reference
        "ETSI GS NFV-MAN 01: Network Functions Virtualisation
        (NFV); Management and Orchestration.
        RFC7665: Service Function Chaining (SFC) Architecture.";
      list tunnel-termination {
        key "id";
        description
          "Each entry of the list represents the connectivity
          relation between a network service/function and
          a tunnel termination point on a TE node.";
        leaf id {
          type uint32;
          description "Identifies the termination entry.";
        }

        leaf service-function-id {
          type string;
          description
            "Reference to a network service or a network
            function.";
        }

        leaf sf-connection-point-id {
          type string;
          description
            "Reference to a connection point on a network
            service or a network function.";
        }

        leaf enabled {
          type boolean;
          description
            "'true' if this connectivity entry is enabled.";
        }

        leaf direction {
          type connectivity-direction;
          description
            "Indicates whether this connectivity is
            uni-directional, or bi-directional. If the
            relation is uni-directional, the value of
```

```

        this leaf indicates the direction.";
    }
  } // link-termination
}
} // service-function-ttp-augmentation

grouping sf-topology-type {
  description
    "Identifies the SF aware TE topology type.";
  container sf {
    presence "Indicates that the TE topology is SF aware.";
    description
      "Its presence identifies that the TE topology is SF aware.";
  }
} // sf-topology-type

/*
 * Augmentations
 */
/* Augmentations to network-types/te-topology */
augment "/nw:networks/nw:network/nw:network-types/"
  + "tet:te-topology" {
  description
    "Defines the SF aware TE topology type.";
  uses sf-topology-type;
}

/* Augmentations to te-node-attributes */
augment "/nw:networks/nw:network/nw:node/tet:te/"
  + "tet:te-node-attributes" {
  description
    "Parameters for SF aware TE topology.";
  uses service-function-node-augmentation;
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
  + "tet:information-source-entry" {
  description
    "Parameters for SF aware TE topology.";
  uses service-function-node-augmentation;
}

/* Augmentations to tunnel-termination-point */
augment "/nw:networks/nw:network/nw:node/tet:te/"
  + "tet:tunnel-termination-point" {
  description
    "Parameters for SF aware TE topology.";
}

```

```

    uses service-function-ttp-augmentation;
  }

  /* Augmentations to connectivity-matrix */
  augment "/nw:networks/nw:network/nw:node/tet:te/"
    + "tet:te-node-attributes/tet-sf:service-function/"
    + "tet-sf:link-terminations/tet-sf:link-termination/"
    + "tet-sf:from" {
    description
      "Add reference to the link termination point.
       This portion cannot be shared with the state module.";
    leaf tp-ref {
      type leafref {
        path "../..//../..//../..//nt:termination-point/"
          + "nt:tp-id";
      }
      description
        "Reference to the link termination point.";
    }
  }
}
}
<CODE ENDS>

```

5. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

```

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

```

```

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

```

This document registers the following YANG modules in the YANG Module Names registry [RFC7950]:

```
-----  
name:          ietf-te-topology-sf  
namespace:     urn:ietf:params:xml:ns:yang:ietf-te-topology-packet  
prefix:        tet-sf  
reference:     RFC XXXX  
-----
```

```
-----  
name:          ietf-te-topology-sf-state  
namespace:     urn:ietf:params:xml:ns:yang:ietf-te-topology-packet-state  
prefix:        tet-sf-s  
reference:     RFC XXXX  
-----
```

6. Security Considerations

The configuration, state, action and notification data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241]. The data-model by itself does not create any security implications. The security considerations for the NETCONF protocol are applicable. The NETCONF protocol used for sending the data supports authentication and encryption.

7. References

7.1. Normative References

- [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>>.
- [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>>.
- [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>>.

[ETSI-NFV-MAN]

ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration", ETSI GS NFV-MAN 001 V1.1.1, December 2014.

[I-D.bryskin-teas-use-cases-sf-aware-topo-model]

Bryskin, I., Liu, X., Guichard, J., Lee, Y., Contreras, L., and D. Ceccarelli, "Use Cases for SF Aware Topology Models", draft-bryskin-teas-use-cases-sf-aware-topo-model-01 (work in progress), October 2017.

[I-D.ietf-i2rs-yang-network-topo]

Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-20 (work in progress), December 2017.

[I-D.ietf-teas-yang-te-topo]

Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-14 (work in progress), February 2018.

[I-D.ietf-netmod-revised-datastores]

Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-10 (work in progress), January 2018.

7.2. Informative References

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <<https://www.rfc-editor.org/info/rfc7498>>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.

[I-D.ietf-netmod-yang-tree-diagrams]

Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

Appendix A. Companion YANG Model for Non-NMDA Compliant Implementations

The YANG module `ietf-te-topology-sf` defined in this document is designed to be used in conjunction with implementations that support the Network Management Datastore Architecture (NMDA) defined in [I-D.ietf-netmod-revised-datastores]. In order to allow implementations to use the model even in cases when NMDA is not supported, the following companion module, `ietf-te-topology-sf-state`, is defined as state model, which mirrors the module `ietf-te-topology-sf` defined earlier in this document. However, all data nodes in the companion module are non-configurable, to represent the applied configuration or the derived operational states.

The companion module, `ietf-te-topology-sf-state`, is redundant and SHOULD NOT be supported by implementations that support NMDA.

As the structure of the companion module mirrors that of the cooresponding NMDA model, the YANG tree of the companion module is not depicted separately.

A.1. SF Aware TE Topology State Module

```
<CODE BEGINS> file "ietf-te-topology-sf-state@2018-02-27.yang"
module ietf-te-topology-sf-state {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology-sf-state";

  prefix "tet-sf-s";

  import ietf-te-topology-sf {
    prefix "tet-sf";
  }

  import ietf-network-state {
    prefix "nw-s";
  }

  import ietf-network-topology-state {
    prefix "nt-s";
  }

  import ietf-te-topology-state {
    prefix "tet-s";
  }

  organization
    "Traffic Engineering Architecture and Signaling (TEAS)"
}
```



```
    Working Group";

contact
  "WG Web:    <http://tools.ietf.org/wg/teas/>
  WG List:    <mailto:teas@ietf.org>

  Editors:    Igor Bryskin
               <mailto:Igor.Bryskin@huawei.com>

               Xufeng Liu
               <mailto:Xufeng_Liu@jabil.com>";

description
  "Network service and function aware aware TE topology operational
  state model for non-NMDA compliant implementations.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

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

revision 2018-02-27 {
  description "Initial revision";
  reference "TBD";
}

/*
 * Augmentations
 */
/* Augmentations to network-types/te-topology */
augment "/nw-s:networks/nw-s:network/nw-s:network-types/"
+ "tet-s:te-topology" {
  description
    "Defines the SF aware TE topology type.";
  uses tet-sf:sf-topology-type;
}

/* Augmentations to connectivity-matrix */
augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
+ "tet-s:te-node-attributes" {
  description
    "Parameters for SF aware TE topology.";
  uses tet-sf:service-function-node-augmentation;
```

```

    }

    augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
      + "tet-s:information-source-entry" {
        description
          "Parameters for SF aware TE topology.";
        uses tet-sf:service-function-node-augmentation;
      }

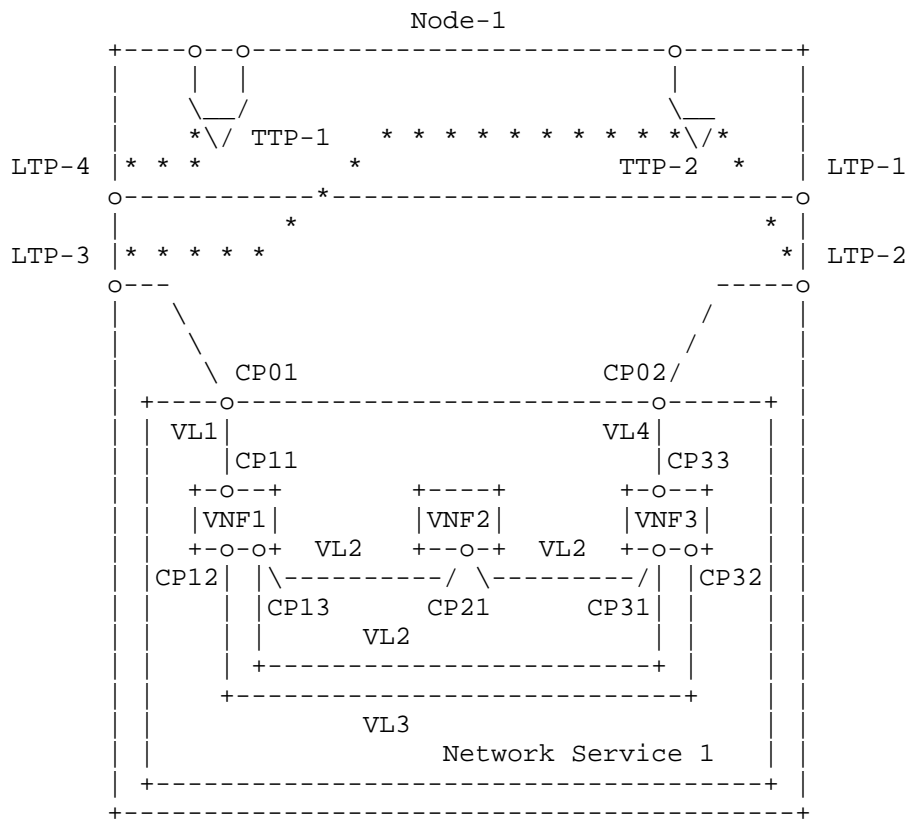
    /* Augmentations to tunnel-termination-point */
    augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
      + "tet-s:tunnel-termination-point" {
        description
          "Parameters for SF aware TE topology.";
        uses tet-sf:service-function-ttp-augmentation;
      }

    /* Augmentations to connectivity-matrix */
    augment "/nw-s:networks/nw-s:network/nw-s:node/tet-s:te/"
      + "tet-s:te-node-attributes/tet-sf-s:service-function/"
      + "tet-sf-s:link-terminations/tet-sf-s:link-termination/"
      + "tet-sf-s:from" {
        description
          "Add reference to the link termination point.
           This portion cannot be shared with the state module.";
        leaf tp-ref {
          type leafref {
            path "../.../.../.../.../nt-s:termination-point/"
              + "nt-s:tp-id";
          }
        }
        description
          "Reference to the link termination point.";
      }
    }
  }
}
<CODE ENDS>

```

Appendix B. Data Examples

B.1. A Topology with Multiple Connected Network Functions



The configuration instance data for Node-1 in the above figure could be as follows:

```
{
  "networks": {
    "network": [
      {
        "network-types": {
          "te-topology": {
            "sf": {}
          }
        },
        "network-id": "network-sf-aware",
        "provider-id": 201,
        "client-id": 300,
        "te-topology-id": "te-topology:network-sf-aware",
        "node": [
          {
            "node-id": "Node-1",
```

```
"te-node-id": "2.0.1.1",
"te": {
  "te-node-attributes": {
    "domain-id": 1,
    "is-abstract": [null],
    "connectivity-matrices": {
    },
    "service-function": {
      "connectivity-matrices": {
        "connectivity-matrix": [
          {
            "id": 10,
            "from": {
              "service-function-id": "Network Service 1",
              "sf-connection-point-id": "CP01"
            },
            "to": {
              "service-function-id": "VNF1",
              "sf-connection-point-id": "CP11"
            }
          },
          {
            "id": 13,
            "from": {
              "service-function-id": "VNF1",
              "sf-connection-point-id": "CP12"
            },
            "to": {
              "service-function-id": "VNF3",
              "sf-connection-point-id": "CP32"
            }
          },
          {
            "id": 12,
            "from": {
              "service-function-id": "VNF1",
              "sf-connection-point-id": "CP13"
            },
            "to": {
              "service-function-id": "VNF2",
              "sf-connection-point-id": "CP21"
            }
          }
        ]
      }
    }
  }
}
```

```

    },
    {
      "id": 23,
      "from": {
        "service-function-id": "VNF2",
        "sf-connection-point-id": "CP21"
      },
      "to": {
        "service-function-id": "VNF3",
        "sf-connection-point-id": "CP31"
      },
      "direction": "bidir",
      "virtual-link-id": "VL2"
    },
    {
      "id": 30,
      "from": {
        "service-function-id": "Network Service 1",
        "sf-connection-point-id": "CP02"
      },
      "to": {
        "service-function-id": "VNF3",
        "sf-connection-point-id": "CP33"
      },
      "direction": "bidir",
      "virtual-link-id": "VL4"
    }
  ]
},
"link-terminations": {
  "link-termination": [
    {
      "id": 2,
      "from": {
        "tp-ref": "LTP-2"
      },
      "to": {
        "service-function-id": "Network Service 1",
        "sf-connection-point-id": "CP02"
      },
      "direction": "bidir"
    },
    {
      "id": 3,
      "from": {
        "tp-ref": "LTP-3"
      },
      "to": {

```

```

        "service-function-id": "Network Service 1",
        "sf-connection-point-id": "CP01"
    }
    "direction": "bidir"
}
]
}
}
}
}
"tunnel-termination-point": [
{
    "tunnel-tp-id": 10001,
    "name": "TTP-1",
    "service-function-terminations": {
    },
},
{
    "tunnel-tp-id": 10002,
    "name": "TTP-2",
    "service-function-terminations": {
    }
}
]
},
"termination-point": [
{
    "tp-id": "LTP-1",
    "te-tp-id": 10001
    "te": {
        "interface-switching-capability": [
            {
                "switching-capability": "switching-l2sc",
                "encoding": "lsp-encoding-ethernet"
            }
        ]
    }
},
{
    "tp-id": "LTP-2",
    "te-tp-id": 10002
    "te": {
        "interface-switching-capability": [
            {
                "switching-capability": "switching-l2sc",
                "encoding": "lsp-encoding-ethernet"
            }
        ]
    }
}
]
}

```

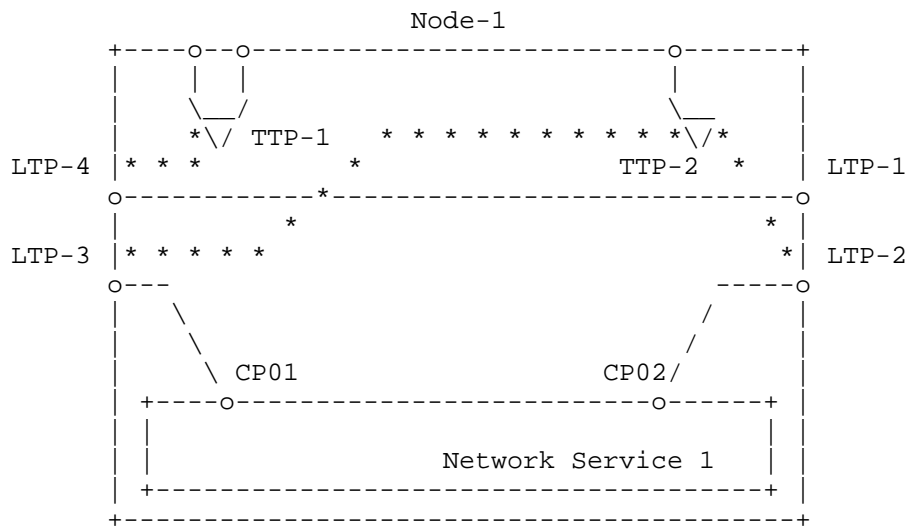
```

    },
    {
      "tp-id": "LTP-3",
      "te-tp-id": 10003
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-l2sc",
            "encoding": "lsp-encoding-ethernet"
          }
        ]
      }
    },
    {
      "tp-id": "LTP-4",
      "te-tp-id": 10004
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-l2sc",
            "encoding": "lsp-encoding-ethernet"
          }
        ]
      }
    }
  ]
}

```

B.2. A Topology with an Encapsulated Network Service

In this example, a network service consists of several interconnected network functions (NFs), and is represented by this model as an encapsulated opaque object without the details between its internals.



The configuration instance data for Node-1 in the above figure could be as follows:

```
{
  "networks": {
    "network": [
      {
        "network-types": {
          "te-topology": {
            "sf": {}
          }
        },
        "network-id": "network-sf-aware",
        "provider-id": 201,
        "client-id": 300,
        "te-topology-id": "te-topology:network-sf-aware",
        "node": [
          {
            "node-id": "Node-1",
            "te-node-id": "2.0.1.1",
            "te": {
              "te-node-attributes": {
                "domain-id": 1,
                "is-abstract": [null],
                "connectivity-matrices": {
                },
              },
              "service-function": {
                "connectivity-matrices": {
                },
              },
            }
          }
        ]
      }
    ]
  }
}
```



```

    "link-terminations": {
      "link-termination": [
        {
          "id": 2,
          "from": {
            "tp-ref": "LTP-2"
          },
          "to": {
            "service-function-id": "Network Service 1",
            "sf-connection-point-id": "CP02"
          },
          "direction": "bidir"
        },
        {
          "id": 3,
          "from": {
            "tp-ref": "LTP-3"
          },
          "to": {
            "service-function-id": "Network Service 1",
            "sf-connection-point-id": "CP01"
          },
          "direction": "bidir"
        }
      ]
    }
  },
  "tunnel-termination-point": [
    {
      "tunnel-tp-id": 10001,
      "name": "TTP-1",
      "service-function-terminations": {
      }
    },
    {
      "tunnel-tp-id": 10002,
      "name": "TTP-2",
      "service-function-terminations": {
      }
    }
  ],
  "termination-point": [
    {
      "tp-id": "LTP-1",
      "te-tp-id": 10001
      "te": {

```

```

        "interface-switching-capability": [
            {
                "switching-capability": "switching-l2sc",
                "encoding": "lsp-encoding-ethernet"
            }
        ]
    },
    {
        "tp-id": "LTP-2",
        "te-tp-id": 10002
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-l2sc",
                    "encoding": "lsp-encoding-ethernet"
                }
            ]
        }
    },
    {
        "tp-id": "LTP-3",
        "te-tp-id": 10003
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-l2sc",
                    "encoding": "lsp-encoding-ethernet"
                }
            ]
        }
    },
    {
        "tp-id": "LTP-4",
        "te-tp-id": 10004
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-l2sc",
                    "encoding": "lsp-encoding-ethernet"
                }
            ]
        }
    }
]
}

```

```
    ]  
  }  
}
```

Authors' Addresses

Igor Bryskin
Huawei Technologies

EMail: Igor.Bryskin@huawei.com

Xufeng Liu
Jabil

EMail: Xufeng_Liu@jabil.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 3, 2018

I. Bryskin
Huawei Technologies
X. Liu
Jabil
J. Guichard
Y. Lee
Huawei Technologies
L. Contreras
Telefonica
D. Ceccarelli
Ericsson
March 2, 2018

Use Cases for SF Aware Topology Models
draft-bryskin-teas-use-cases-sf-aware-topo-model-02

Abstract

This document describes some use cases that benefit from the network topology models that are service and network function aware.

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 September 3, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Exporting SF/NF Information to Network Clients and Other Network SDN Controllers	4
4. Flat End-to-end SFCs Managed on Multi-domain Networks	5
5. Managing SFCs with TE Constraints	6
6. SFC Protection and Load Balancing	7
7. Network Clock Synchronization	10
8. Client - Provider Network Slicing Interface	10
9. Dynamic Assignment of Regenerators for L0 Services	10
10. Dynamic Assignment of OAM Functions for L1 Services	12
11. SFC Abstraction and Scaling	13
12. Dynamic Compute/VM/Storage Resource Assignment	13
13. Application-aware Resource Operations and Management	14
14. IANA Considerations	15
15. Security Considerations	15
16. Acknowledgements	15
17. References	15
17.1. Normative References	15
17.2. Informative References	16
Authors' Addresses	16

1. Introduction

Normally network connectivity services are discussed as a means to inter-connect various abstract or physical network topological elements, such as ports, link termination points and nodes [I-D.ietf-teas-yang-te-topo] [I-D.ietf-teas-yang-te]. However, the connectivity services, strictly speaking, interconnect not the network topology elements per-se, rather, located on/associated with the various network and service functions [RFC7498] [RFC7665]. In many scenarios it is beneficial to decouple the service/network functions from the network topology elements hosting them, describe them in some unambiguous and identifiable way (so that it would be possible, for example, to auto-discover on the network topology service/network functions with identical or similar functionality and characteristics) and engineer the connectivity between the service/network functions, rather than between their current topological locations. The purpose of this document is to describe some use cases that could benefit from such an approach.

2. Terminology

- o Network Function (NF): A functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behaviour [ETSI-NFV-TERM]. Such functions include message router, CDN, session border controller, WAN acceleration, DPI, firewall, NAT, QoE monitor, PE router, BRAS, and radio/fixed access network nodes.
- o Network Service: Composition of Network Functions and defined by its functional and behavioural specification. The Network Service contributes to the behaviour of the higher layer service, which is characterized by at least performance, dependability, and security specifications. The end-to-end network service behaviour is the result of the combination of the individual network function behaviours as well as the behaviours of the network infrastructure composition mechanism [ETSI-NFV-TERM].
- o Service Function (SF): A function that is responsible for specific treatment of received packets. A service function can act at various layers of a protocol stack (e.g., at the network layer or other OSI layers). As a logical component, a service function can be realized as a virtual element or be embedded in a physical network element. One or more service functions can be embedded in the same network element. Multiple occurrences of the service function can exist in the same administrative domain. A non-exhaustive list of service functions includes: firewalls, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, NAT44 [RFC3022], NAT64 [RFC6146], HTTP header enrichment functions, and TCP optimizers. The generic term "L4-L7 services" is often used to describe many service functions [RFC7498].
- o Service Function Chain (SFC): A service function chain defines an ordered or partially ordered set of abstract service functions and ordering constraints that must be applied to packets, frames, and/or flows selected as a result of classification. An example of an abstract service function is a firewall. The implied order may not be a linear progression as the architecture allows for SFCs that copy to more than one branch, and also allows for cases where there is flexibility in the order in which service functions need to be applied. The term "service chain" is often used as shorthand for "service function chain" [RFC7498].
- o Connectivity Service: Any service between layer 0 and layer 3 aiming at delivering traffic among two or more end customer edge nodes connected to provider edge nodes. Examples include L3VPN, L2VPN etc.

- o Link Termination Point (LTP): A conceptual point of connection of a TE node to one of the TE links, terminated by the TE node. Cardinality between an LTP and the associated TE link is 1:0..1 [I-D.ietf-teas-yang-te-topo].
 - o Tunnel Termination Point (TTP): An element of TE topology representing one or several of potential transport service termination points (i.e. service client adaptation points such as WDM/OCh transponder). TTP is associated with (hosted by) exactly one TE node. TTP is assigned with the TE node scope unique ID. Depending on the TE node's internal constraints, a given TTP hosted by the TE node could be accessed via one, several or all TE links terminated by the TE node [I-D.ietf-teas-yang-te-topo].
3. Exporting SF/NF Information to Network Clients and Other Network SDN Controllers

In the context of Service Function Chain (SFC) orchestration one existing problem is that there is no way to formally describe a Service or Network Function in a standard way (recognizable/understood by a third party) as a resource of a network topology node.

One implication of this is that there is no way for the orchestrator to give a network client even a ball-park idea as to which network's SFs/NFs are available for the client's use/control and where they are located in the network even in terms of abstract topologies/virtual networks configured and managed specifically for the client. Consequently, the client has no say on how the SFCs provided for the client by the network should be set up and managed (which SFs are to be used and how they should be chained together, optimized, manipulated, protected, etc.).

Likewise, there is no way for the orchestrator to export SF/NF information to other network controllers. The SFC orchestrator may serve, for example, a higher level controller (such as Network Slicing Orchestrator), with the latter wanting at least some level of control as to which SFs/NFs it wants on its SFCs and how the Service Function Paths (SFPs) are to be routed and provisioned, especially, if it uses services of more than one SFC orchestrator.

The issue of exporting of SF/NF information could be addressed by defining a model, in which formally described/recognizable SF/NF instances are presented as topological elements, for example, hosted by TE, L3 or L2 topology nodes (see Figure 1). The model could describe whether, how and at what costs the SFs/NFs hosted by a given node could be chained together, how these intra-node SFCs could be connected to the node's Service Function Forwarders (SFFs, entities

dealing with SFC NSHs and metadata), and how the SFFs could be connected to the node's Tunnel and Link Termination Points (TTPs and LTPs) to chain the intra-node SFCs across the network topology.

The figure is available in the PDF format.

Figure 1: SF/NF aware TE topology

4. Flat End-to-end SFCs Managed on Multi-domain Networks

SFCs may span multiple administrative domains, each of which controlled by a separate SFC controller. The usual solution for such a scenario is the Hierarchical SFCs (H-SFCs), in which the higher level orchestrator controls only SFs located on domain border nodes. Said higher level SFs are chained together into higher level SFCs via lower level (intra-domain) SFCs provisioned and controlled independently by respective domain controllers. The decision as to which higher level SFCs are connected to which lower level SFCs is driven by packet re-classification every time the packet enters a given domain. Said packet re-classification is a very time-consuming operation. Furthermore, the independent nature of higher and lower level SFC control is prone to configuration errors, which may lead to long lasting loops and congestions. It is highly desirable to be able to set up and manage SFCs spanning multiple domains in a flat way as far as the data plane is concerned (i.e. with a single packet classification at the ingress into the multi-domain network but without re-classifications on domain ingress nodes).

One way to achieve this is to have the domain controllers expose SF/NF-aware topologies, and have the higher level orchestrator operate on the network-wide topology, the product of merging of the topologies catered by the domain controllers. This is similar in spirit to setting up, coordinating and managing the transport connectivity (TE tunnels) on a multi-domain multi-vendor transport network.

5. Managing SFCs with TE Constraints

Some SFCs require per SFC link/element and end-to-end TE constraints (bandwidth, delay/jitter, fate sharing/diversity. etc.). Said constraints could be ensured via carrying SFPs inside overlays that are traffic engineered with the constraints in mind. A good analogy would be orchestrating delay constrained L3 VPNs. One way to support such L3 VPNs is to carry MPLS LSPs interconnecting per-VPN VRFs inside delay constrained TE tunnels interconnecting the PEs hosting the VRFs.

—

Figure 2: L3 VPN with delay constraints

Planning, computing and provisioning of TE overlays to constrain arbitrary SFCs, especially those that span multiple administrative domains with each domain controlled by a separate controller, is a very difficult challenge. Currently it is addressed by pre-provisioning on the network of multiple TE tunnels with various TE characteristics, and "nailing down" SFs/NFs to "strategic" locations (e.g. nodes terminating many of such tunnels) in a hope that an adequate set of tunnels could be found to carry the SFP of a given TE-constrained SFC. Such an approach is especially awkward in the case when some or all of the SFs/NFs are VNFs (i.e. could be instantiated at multiple network locations).

SF/NF-aware TE topology model in combination with TE tunnel model will allow for the network orchestrator (or a client controller) to compute, set up and manipulate the TE overlays in the form of TE tunnel chains (see Figure 3).

Said chains could be dual-optimized compromising on optimal SF/NF locations with optimal TE tunnels interconnecting them. The TE tunnel chains (carrying multiple similarly constrained SFPs) could be adequately constrained both at individual TE tunnel level and at the chain end-to-end level.

—

Figure 3: SFC with TE constraints

6. SFC Protection and Load Balancing

Currently the combination of TE topology & tunnel models offers to a network controller various capabilities to recover an individual TE tunnel from network failures occurred on one or more network links or transit nodes on the TE paths taken by the TE tunnel's connection(s). However, there is no simple way to recover a TE tunnel from a failure affecting its source or destination node. SF/NF-aware TE topology model can decouple the association of a given SF/NF with its location on the network topology by presenting multiple, identifiable as mutually substitutable SFs/NFs hosted by different TE topology nodes. So, for example, if it is detected that a given TE tunnel destination node is malfunctioning or has gone out of service, the TE tunnel

could be re-routed to terminate on a different node hosting functionally the same SFs/NFs as ones hosted by the failed node (see Figures 6).

This is in line with the ACTN edge migration and function mobility concepts [I-D.ietf-teas-actn-framework]. It is important to note that the described strategy works much better for the stateless SFs/NFs. This is because getting the alternative stateful SFs/NFs into the same respective states as the current (i.e. active, affected by failure) are is a very difficult challenge.

—

Figure 4: SFC recovery: SF2 on node NE1 fails

At the SFC level the SF/NF-aware TE topology model can offer SFC dynamic restoration capabilities against failed/malfunctioning SFs/NFs by identifying and provisioning detours to a TE tunnel chain, so that it starts carrying the SFC's SFPs towards healthy SFs/NFs that are functionally the same as the failed ones. Furthermore, multiple parallel TE tunnel chains could be pre-provisioned for the purpose of SFC load balancing and end-to-end protection. In the latter case said parallel TE tunnel chains could be placed to be sufficiently disjoint from each other.

—

Figure 5: SFC recovery: SFC SF1-SF2-SF6 is recovered after SF2 on node N1 has failed

—

Figure 6: SFC recovery: SFC SF1-SF2-SF6 is recovered after node N1 has failed

7. Network Clock Synchronization

Many current and future network applications (including 5g and IoT applications) require very accurate time services (PTP level, ns resolution). One way to implement the adequate network clock synchronization for such services is via describing network clocks as NFs on an NF-aware TE topology optimized to have best possible delay variation characteristics. Because such a topology will contain delay/delay variation metrics of topology links and node cross-connects, as well as costs in terms of delay/delay variation of connecting clocks to hosting them node link and tunnel termination points, it will be possible to dynamically select and provision bi-directional time-constrained deterministic paths or trees connecting clocks (e.g. grand master and boundary clocks) for the purpose of exchange of clock synchronization information. Note that network clock aware TE topologies separately provided by domain controllers will enable multi-domain network orchestrator to set up and manipulate the clock synchronization paths/trees spanning multiple network domains.

8. Client - Provider Network Slicing Interface

3GPP defines network slice as "a set of network functions and the resources for these network functions which are arranged and configured, forming a complete logical network to meet certain network characteristics" [I-D.defoy-netslices-3gpp-network-slicing] [_3GPP.28.801]. Network slice could be also defined as a logical partition of a provider's network that is owned and managed by a tenant. SF/NF-aware TE topology model has a potential to support a very important interface between network slicing clients and providers because, on the one hand, the model can describe holistically and hierarchically the client's requirements and preferences with respect to a network slice functional, topological and traffic engineering aspects, as well as of the degree of resource separation/ sharing between the slices, thus allowing for the client (up to agreed upon extent) to dynamically (re-)configure the slice or (re-)schedule said (re-)configurations in time, while, on the other hand, allowing for the provider to convey to the client the slice's operational state information and telemetry the client has expressed interest in.

9. Dynamic Assignment of Regenerators for L0 Services

On large optical networks, some of provided to their clients L0 services could not be provisioned as single OCh trails, rather, as chains of such trails interconnected via regenerators, such as 3R regenerators. Current practice of the provisioning of such services requires configuration of explicit paths (EROs) describing identity

and location of regenerators to be used. A solution is highly desirable that could:

- o Identify such services based, for example, on optical impairment computations;
- o Assign adequate for the services regenerators dynamically out of the regenerators that are grouped together in pools and strategically scattered over the network topology nodes;
- o Compute and provision supporting the services chains of optical trails interconnected via so selected regenerators, optimizing the chains to use minimal number of regenerators, their optimal locations, as well as optimality of optical paths interconnecting them;
- o Ensure recovery of such chains from any failures that could happen on links, nodes or regenerators along the chain path.

NF-aware TE topology model (in this case L1 NF-aware L0 topology model) is just the model that could provide a network controller (or even a client controller operating on abstract NF-aware topologies provided by the network) to realize described above computations and orchestrate the service provisioning and network failure recovery operations (see Figure 7).

—

Figure 7: Optical tunnel as TE-constrained SFC of 3R regenerators.
Red trail (not regenerated) is not optically reachable, but blue
trail (twice regenerated) is

10. Dynamic Assignment of OAM Functions for L1 Services

OAM functionality is normally managed by configuring and manipulating TCM/MEP functions on network ports terminating connections or their segments over which OAM operations, such as performance monitoring, are required to be performed. In some layer networks (e.g. Ethernet) said TCMS/MEPs could be configured on any network ports. In others (e.g. OTN/ODUk) the TCMS/MEPs could be configured on some (but not all network ports) due to the fact that the OAM functionality (i.e. recognizing and processing of OAM messages, supporting OAM protocols and FSMs) requires in these layer networks certain support in the data plane, which is not available on all network nodes. This makes TCMS/MEPs good candidates to be modeled as NFs. This also makes TCM/MEP aware topology model a good basis for placing dynamically an ODUk connection to pass through optimal OAM locations without mandating the client to specify said locations explicitly.

—

Figure 8: Compute/storage resource aware topology

11. SFC Abstraction and Scaling

SF/NF-aware topology may contain information on native SFs/NFs (i.e. SFs/NFs as known to the provider itself) and/or abstract SFs/NFs (i.e. logical/macro SFs/NFs representing one or more SFCs each made of native and/or lower level abstract SFs/NFs). As in the case of abstracting topology nodes, abstracting SFs/NFs is hierarchical in nature - the higher level of SF/NF-aware topology, the "larger" abstract SFs/NFs are, i.e. the larger data plane SFCs they represent. This allows for managing large scale networks with great number of SFs/NFs (such as Data Center interconnects) in a hierarchical, highly scalable manner resulting in control of very large number of flat in the data plane SFCs that span multiple domains.

12. Dynamic Compute/VM/Storage Resource Assignment

In a distributed data center network, virtual machines for compute resources may need to be dynamically re-allocated due to various reasons such as DCI network failure, compute resource load balancing, etc. In many cases, the DCI connectivity for the source and the destination is not predetermined. There may be a pool of sources and a pool of destination data centers associated with re-allocation of compute/VM/storage resources. There is no good mechanism to date to capture this dynamicity nature of compute/VM/storage resource reallocation. Generic Compute/VM/Storage resources can be described and announced as a SF, where a DC hosting these resources can be modeled as an abstract node. Topology interconnecting these abstract nodes (DCs) in general is of multi-domain nature. Thus, SF-aware topology model can facilitate a joint optimization of TE network resources and Compute/VM/Storage resources and solve Compute/VM/Storage mobility problem within and between DCs (see Figure 8).

13. Application-aware Resource Operations and Management

Application stratum is the functional grouping which encompasses application resources and the control and management of these resources. These application resources are used along with network services to provide an application service to clients/end-users. Application resources are non-network resources critical to achieving the application service functionality. Examples of application resources include: caches, mirrors, application specific servers, content, large data sets, and computing power. Application service is a networked application offered to a variety of clients (e.g., server backup, VM migration, video cache, virtual network on-demand, 5G network slicing, etc.). The application servers that host these application resources can be modeled as an abstract node. There may be a variety of server types depending on the resources they host. Figure 9 shows one example application aware topology for video cache server distribution.

—

Figure 9: Application aware topology

14. IANA Considerations

This document has no actions for IANA.

15. Security Considerations

This document does not define networking protocols and data, hence is not directly responsible for security risks.

16. Acknowledgements

The authors would like to thank Maarten Vissers, Joel Halpern, and Greg Mirsky for their helpful comments and valuable contributions.

17. References

17.1. Normative References

- [RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <<https://www.rfc-editor.org/info/rfc7498>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [ETSI-NFV-TERM] ETSI, "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV", ETSI GS NFV 003 V1.2.1, December 2014.
- [I-D.ietf-teas-yang-te-topo] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-14 (work in progress), February 2018.
- [I-D.ietf-teas-yang-te] Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-12 (work in progress), February 2018.

17.2. Informative References

- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, DOI 10.17487/RFC3022, January 2001, <<https://www.rfc-editor.org/info/rfc3022>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [I-D.ietf-sfc-hierarchical]
Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", draft-ietf-sfc-hierarchical-07 (work in progress), February 2018.
- [I-D.defoy-netslices-3gpp-network-slicing]
Foy, X. and A. Rahman, "Network Slicing - 3GPP Use Case", draft-defoy-netslices-3gpp-network-slicing-02 (work in progress), October 2017.
- [_3GPP.28.801]
3GPP, "Study on management and orchestration of network slicing for next generation network", 3GPP TR 28.801 V2.0.0, September 2017, <<http://www.3gpp.org/ftp/Specs/html-info/28801.htm>>.
- [I-D.ietf-teas-actn-framework]
Ceccarelli, D. and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework-11 (work in progress), October 2017.

Authors' Addresses

Igor Bryskin
Huawei Technologies

EMail: Igor.Bryskin@huawei.com

Xufeng Liu
Jabil

EMail: Xufeng_Liu@jabil.com

Jim Guichard
Huawei Technologies

EMail: james.n.guichard@huawei.com

Young Lee
Huawei Technologies

EMail: leeyoung@huawei.com

Luis Miguel Contreras Murillo
Telefonica

EMail: luismiguel.contrerasmurillo@telefonica.com

Daniele Ceccarelli
Ericsson

EMail: daniele.ceccarelli@ericsson.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 6, 2018

T. Eckert
Huawei
Mar 5, 2018

Framework for Traffic Engineering with BIER-TE forwarding (Bit Index
Explicit Replication with Traffic Engineering)
draft-eckert-teas-bier-te-framework-00

Abstract

BIER-TE is an application-state free, (loose) source routed multicast forwarding method where every hop and destination is identified via bits in a bitstring of the data packets. It is described in [I-D.ietf-bier-te-arch]. BIER-TE is a variant of [RFC8279] in support of such explicit path engineering.

This document describes the traffic engineering control framework for use with the BIER-TE forwarding plane: How to enable the ability to calculate paths and integrate this forwarding plane into an overall TE solution.

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 September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction and Overview	2
2. BIER-TE Topology management	6
2.1. Operational model	6
2.2. BIER-TE topology model	7
2.3. Consistency checking	10
2.4. Auto-configuration	11
3. Flow Management	12
3.1. Operational / Architectural Models	12
3.1.1. Overprovisioning	13
3.1.2. PCEC	13
3.1.2.1. per-flow QoS - policer/shaper/EF	14
3.1.2.2. DiffServ QoS	15
3.2. BIER-TE flow model	15
4. Security Considerations	17
5. IANA Considerations	17
6. Acknowledgements	17
7. Change log [RFC Editor: Please remove]	17
8. References	18
Author's Address	19

1. Introduction and Overview

This document proposes a framework and abstract data model for the control plane of BIER-TE as defined in [I-D.ietf-bier-te-arch] (BIER-TE-ARCH). That document primarily defines the forwarding plane and provides some example scenarios how to use it.

BIER-TE is a forwarding plane derived from BIER ([RFC8279]) in which the destinations of packets are bits in a bitstring. Every bit indicates a destination (BFER - BIER Forwarding Exit Router) and an IGP is used to flood those "bit addresses" so hops along the path from sender (BFIR - BIER Forwarding Ingress Router) through intermediate nodes (BFR) can calculate the shortest path for each destination (bit) and simply copy the received packet to every interface to one or more bits set in the packet.

In BIER-TE, shortest path calculation is replaced by bits of the bitstring indicating intermediate hops and pre-populated forwarding tables (BIFT - Bit Index Forwarding Tables) on every BFR indicating

those bits. In the simplest case, every interface on a BFR has a unique bit assigned to it, and the BIFT of only that BFR will have in its BIFT for this bit an adjacency entry indicating that interface. This ultimately allows to indicate any sub-graph of the network topology as a bitstring and hop-by-hop perform the necessary forwarding/replication for a packet with such a bitstring. More complex semantics of bits are used to help saving bits. A typical bitstring size supportable is 256 bits, the original BIER specification allows up to 1024 bits. BIER-TE may be specifically interesting for typically smaller topologies such as often encountered in DetNet scenarios, or else through intelligent allocating and saving of bits for larger topologies, some of which is exemplified in BIER-TE-ARCH.

One can compare BIER-TE in function to Segment Routing in so far that it attempts to be as much as possible a per-packet "source-routed" (for lack of better term) forwarding paradigm without per-application/flow state in the network. Whereas SR primarily supports simple sequential paths indicated as a sequence of SIDs, in BIER-TE, the bitstring indicate a directed and acyclic graphs (DAG) - with replications. BIER-TE can also be combined with SR and then bits in the bitstring are only required for the nodes (BFR) where replication is desired, and the paths between any two such replication nodes could be SIDs or stack of SIDs that are selected by assigning bits to them (routed adjacencies in the BIER-TE terminology).

In BIER-TE-ARCH, the control plane is not considered. In its place, a theoretical BIER-TE Controller Host uses unspecified signaling to control the setup of the BIER-TE forwarding-plane end to end (all bits/adjacencies in all BFR BIFTs) and during the lifecycle of network device install through the determination of paths for specific traffic and changes to the topology. This document expands and refines this simplistic model and intends to serve as the framework for follow-up protocol and data model specification work.

The core forwarding documents relevant to this document are as follows:

- o [RFC8279] (BIER-ARCH): as summarized above.
- o [RFC8296] (BIER-ENCAP): The encapsulation for BIER packets using MPLS or non-MPLS networks underneath.
- o [I-D.ietf-bier-te-arch] (BIER-TE-ARCH): as summarized above.
- o [I-D.thubert-bier-replication-elimination] (BIER-EF-OAM): Extends the BIER-TE forwarding from BIER-TE-ARCH to support the Elimination Function (EF) and an OAM function. The Elimination

Function is a term from DetNets resilience architecture: Multiple copies of traffic flows are carried across disjoint path, merged in a BFR running the EF and duplicates are eliminated on that BFR based on recognizing duplicate sequence numbers. Engineered multiple transmission paths are a key reason to leverage BIER-TE.

- o [I-D.huang-bier-te-encapsulation] (BIER-TE-ENCAP): Proposed encapsulation based on an extension of BIER-ENCAP. Identifies whether the packet expects to use a BIER or BIER-TE BIFT. Also adds a control-word in support of (optional) elimination function (EF) and interprets the pre-existing BFIR-ID and entropy fields as a flow-id.
- o [I-D.eckert-bier-te-frr] (BIER-TE-FRR): This document describes protections methods applicable to BIER-TE. 1:1 / end-to-end path protection is referenced in this document in the context of DetNet style PREF path protection. The options not discussed yet (TBD) in this document are link protection tunnels (such as used in RSVP-TE as well) and the novel BIER-TE specific protection method, in which nodes modify the bitstring upon local discovery of a failure.

The relevant routing underlay documents are as follows:

- o [I-D.ietf-bier-isis-extensions] (BIER-ISIS), [I-D.ietf-bier-ospf-bier-extensions] (BIER-OSPF): The BIER-ISIS and BIER-OSPF documents describe extensions to those two IGPs in support of BIER. Effectively, every BFR announces the <SD,SI-range> BIFTs it is configured for, the MT-ID (IGP Multitopology-ID) they are using, and the BFR-ID it has in each SD (none if it does not need to operate as a BFER). For MPLS encapsulation, the base label for every SD is announced as well as the SI-range (one label per <SD,SI> is used).
- o There is currently no document describing IGP extensions for BIER-TE, but the goal is to define those based, using the proposals made in this framework, and as feasible re-using and/or amending those existing BIER IGP extensions.
- o [I-D.ietf-bier-bier-yang] (BIER-YANG): This document describes the YANG data model to provision on every BFR BIER. It also provides OAM functions. There is currently no model expanding this to support BIER-TE. This framework document defines elements that should be included in a BIER-TE YANG model.
- o TBD: incomplete list ?.

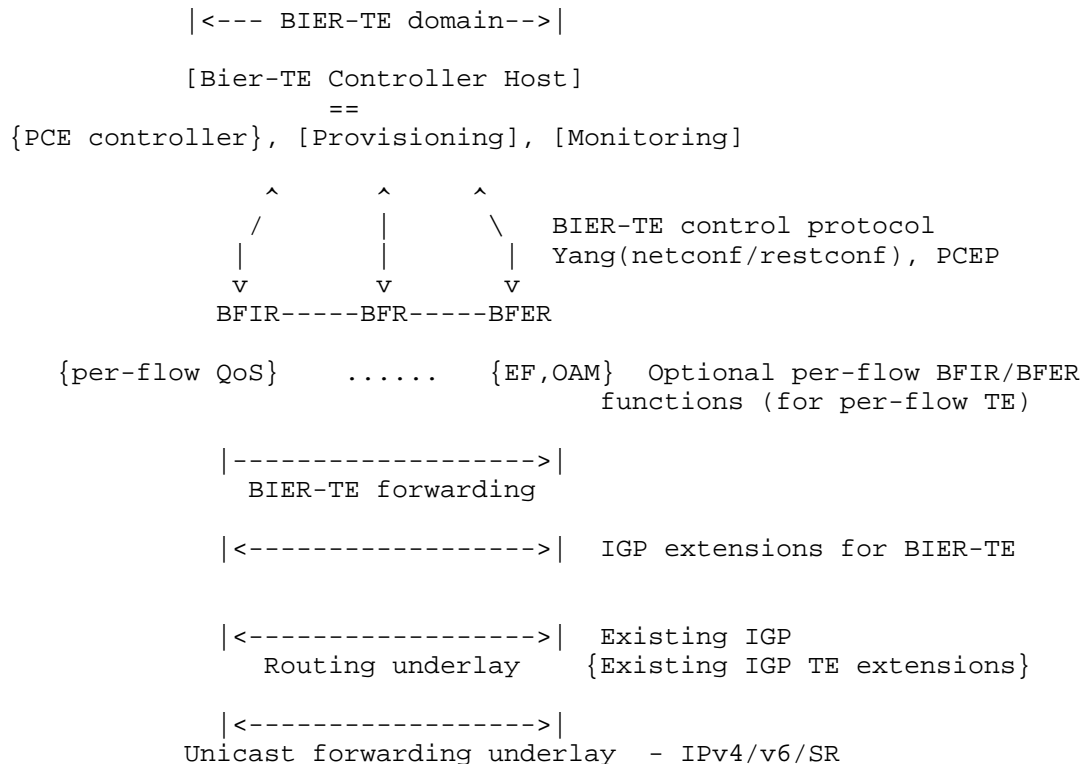


Figure 1: BIER-TE signaling architecture

The above picture is a modified version of Picture 2 from BIER-TE-ARCH reduced by the elements not considered in this document, and refined with those that are intended to be described by this document.

In comparison with BIER-TE-ARCH, Picture 2, this picture and this document do not include considerations for specific multicast flow overlay elements. Instead, it adds description of optional BFIR/BFER elements for per-flow QoS/EF (Elimination Function) and OAM, which are optional parts of an overall BIER-TE traffic engineering architecture. See BIER-EF-OAM for more background.

The routing underlay is refined in this document to consider a unicast forwarding underlay of IPv4/IPv6 and/or unicast SR (Segment Routing) for BIER-TE "forward_routed" adjacencies. It also assumes an existing IGP, such as ISIS or OSPF as the routing underlay. This may include (TBD) extensions already supporting TE aspects (like those IGP extensions done for RSVP-TE).

This framework intends to support a wide range of options to instantiate it:

In one extreme (PCEC only), there is no IGP in the network that BIER-TE depends on, but all BIER-TE operations is managed in an SDN-style fashion from centralized components called "BIER-TE Controller Host" in BIER-TE-ARCH. This central packend can be further subdivided into a Configuration/Provisioning component to install the BIER-TE topology into the network and a PCEC (Pat Computation Engine Controller) and (TBD) monitoring components. After BIER-TE is operational, the PCEC calculates BIER-TE bitstrings for BFIR when they need to send traffic flow to

In the other extreme (IGP only), there is no need for a PCEC or NMS. The initial setup of the BIER-TE topology can be performed manually, using configuration options to support automatic consistency checking and partial auto-configuration to simplify this work. BIER-TE extensions of the IGP are used for consistency checking and autoconfiguration and finally to provide the whole BIER-TE topology to BFIR that can then autonomously calculate BIER-TE bitstrings without the help of a PCEC.

2. BIER-TE Topology management

2.1. Operational model

When a network is installed, BIER-TE is added as a service or later when it is meant to change, BFR need to be (re)provisioned. This involves a planning phase which physical adjacencies (links) should be used in the BIER-TE topology, and which virtual adjacencies (routed adjacencies) should be created and assigned bits. Ultimately this means the definition of the BIER-TE topology.

When the physical topology if the network is smaller than the possible bitstring size (e.g.: 256 bits), then this can be a simple, fully automated process. Likewise, if multiple disjointed services for BIER-TE each require active subsets of the network topology smaller than the network topology, it likewise can be simple to create a different SD (subdomain) BIER-TE topologies for each such service.

When the required network topology for a BIER-TE service exceeds the supportable bitstring size, bit-saving mechanisms can be employed as described in BIER-ARCH. Some of them such as p2p link bits or lan-bits are easily automatically calculated. Creation of virtual adjacencies (routed adjacencies) may likely best be done with operator defined policies applied to a system a system calculating the bits for the BIER-TE topology.

Ultimately, if the set of required destinations plus transit hops exceeds the size of available bitstrings after optimization, multiple BIFT == bitstrings need to be allocated to support this case. These multiple BIFT will likely need to be engineered to minimize duplicate traffic load on the network and minimize bit use. One example shown in BIER-TE-ARCH is to allocate different <SD,SI> BIFT to different areas of a network, therefore having to create one BIER-TE packet copy per required destination region, but in result having only one packet copy in each of those regions.

Provisioning / initial setup can be done manually in simpler networks or through a provisioning system. A PCEP may equally perform this function. If a PCEP is not used to perform this function, but a PCEP is used later for Flow Management, then the PCEP does of course need to also learn the BIER-TE topologies created by the provisioning system.

Unless a PCEC is used for provisioning/initial setup, YANG is likely the preferred model to install the BIER-TE topology information into the BFR. If a PCEC is used, YANG or PCEC seem to be valid choices.

When the network topology expands, bit assignments for the new parts of the topology need to be made. If expansion was not factored into the initial bit assignment plans, this can lead to the need to reassign bits for existing parts of the topology. Support for such processes could be simplified through additional topology information, for example to enable seamless switching of traffic flows from bits in one SD over to bits in another SD. This is currently not considered in this document.

2.2. BIER-TE topology model

```

<BFR> BIFT information:
  Instance: "configured", "operational",
            "learned-configured", "learned-operational" (pce, igp)
  BIFT-ID: <SD subdomain,BSL bitstring length,SI Set Identifier>
  BIFT-Name: string (optional)
  BFR-ID: 16 bit (BIER-TE ID of the <bfr> in this BIFT
                or undefined if not BFER in this BIFT)
  Ingres-groups: (list of) string (1..16 bytes)
                 (that <bfr> is a member of)
  EF: <TBD> (optional, parameters for EF Function on this BIFT)
  OAM: <TBD> (optional, parameter for OAM Function on this BIFT)
  Bits: (#BSL - BitStringLength)
        BitIndex: 1...BSL
        BitType(/Tag): "unassigned",
                       (if unassigned, must have no adjacencies)
                       "unique", "p2p", "lan", "leaf", "node", "flood",
                       "group"
                       (more BitTypes defined in text below)
  Names: (list of 0 or more) string (1..16 bytes)
         (for BitTypes that require it)
  List of 0 or more adjacencies:
    (The following is the list of possible types of adjacencies,
     as defined in BIER-TE-ARCH with parameters)
    local_decap:
      VRFcontext: string (TBD)
    forward_connected:
      destination-id: ip-addr (4/16 bytes, router-id/link-local)
      link-id: ifIndex Value (connecting to destination)
      boolean: DNR (Do Not Reset)
    forward_routed:
      destination-id: 20 bit (SID), 4 or 16 bytes (router-id)
      TBD: path/encap information (e.g: SR SID stack)
  ECMP:
    list of 2 or more forward_connect and/or
    forward_routed adjacencies

```

Figure 2: BIER-TE topology information

The above picture shows informally the data model for BIER-TE topology information. <BFR> is a domain-wide unique identifier of a BFR, for example the router-id of the IGP (if an IGP is used). Every <BFR> has a "configured" instance of the BIFT information for every BIFT configured on it. This configuration could be created from legacy models, a YANG model, PCEP, or other means.

Every <BFR> also has an "operational" instance of the BIFT information. If the BFR has nor "learned-configured" / "learned-operational" information, then the "operational" instance is just a

copy of the "configuration" instance, but would take additional local information into account. For example, if resource limits do not allow to activate configured BIFT. Or when bits in the BIFT point to interfaces/adjacencies that are down, this could potentially also be reflected in the operational instance. While the "configuration" instance is read/write, the operational instance is read-only (from NMS or PCEC).

To calculate paths/bitstrings through the topology without the help of a PCEC, a BIFT would need to know the network wide BIER-TE topology. This topology consists of the "operational" BIFT informations of the BFR itself plus the "learned-operational" BIFT information from all other BIER-TE nodes in the network plus the underlay routing topology information, for example from an IGP. When an IGP is used, the "learned-operational" information of another BFR is simply learned because the BFRs are flooding this information as IGP information.

In the absence of any IGP, or the desire not to use it to distribute BIER-TE topology information, an NMS or PCEC could collect the "operational" BIER-TE topology information from BFRs and distribute it to BFRs to enable them to calculate BIER-TE bitstrings autonomously.

The operational instance of the topology information can depend on the presence of an IGP. If the adjacency of a bit in the BIFT is configured to use a nexthop identifier that has to be learned from an IGP, such as a Segment Routing SID or a router-ID, then the operational instance (as well as distributed learned-operational ones) would indicate that such an adjacency is non-operational if the BFR could not resolve this nexthop information. Forward_connected adjacencies do not require a routing underlay, but just link-local connectivity.

Some information elements in the BIER-TE topology information is metadata to support automatic consistency checking of learned topology information which permit to prohibit use of adjacencies that would not lead to working paths or worst case could create loops. The same information can also be used to auto-configure some adjacencies, specifically routed adjacencies, allowing to minimize operator work in case BIFT topology information is not auto-created from an NMS/PCEP but through manual mechanisms, but also to automatically discover mis-wirings and avoid them to be used.

The semantic of BitType and Names are described in conjunction with consistency checking and autoconfiguration in the following sections.

2.3. Consistency checking

The BitType and associated Name or Names for the bit are intended to support automated consistency checking and different reactions. an NMS can for example discover misconfiguration or miscablings and alert the operator. BFIR can likewise discover misconfiguration when the "configured" and "operational" instances of BFR are distributed via the IGP and are therefore available as "learned-configured" and "learned-operational" on the BFIR. The BFIR can then fr example stop using those misconfigured bits in any bitstrings it calculates and further escalate (e.g.: overlay signaling) unreachability of any BFER (or inability to calculate paths supporting required TE features).

"Unique" bits doe not require a name, but the <SD,SI> bit in question must only have an adjacency on one BFR. If it shows up with adjacencies on more than one BFR, this is an inconsistency.

"p2p" bits need to be the same bit on both BFR connected to each other via a subnet, and must be pointing to each other via "forward_connected" adjacencies. A "p2p" bit needs to have one Name parameter unique in the domain - for example constructed from concatenating the IfIndex of both sides. Note that the actual subnet does not need to be p2p, a BFR can have multiple bits across a multiaccess subnet, one for each neighbor.

Not listed in the above picture, but a "remote-p2p" could be a BitType when a bidirectional adjacency between two remode BFR using forward_routed adjacencies.

A "leaf" bit is the one shared bit in a <SD,SI> bitstring assigned to the "local_decap" adjacency on all leaf BFER. Leaf BFER do not need a separate bit. See BIER-TE-ARCH. If more then one "lead" bits are used in an <SD,SI> across the domain that is an inconsistency - waste of bits.

A "node" bit is associated with a Name that follows a standardized form to identify a node - e.g.: its router-id. On a non-leaf BFER, this bit can only have one local_decap adjacency on the node indicated itself. On a leaf BFER, the "node" bit must be assigned to adjacencies on one or BFR that connect to the indicated BFER. Other configurations (or wirings) are a misconfiguration.

A "lan" bit indicates a bit for a LAN, as discussed in BIER-TE-ARCH. It must have one domain wide unique name. It must only be used by BFR connecting to the same subnet with a set of forward_connected adjacencies pointing to the other BFR on that subnet. Disabling the use of a "lan" bit either on a BFIR when sending packets, or even more son on the actual BFR connecting to a subnet and recognizing

inconsistent BIER-TE topology configuraiton for it - is the most important automatic function to avoid mis-routing of BIER-TE packets. The looping will be also stopped because bits are reset when packets traverse the paths, or ultimately by TTL, but neither mechanism can provide as specifica OAM information about what went wrong than recognizing inconsistencies via the IGP.

TBD: flood bit, DNR (like lan bit, but more complex.

Consistency checking may happen directly during configuration as well as later during rewiring/remot changes of topology.

In general, the operational instance of the BIER-TE topology are relevant to topology consistency checking (as hey are for path calculations). For example, future extensions may actually introduce some form of node/BFR redundancy where different BFR are configured for the same bits, but only one at a time is actively using a bit, and therefore announcing it in the operational instance of the BIER-TE topology.

2.4. Auto-configuration

For subnets, the actual adjacency to the neighbor on a link may not actually be configured explicitly, but only the interface. Discovery of the neighbor via the IGP would result in a complete working adjacency for a bit, and that adjacency would show then in the operational instance - while the configured instance would only show an incomplete adjacency and the bit that was configured for the adjacency. The Name parameter can be used in configuration to lock in the BFR that is expected to be on the other side of a subnet interface. If that node is not the one actually connected, the adjacency in the operational instance would not be completed.

When a "p2p" BitType is used, but the bit is configured inconsistently on both sides of a p2p link, an autoconfiguration mechanism may be specified to select which of the two bits should be used (e.g.: bit number configured on the higher router-id peer). This could help to auto-correct a configuration mistake, but it does of course not recover the inconsistently configured bit directly, it just ignores it.

When a "lan" or "flood" BitType is configured, likewise auto-configuration can be done to overcome misconfigurations. TBD: more details.

Most importantly, configuration of routed adjacencies can create most need for network-wide consistent configuration. This can be automated with the proposed "group" bittype.

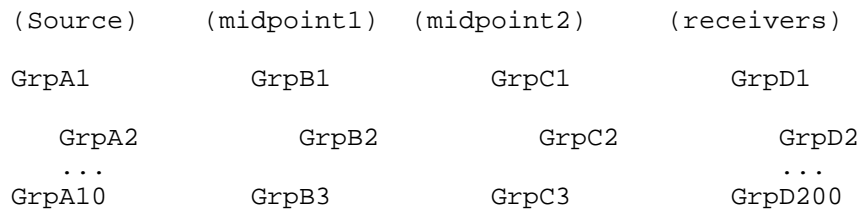


Figure 3: Group BitType use

The typical set of forward_routed adjacency is to allow steering of BIER-TE packets through a sequence of one or more members of a hop-group, load-balancing across them for TE reasons. In the above picture, those paths would start from a BFIR in GrpA and go via one (or more) nodes in GrpB, then GrpC and then BFER (GrpD).

To half-automate the setup of such loose hops, each member of GrpC would for example be configured with one unique bit of BitType "group" and the Name parameter would be set to "GrpB". Each midpoint1 BFR would "GrpB" in the list of strings for the BIFT Ingres-Group parameter. When such a BFR discovers (e.g.: via the IGP) a BFR "learned-operational" bit of BitType group with a name "GrpB" (and no adjacency!), then that midpoint1 BFR would create an adjacency in its "operational" instance, pointing to the announcing BFR with a "forward_routed" adjacency.

The saving through such group BitTypes is therefore that the bit had only to be configured on one node (the receiver side of the forward_routed adjacency), but would be configured on any number of ingres BFR for the adjacency. In the above picture, the benefit would be biggest if forward_routed adjacencies were used from Source to midpoint1, because the number of Sources is potentially largest (e.g: as shown in the picture 10 BFIR in Source group).

3. Flow Management

3.1. Operational / Architectural Models

Once a BIER-topology is active in a network, it can be used to pass BIER-TE packets. Typically this also requires the provisioning of some routing overlay because today, all applications defined for BIER today are classical SP PE-PE application where some customer traffic is mapped to SP traffic via PE-PE "overlay" signaling.

Applications in future environments such as industrial control or IoT may result in different overlay signaling. Even native end-to-end BIER-TE from application stacks is possible, but has so far not been defined.

Overlay signaling is currently out of scope of this document.

3.1.1. Overprovisioning

In the "overprovisioning flow management" model, the network operator is responsible to engineer the available network resources, BIER-TE Topology and applications generating BIER-TE flows such that the required resources can be guaranteed without contention - and potentially without the help of either PCEP or IGP, but simply using provisioning to configure BFIR and overlay signaling to determine active destinations.

Overprovisioning is the most control/signaling lightweight approach and currently the standard approach in most enterprises and service provider for IP multicast traffic.

For example: An ISP with a ++40Gbps network and a comparable small amount of high-value muticast traffic requiring in aggregate less than 5 Gbps can easily carry all of that multicast traffic across any available path. This is especially easy when the majority of traffic is best effort traffic (such as Internet traffic). In that case, the multicast traffic would be carried in a traffic class that is overprovisioned, for example with 6 Gbps guaranteed on every link. Calculated BIER-TE bitstrings would for example be used to reduce cost of multicast distribution (e.g.: steiner tree calculation), use disjoint paths (in conjunction with EF), or simply load-balance across all available non-ECMP paths. Overprovisioning flow management is traditional in most SP networks (core/edge/access) for IP multicast traffic and requires no additional signaling.

The overprovisioning flow management model is one that likely would request for (only) a YANG model to provision the BIER-TE topology.

3.1.2. PCEC

In the PCEC based flow management model, a PCEP determines (calculates) the (flow-id,<SD,SI>,bitstring) for a traffic flows and signals this to the BFIR sourcing the flow (its BFR-ID is part of the flow-id). If the flow was not statically defined, then this step would be preceeded with the BFIR requesting the resources for the, indicating the requested resources as well as the set of destinations. The destinations could be indicated as BFR-ID or (likely easier for the BFIR) by their unique identifiers in unicast routing (e.g.: router-ID). The bitstring returned by the PCEP would include not only engineered paths to all these destinations, but those paths could also be disjoint paths, carrying the traffic twice towards each destination and merging them via the EF function. The BFIR could be fully agnostic to these PCEP choices.

One of the core benefits of using BIER-TE forwarding is the ability to change the bitstring on a per-packet basis to re-route traffic by setting different transit bits, or to quickly add/delete destinations. When the BFIR should be empowered to perform any of these functions without the need for help by the PCEP, then the PCEP needs to provide additional information back to the BFIR.

If a BFIR has for example an OAM capability to determine without the help of a controller that a path has failed (too much packet loss on destination, signalled back to BFIR), and dual-transmission is not desired (due to double resource usage), then the PCP and BFIR could co-operate on a path-protection scheme in which the PCEP provides for flows not one, but two bitstrings, one being the backup path which is used by the BFIR when it discovers via OAM loss on the currently used path. This approach can extremely reduce the need to rely on controller help during failures.

When the destinations for a particular flow can potentially change over time, this can often be faster and more efficiently signalled directly via the overlay signaling to the BFIR instead of going through the PCEP. To support this mode of operations, the BFIR could request from the PCEP not simply the current set of destinations for a flow, but instead the maximum superset of receivers and request per-destination information. The PCEP would then return not just one bitstring, but one bitstring per destination (BFER). The BFIR would simply OR the bitstrings for all required destinations for each packet to create the final bitstring for that packet. Note that this description is of course on a per- $\langle SD, SI \rangle$ (aka: per BIFT) basis. Destinations using different BIFTs require always different BIER-TE packets to be sent by the BFIR.

3.1.2.1. per-flow QoS - policer/shaper/EF

In the PCEP based resource management model, it is up to the PCEP to determine how explicit resource reservations should be managed, e.g.: whether or how it tracks resource consumption. The BIER-TE forwarding plane itself does not support per-flow state with the exception of EF, which would usually be a function enabled on BFER.

Likewise, per-flow policer and/or shaper state may be a useful optional feature that the PCEP should be able to request to be enabled on a BFIR to ensure that the traffic passed by the BFIR into the BIER-TE domain does not overrun resources available. In the simplest case, such a shaper/policer could simply reflect the resources indicated by the BFIR in its request to the PCEP.

Per-flow policer/shaper or EF may need to be explicitly instantiated by BFIR/BFER. Instantiation of the Policer/Shaper on the BFIR can

happen as a function of the PCEP signaling to the BFIR, but instantiation of the EF would also require signaling of the PCEP to the BFER(s) for flows. Note that EF could also be instantiated on any midpoint BFR, so the PCEP would need to know the BIER-TE topology including where EF is considered and manage it through appropriate signaling.

Note that it is unclear yet, if EF implementations could or should be implemented with or without the need for explicit instantiation, the BIER-TE-EF-OAM document allows both options. Even in the absence of explicit signaling, per-flow Policer/Shaper and EF are limited resources and PCEP should keep track of how much of these resources are allocated and available for future flows. Like other path resources, exhaustion may require PCEP failure to allocate responses or other mitigating options.

3.1.2.2. DiffServ QoS

The only resource management that could be expected to exist in the BIER-TE domain hop-by-hop would be DiffServ QoS. As outlined in the above overprovisioning resource management model, it can serve as an easy method for lightweight resource management, and as soon as the network intends to use more than one such DiffServ codepoint across different BIER-TE flows, the PCEP should likely be able to understand and manage the DiffServ assignments of BIER-TE flows and signal the selected codepoint back to the BFIR.

3.2. BIER-TE flow model

```

BIER-TE traffic flow (change) request (from BFIR):
  Flow-control-ID: <identifier>
  Ingres BFIR of flow: (IGP router-id ?!)
  Destination-ID: set of BFER identifiers (IGP router-id ?!)
  extended-reply-required (boolean)
  Requirements:
    TSPEC (bandwidth, burst size,...)
    resilience: dual-transmission with EF
    shared-group: name

BIER-TE traffic flow reply/command (to BFIR):
  Flow-control-ID: <identifier>
  Ingres Policer/Shaper parameters (applies to each BIFT)
  Set of 1 or more BIFT:
    <SD, SI, BSL>
    BFIR-ID, entropy  (form together flow-ID)
    Bitstring
    QoS, TTL,

BIER-TE traffic flow extended reply/command (to BFER):
  Flow-control-ID: <identifier>
  Ingres Policer/Shaper parameters (applies to each BIFT)
  Set of 1 or more BIFT:
    <SD, SI, BSL>
    BFIR-ID, entropy  (form together flow-ID)
    QoS, TTL
  List of 1 or more destinations
    Destination-ID, Bitstring

BIER-TE traffic flow command (to BFER):
  Flow-control-ID: <identifier>
  Ingres BFIR of flow: BFIR-ID (in BIER-TE packet header)
  Set of 1 or more BIFT:
    <SD, SI, BSL>
    BFIR-ID, entropy  (form together flow-ID)
    EF parameter (window size etc..)

```

Figure 4: Flow request/reply/commands

The above picture shows an initial abstract representation of the data models for the different type of request/replies discussed in the previous section between PCEC and BFIR (and in one case BFER).

The Flow-control-ID identifies the managed object itself: a flow to be sent from one BFIR to a set of BFER with some TE requirements, which ultimately may require BIER-TE packets for one or more BIFT.

BFIR and BFER need to be identified in the request in a form not specific to the bits of BIFT, so the PCEP can select the appropriate BIFT(s) to use. The above picture assumes the router-id of BFIR and BFER are appropriate.

The request includes TE requirements, including (something like a) TSPEC for bandwidth, burst-size or the like, whether or not dual-transmission via PREF is required, and if the resource used are to be shared across multiple flows, then the name of a shared group. One example of sharing would for example be a video-conference where the speaker transmits video, every speaker requests/allocates a BIER-TE flow from the PCEP, but the resources for those flows are of course shared (only one flow active at a time).

The reply from the PCEP lists the BIFTS/packets that must be sent by the BFIR to reach the desired destinations as well as any other BIER-TE packet header fields relevant <SD,SI,BSL>, BFIR-ID, entropy, QoS, TTL. Beside the BIER-TE packet header, the parameters for the policer and/or shaper to be used by the BFIR are signalled back.

The extended reply does not provide simply the bitstring to use for each BIFT, but instead lists the bitstrings required for each destination so that (as described above), the BFIR can simply add/delete destinations on a packet-by-packet basis OR'ing those bitstrings.

Finally, a command to BFER is required to instruct the creation of EF state in case this can not be done automatically.

4. Security Considerations

TBD.

5. IANA Considerations

This document requests no action by IANA.

6. Acknowledgements

TBD.

7. Change log [RFC Editor: Please remove]

00: Initial version.

8. References

- [I-D.eckert-bier-te-frr]
Eckert, T., Cauchie, G., Braun, W., and M. Menth,
"Protection Methods for BIER-TE", draft-eckert-bier-te-frr-02 (work in progress), June 2017.
- [I-D.huang-bier-te-encapsulation]
Huang, R., Eckert, T., Wei, N., and P. Thubert,
"Encapsulation for BIER-TE", draft-huang-bier-te-encapsulation-00 (work in progress), March 2018.
- [I-D.ietf-bier-bier-yang]
Chen, R., hu, f., Zhang, Z., dai.xianxian@zte.com.cn, d.,
and M. Sivakumar, "YANG Data Model for BIER Protocol",
draft-ietf-bier-bier-yang-03 (work in progress), February 2018.
- [I-D.ietf-bier-isis-extensions]
Ginsberg, L., Przygienda, T., Aldrin, S., and Z. Zhang,
"BIER support via ISIS", draft-ietf-bier-isis-extensions-09 (work in progress), February 2018.
- [I-D.ietf-bier-ospf-bier-extensions]
Psenak, P., Kumar, N., Wijnands, I., Dolganow, A.,
Przygienda, T., Zhang, Z., and S. Aldrin, "OSPF Extensions
for BIER", draft-ietf-bier-ospf-bier-extensions-15 (work
in progress), February 2018.
- [I-D.ietf-bier-te-arch]
Eckert, T., Cauchie, G., Braun, W., and M. Menth, "Traffic
Engineering for Bit Index Explicit Replication (BIER-TE)",
draft-ietf-bier-te-arch-00 (work in progress), January 2018.
- [I-D.thubert-bier-replication-elimination]
Thubert, P., Eckert, T., Brodard, Z., and H. Jiang, "BIER-
TE extensions for Packet Replication and Elimination
Function (PREF) and OAM", draft-thubert-bier-replication-elimination-03 (work in progress), March 2018.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A.,
Przygienda, T., and S. Aldrin, "Multicast Using Bit Index
Explicit Replication (BIER)", RFC 8279,
DOI 10.17487/RFC8279, November 2017,
<<https://www.rfc-editor.org/info/rfc8279>>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.

Author's Address

Toerless Eckert
Futurewei Technologies Inc.
2330 Central Expy
Santa Clara 95050
USA

Email: tte+ietf@cs.fau.de

TEAS WG
Internet Draft
Intended status: Informational
Expires: August 28, 2018

Young Lee
Haomian Zheng
Huawei

Daniel Ceccarelli
Ericsson

Bin Yeong Yoon
ETRI

Sergio Belotti
Nokia

February 28, 2018

Applicability of YANG models for Abstraction and Control of Traffic
Engineered Networks

draft-ietf-teas-actn-yang-01

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 28, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

Abstraction and Control of TE Networks (ACTN) refers to the set of virtual network operations needed to orchestrate, control and manage large-scale multi-domain TE networks, so as to facilitate network programmability, automation, efficient resource sharing, and end-to-end virtual service aware connectivity and network function virtualization services.

This document explains how the different types of YANG models defined in the Operations and Management Area and in the Routing Area are applicable to the ACTN framework. This document also shows how the ACTN architecture can be satisfied using classes of data model that have already been defined, and discusses the applicability of specific data models that are under development. It also highlights where new data models may need to be developed.

Table of Contents

1. Introduction.....	3
2. Abstraction and Control of TE Networks (ACTN) Architecture.....	3
3. Service Models.....	5
4. Service Model Mapping to ACTN.....	6
4.1. Customer Service Models in the ACTN Architecture (CMI)....	7
4.2. Service Delivery Models in ACTN Architecture.....	8
4.3. Network Configuration Models in ACTN Architecture (MPI)...	8
4.4. Device Models in ACTN Architecture (SBI).....	9
5. Examples of Using Different Types of YANG Models.....	9

5.1. Topology Collection.....	9
5.2. Connectivity over Two Client Nodes.....	10
5.3. VN service example.....	11
5.4. Data Center-Interconnection Example.....	11
5.4.1. CMI (CNC-MDSC Interface).....	13
5.4.2. MPI (MDSC-PNC Interface).....	13
6. Security.....	14
7. Acknowledgements.....	14
8. References.....	14
8.1. Informative References.....	14
9. Contributors.....	16
Authors' Addresses.....	17

1. Introduction

Abstraction and Control of TE Networks (ACTN) describes a method for operating a Traffic Engineered (TE) network (such as an MPLS-TE network or a layer 1 transport network) to provide connectivity and virtual network services for customers of the TE network. The services provided can be tuned to meet the requirements (such as traffic patterns, quality, and reliability) of the applications hosted by the customers. More details about ACTN can be found in Section 2.

Data models are a representation of objects that can be configured or monitored within a system. Within the IETF, YANG [RFC6020] is the language of choice for documenting data models, and YANG models have been produced to allow configuration or modelling of a variety of network devices, protocol instances, and network services. YANG data models have been classified in [Netmod-Yang-Model-Classification] and [Service-YANG].

This document shows how the ACTN architecture can be satisfied using classes of data model that have already been defined, and discusses the applicability of specific data models that are under development. It also highlights where new data models may need to be developed.

2. Abstraction and Control of TE Networks (ACTN) Architecture

[ACTN-Requirements] describes the high-level ACTN requirements. [ACTN-Frame] describes the architecture model for ACTN including the entities (Customer Network Controller (CNC), Multi-domain Service Coordinator (MDSC), and Physical Network Controller (PNC)) and their interfaces.

Figure 1 depicts a high-level control and interface architecture for ACTN and is a reproduction of Figure 3 from [ACTN-Frame]. A number of key ACTN interfaces exist for deployment and operation of ACTN-based networks. These are highlighted in Figure 1 (ACTN Interfaces) below:

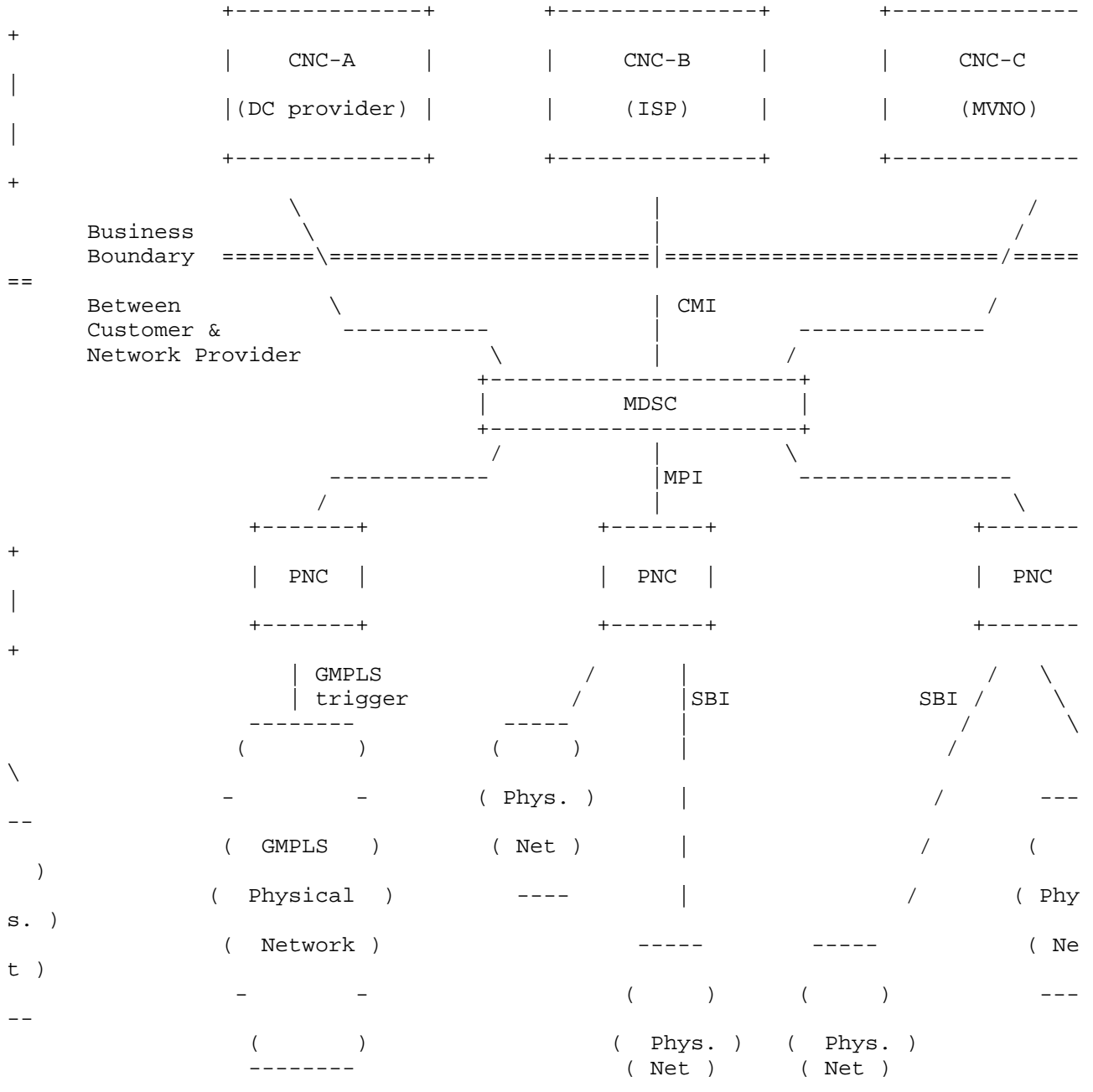


Figure 1 : ACTN Interfaces

The interfaces and functions are described below (without modifying the definitions) in [ACTN-Frame]:

- . The CNC-MDSC Interface (CMI) is an interface between a Customer Network Controller and a Multi Domain Service Controller. The interface will communicate the service request or application

demand. A request will include specific service properties, for example, services type, bandwidth and constraint information. These constraints SHOULD be measurable by MDSC and therefore visible to CNC via CMI. The CNC can also request the creation of the virtual network based on underlying physical resources to provide network services for the applications. The CNC can provide the end-point information/characteristics, traffic matrix specifying specific customer constraints. The MDSC may also report potential network topology availability if queried for current capability from the Customer Network Controller.

- . The MDSC-PNC Interface (MPI) is an interface between a Multi Domain Service Coordinator and a Physical Network Controller. It allows the MDSC to communicate requests to create/delete connectivity or to modify bandwidth reservations in the physical network. In multi-domain environments, each PNC is responsible for a separate domain. The MDSC needs to establish multiple MPIs, one for each PNC and perform coordination between them to provide cross-domain connectivity.
- . The South-Bound Interface (SBI) is the provisioning interface for creating forwarding state in the physical network, requested via the Physical Network Controller. The SBI is not in the scope of ACTN, however, it is included in this document so that it can be compared to models in [Service-Yang].

3. Service Models

[Service-YANG] introduces a reference architecture to explain the nature and usage of service YANG models in the context of service orchestration. Figure 2 below depicts this relationship and is a reproduction of Figure 2 from [Service-YANG]. Four models depicted in Figure 2 are defined as follows:

- . Customer Service Model: A customer service model is used to describe a service as offer or delivered to a customer by a network operator.
- . Service Delivery Model: A service delivery model is used by a network operator to define and configure how a service is provided by the network.
- . Network Configuration Model: A network configuration model is used by a network orchestrator to provide network-level configuration model to a controller.
- . Device Configuration Model: A device configuration model is used by a controller to configure physical network elements.

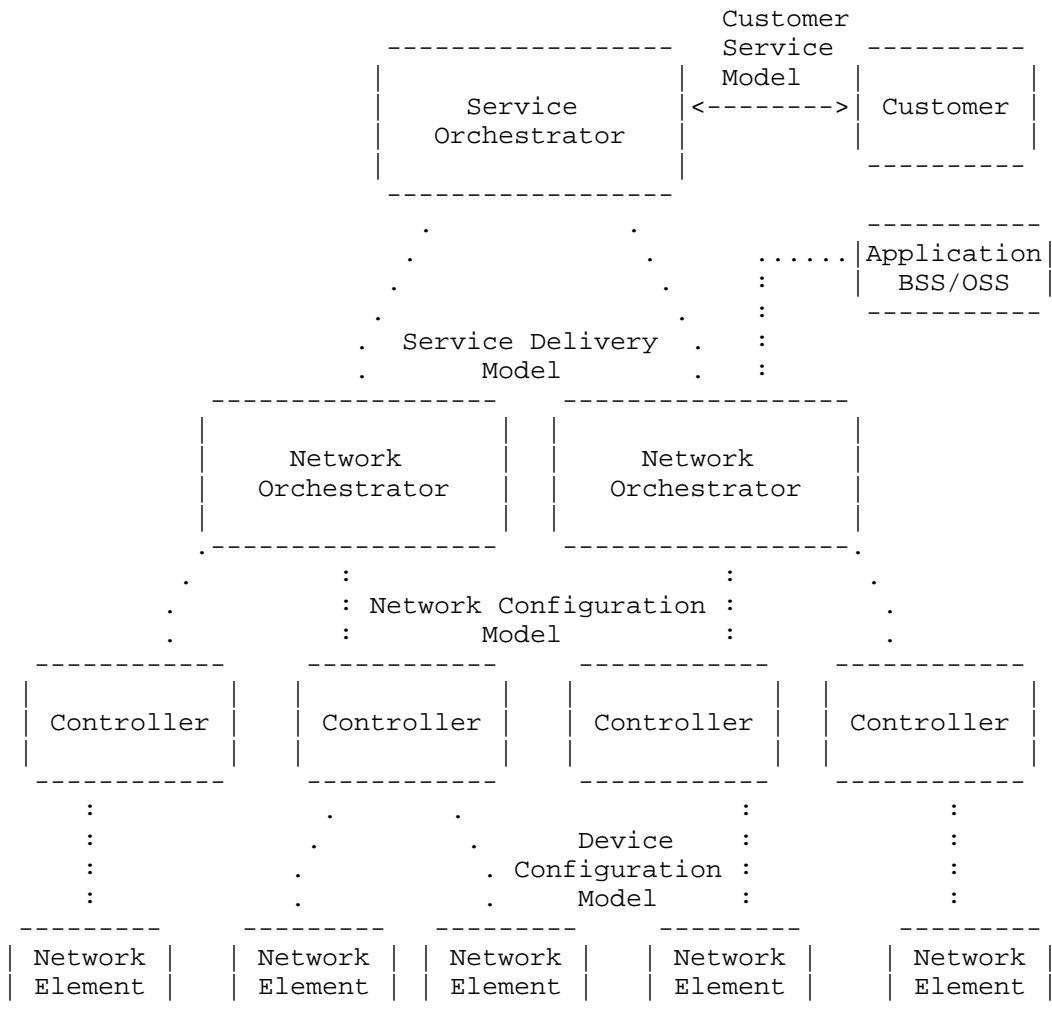


Figure 2: An SDN Architecture with a Service Orchestrator

4. Service Model Mapping to ACTN

YANG models coupled with the RESTCONF/NETCONF protocol [Netconf][Restconf] provides solutions for the ACTN framework. This section explains which types of YANG models apply to each of the ACTN interfaces.

Refer to Figure 5 of [ACTN-Frame] for details of the mapping between ACTN functions and service models. In summary, the following

mappings are held between and Service Yang Models and the ACTN interfaces.

- o Customer Service Model <-> CMI
- o Network Configuration Model <-> MPI
- o Device Configuration Model <-> SBI

4.1. Customer Service Models in the ACTN Architecture (CMI)

Customer Service Models, which are used between a customer and a service orchestrator as in [Service-YANG], should be used between the CNC and MDSC (e.g., CMI) serving as providing a simple intent-like model/interface.

Among the key functions of Customer Service Models on the CMI is the service request. A request will include specific service properties, including: service type and its characteristics, bandwidth, constraint information, and end-point characteristics.

The following table provides a list of functions needed to build the CMI. They are mapped with Customer Service Models.

Function	Yang Model

VN Service Request	[ACTN-VN-YANG]
VN Computation Request	[ACTN-VN-YANG]*
TE & Service Mapping	[TE-Service-Mapping]**
VN Performance Monitoring Telemetry	[ACTN-PM-Telemetry]***
Topology Abstraction	[TE-topology]****

*VN computation request in the CMI context means network path computation request based on customer service connectivity request constraints prior to the instantiation of a VN creation.

**[TE-Service-Mapping] provides a mapping and cross-references between service models (e.g., L3SM, L2SM, L1CSM) and TE model via [ACTN-VN-YANG] and [TE-topology].

***ietf-actn-te-kpi-telemetry model describes performance telemetry for ACTN VN model. This module also allows autonomic traffic engineering scaling intent configuration mechanism on the VN level. Scale in/out criteria might be used for network autonomies in order

the controller to react to a certain set of variations in monitored parameters. Moreover, this module also provides mechanism to define aggregated telemetry parameters as a grouping of underlying VN level telemetry parameters.

****TE-Topology's Connectivity Matrices/Matrix construct can be used to instantiate VN Service via a suitable referencing and mapping with [ACTN-VN-YANG].

4.2. Service Delivery Models in ACTN Architecture

The Service Delivery Models where the service orchestration and the network orchestration could be implemented as separate components as seen in [Service-YANG]. This is also known as Network Service Models. On the other hand, from an ACTN architecture point of view, the service delivery model between the service orchestrator and the network orchestrator is an internal interface between sub-components of the MDSC in a single MDSC model.

In the MDSC hierarchical model where there are multiple MDSCs, the interface between the top MDSC and the bottom MDSC can be mapped to service delivery models.

4.3. Network Configuration Models in ACTN Architecture (MPI)

The Network Configuration Models is used between the network orchestrator and the controller in [Service-YANG]. In ACTN, this model is used primarily between a MDSC and a PNC. The Network Configuration Model can be also used for the foundation of more advanced models, like hierarchical MDSCs (see Section 4.5)

The Network Configuration Model captures the parameters which are network wide information.

The following table provides a list of functions needed to build the MPI. They are mapped with Network Configuration Yang Models. Note that various Yang models are work in progress.

Function	Yang Model
Configuration Scheduling	[Schedule]
Path computation	[PATH_COMPUTATION-API]
Tunnel/LSP Provisioning	[TE-Tunnel]
Topology Abstraction	[TE-topology]
Client Signal Description	[Client-signal]

Service Provisioning	TBD*
OTN Topology Abstraction	[OTN-YANG]
WSON Topology Abstraction	[WSON-YANG]
Flexi-grid Topology Abstraction	[Flexi-YANG]
OTN Tunnel Model	[OTN-Tunnel]
WSON TE Tunnel Model	[WSON-Tunnel]
Flexi-grid Tunnel Model	[Flexigrid-Tunnel]

* This function needs to be investigated further. This can be a part of [TE-Tunnel] which is to be determined. Service provisioning is an optional function that builds on top the path provisioning one.

[T-NBI Applicability] provides a summary on the applicability of existing YANG model usage in the current network configuration, especially for transport network.

4.4. Device Models in ACTN Architecture (SBI)

Note that SBI is not in the scope of ACTN, as there are already mature protocol solutions for various purpose on the device level of ACTN architecture, such as RSVP-TE, OSPF-TE and so on. The interworking of such protocols and ACTN controller hierarchies can be found in [gmpls-controller-inter-work].

For the device YANG models are used for per-device configuration purpose, they can be used between the PNC and the physical network/devices. One example of Device Models is ietf-te-device yang module defined in [TE-tunnel].

5. Examples of Using Different Types of YANG Models

This section provides some examples on the usage of IETF YANG models in the network operation. A few typical generic scenarios are involved. In [T-NBI Applicability], there are more transport-related scenarios and examples.

5.1. Topology Collection

Before any connection is requested and delivered, the controller needs to understand the network topology. The topology information is exchanged among controllers with topology models, such as [te-

topology]. Moreover, technology-specific topology reporting may use the model described in [OTN-YANG] [WSON-YANG], and [Flexi-YANG] for OTN, WSON and Flexi-grid, respectively. By collecting the network topology, each controller can therefore construct a local database, which can be used for the further service deployment.

There can be different types of abstraction applied between each pair of controllers as discussed in [ACTN-frame]. The technology-specific features may be hidden after abstraction to make the network operation easier.

When there are topology changes in the physical network, the PNC should report the change to upper level of controllers via updating messages using topology models. Accordingly, such changes are propagated between different controllers for further synchronization.

5.2. Connectivity over Two Client Nodes

The service models, such as described in [RFC8049], [L2SM] and [L1CSM], provide a connectivity service model which can be used in connection-oriented networks.

It would be used as follows in the ACTN architecture:

- . A CNC uses the service models to specify the two client nodes that are to be connected, and also indicates the amount of traffic (i.e., the bandwidth required) and payload type. What may be additionally specified is the SLA/Policy that describes the required quality and resilience of the service especially on TE binding with the service (e.g., soft isolation, hard isolation, etc.) as defined in [TE-Service-Mapping].
- . The MDSC uses the information in the request to pick the right network (domain) and also to select the provider edge nodes corresponding to the customer edge nodes.

If there are multiple domains, then the MDSC needs to coordinate across domains to set up network tunnels to deliver a service. Thus coordination includes, but is not limited to, picking the right domain sequence to deliver a service.

Additionally, an MDSC can initiate the creation of a tunnel (or tunnel segment) in order to fulfill the service request from CNC based on path computation upon the overall topology information it synthesized from different PNCs. The based model that can cater this purpose is the TE tunnel model specified in

[te-tunnel]. Technology-specific tunnel configuration may use the model described in [OTN-Tunnel] [WSON-Tunnel], and [Flexigrid-Tunnel] for OTN, WSON and Flexi-grid, respectively.

- . Then, the PNCs need to decide the explicit route of such a tunnel or tunnel segment (in case of multiple domains) for each domain, and then create such a tunnel using protocols such as PCEP and RSVP-TE or using per-hop configuration.

5.3. VN service example

The service model defined in [ACTN-VN-YANG] describes a virtual network (VN) as a service which is a set of multiple connectivity services:

- . A CNC will request VN to the MDSC by specifying a list of VN members. Each VN member specifies either a single connectivity service, or a source with multiple potential destination points in the case that the precise destination sites are to be determined by MDSC.
 - o In the first case, the procedure is the same as the connectivity service, except that in this case, there is a list of connections requested.
 - o In the second case, where the CNC requests the MDSC to select the right destination out of a list of candidates, the MDSC needs to evaluate each candidate and then choose the best one and reply with the chosen destination for a given VN member. After this is selected, the connectivity request setup procedure is the same as in the connectivity example in section 5.2.

After the VN is set up, a successful reply message is sent from MDSC to CNC, indicating the VN is ready. This message can also be achieved by using the model defined in [ACTN-VN-YANG].

5.4. Data Center-Interconnection Example

This section describes more concretely how existing YANG models described in Section 4 map to an ACTN data center interconnection use case. Figure 3 shows a use-case which shows service policy-driven Data Center selection and is a reproduction of Figure A.1 from [ACTN-Info].

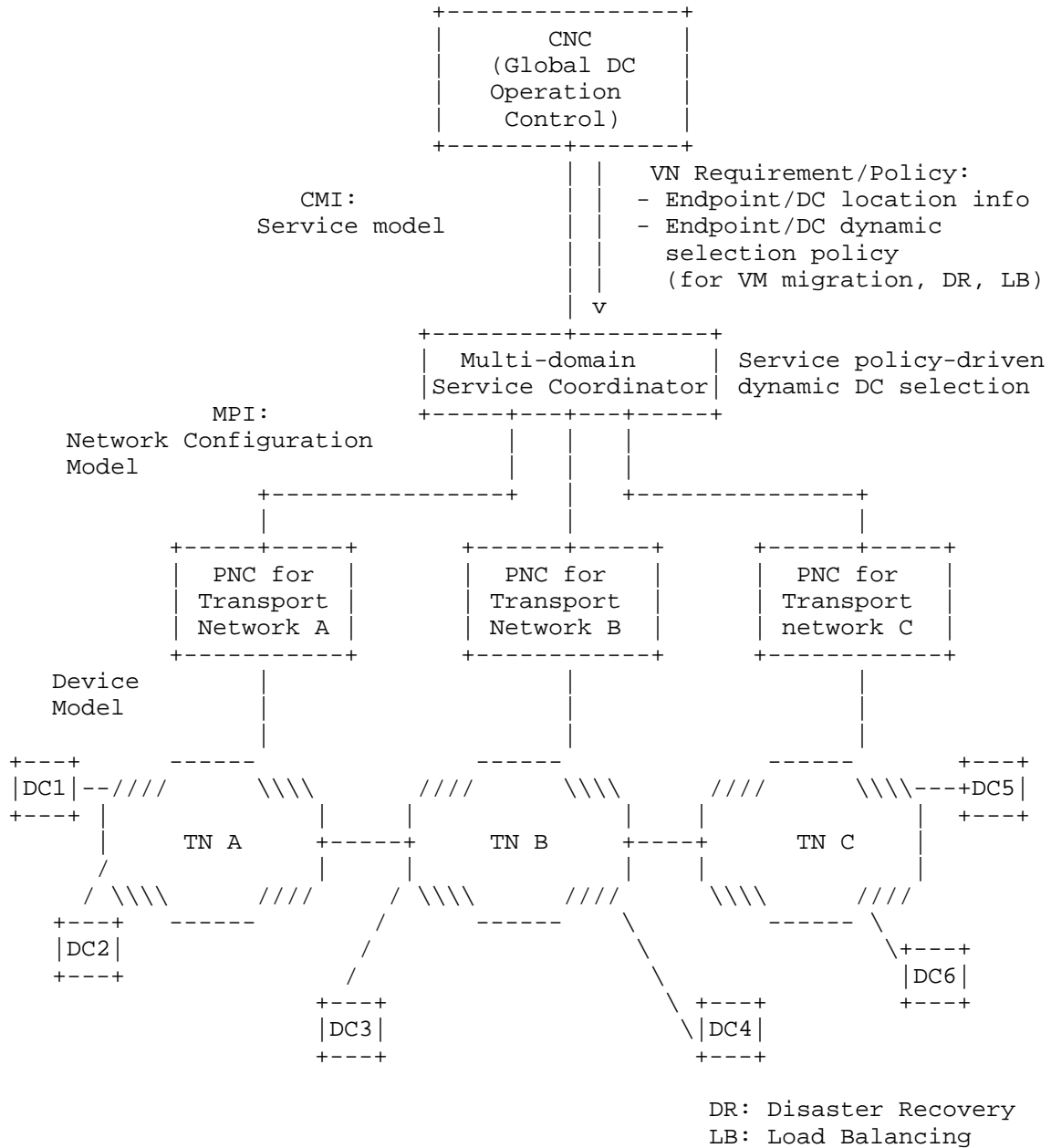


Figure 3: Service Policy-driven Data Center Selection

Figure 3 shows how VN policies from the CNC (Global Data Center Operation) are incorporated by the MDSC to support multi-destination applications. Multi-destination applications refer to applications in which the selection of the destination of a network path for a given source needs to be decided dynamically to support such applications.

Data Center selection problems arise for VM mobility, disaster recovery and load balancing cases. VN's policy plays an important role for virtual network operation. Policy can be static or dynamic. Dynamic policy for data center selection may be placed as a result of utilization of data center resources supporting VMs. The MDSC would then incorporate this information to meet the objective of this application.

5.4.1. CMI (CNC-MDSC Interface)

[ACTN-VN-YANG] is used to express the definition of a VN, its VN creation request, the service objectives (metrics, QoS parameters, etc.), dynamic service policy when VM needs to be moved from one Data Center to another Data Center, etc. This service model is used between the CNC and the MDSC (CMI). The CNC in this use-case is an external entity that wants to create a VN and operates on the VN.

5.4.2. MPI (MDSC-PNC Interface)

The Network Configuration Model is used between the MDSC and the PNCs. Based on the Customer Service Model's request, the MDSC will need to translate the service model into the network configuration model to instantiate a set of multi-domain connections between the prescribed DC sources and DC destinations. The MDSC will also need to dynamically interact with the CNC for dynamic policy changes initiated by the CNC. Upon the determination of the multi-domain connections, the MDSC will need to use the network configuration model such as [TE-Tunnel] to interact with each PNC involved on the path. [TE-Topology] is used to for the purpose of underlying domain network abstraction from the PNC to the MDSC.

6. Security

This document is an informational draft. When the models mentioned in this draft are implemented, detailed security consideration will be given in such work.

How security fits into the whole architecture has the following components:

- the use of Restconf security between components
- the use of authentication and policy to govern which services can be requested by different parties.
- how security may be requested as an element of a service and mapped down to protocol security mechanisms as well as separation (slicing) of physical resources)

7. Acknowledgements

We thank Adrian Farrel for providing useful comments and suggestions for this draft.

8. References

8.1. Informative References

- [Service-YANG] Q. Wu, W. Liu and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained, work in progress.
- [Netmod-Yang-Model-Classification] D. Bogdanovic, B. Claise, and C. Moberg, "YANG Module Classification", draft-ietf-netmod-yang-model-classification, work in progress.
- [Netconf] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241.
- [Restconf] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf, work in progress.

- [ACTN-Requirements] Y. Lee, et al., "ACTN Requirements for Abstraction and Control of TE Networks", draft-ietf-teas-actn-requirements, work in progress.
- [ACTN-Frame] D. Cecarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework, work in progress.
- [TE-Topology] X. Liu, et. al., "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo, work in progress.
- [TE-Tunnel] T. Saad (Editor), "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.
- [ACTN-VN-YANG] Y. Lee (Editor), "A Yang Data Model for ACTN VN Operation", draft-lee-teas-actn-vn-yang, work in progress.
- [ACTN-Info] Y. Lee & S. Belotti, "Information Model for Abstraction and Control of TE Networks (ACTN)", draft-leebelotti-teas-actn-info, work in progress.
- [Transport-Service-Model] X. Zhang (Editor), "A Service YANG Model for Connection-oriented Transport Networks", draft-zhang-teas-transport-service-model, work in progress.
- [PATH-COMPUTATION-API] I.Busi/S.Belotti et al. "Path Computation API", draft-busibel-ccamp-path-computation-api-00.txt, work in progress
- [RSVP-TE-YANG] T. Saad (Editor), "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp, work in progress.
- [Schedule] X. Liu, et. al., "A YANG Data Model for Configuration Scheduling", draft-liu-netmod-yang-schedule, work in progress.
- [ACTN-Abstraction] Y. Lee, D. Dhody, and D. Ceccarelli, "ACTN Abstraction Methods", draft-lee-tease-actn-abstraction, work in progress.
- [OTN-YANG] X. Zhang, A. Sharma, and X. Liu, "A YANG Data Model for Optical Transport Network Topology", draft-zhang-ccamp-ll-topo-yang, work in progress.

- [WSON-YANG] Y. Lee, et. al., "A Yang Data Model for WSON Optical Networks", draft-ietf-ccamp-wson-yang, work in progress.
- [Flexi-YANG] J.E. Lopez de Vergara, et. al., "YANG data model for Flexi-Grid Optical Networks", draft-vergara-ccamp-flexigrid-yang, work in progress.[ODU-Tunnel] Sharma, R. Rao, and X. Zhang, "OTN Service YANG Model", draft-sharma-ccamp-otn-service-model, work in progress.
- [ACTN-PM-Telemetry] Y. Lee, D. Dhody, S. Karunanithi, R. Vilalta, D. King, and D. Ceccarelli, "YANG models for ACTN TE Performance Monitoring Telemetry and Network Autonomics", draft-lee-teas-actn-pm-telemetry-autonomics, work in progress.
- [WSON-Tunnel] Y. Lee, D. Dhody, V. Lopez, D. King, B. Yoon, and R. Vilalta, "A Yang Data Model for WSON Tunnel", draft-ietf-ccamp-wson-tunnel-model, work in progress.
- [Flexigrid-Tunnel] J. Vergara, D. Perdices, V. Lopez, O. Gonzalez de Dios, D. King, Y. Lee, and G. Galimberti, "YANG data model for Flexi-Grid media-channels", draft-ietf-ccamp-flexigrid-media-channel-yang, work in progress.
- [TE-Service-Mapping] Y. Lee, et al, "Traffic Engineering and Service Mapping Yang Model", draft-lee-teas-te-service-mapping-yang, work in progress.
- [Client-signal] H. Zheng, et al, "A YANG Data Model for Optical Transport Network Client Signals", draft-zheng-ccamp-otn-client-signal-yang, work in progress.
- [T-NBI Applicability] I. Busi, et al, "Transport Northbound Interface Applicability Statement and Use Cases", draft-ietf-ccamp-transport-nbi-app-statement, work in progress.
- [gmpls-controller-inter-work] H. Zheng, et al, "Interworking of GMPLS Control and Centralized Controller System", draft-zheng-ccamp-gmpls-controller-inter-work, work in progress.

9. Contributors

Contributor's Addresses

Dhruv Dhody
Huawei Technologies

Email: dhruv.ietf@gmail.com

Xian Zhang
Huawei Technologies

Email: zhang.xian@huawei.com

Authors' Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838

Email: leeyoung@huawei.com

Haomian Zheng
Huawei Technologies

Email: zhenghaomian@huawei.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan, 48
Stockholm, Sweden

Email: daniele.ceccarelli@ericsson.com

Bin Yeong Yoon
ETRI

Email: byyun@etri.re.kr

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

Jong Yoon Shin
SKT

Email: jongyoon.shin@sk.com

Sergio Belotti
Nokia

Email: sergio.belotti@nokia.com

TEAS Working Group
Internet Draft
Intended status: Informational

Igor Bryskin
Huawei Technologies
Vishnu Pavan Beeram
Juniper Networks
Tarek Saad
Cisco Systems Inc
Xufeng Liu
Jabil

Expires: September 5, 2018

March 5, 2018

TE Topology and Tunnel Modeling for Transport Networks
draft-ietf-teas-te-topo-and-tunnel-modeling-01

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on September 5, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This document describes how to model TE topologies and tunnels for transport networks, by using the TE topology YANG model [I-D.ietf-teas-yang-te-topo] and the TE tunnel YANG model [I-D.ietf-teas-yang-tel].

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

Table of Contents

1. Modeling Considerations.....	3
1.1. TE Topology Model.....	3
1.2. TE Topology Modeling Constructs.....	5
1.3. Abstract TE Topology Calculation, Configuration and Maintenance.....	22
1.3.1. Single-Node Abstract TE Topology.....	24
1.3.2. Full Mesh Link Abstract TE Topology.....	26
1.3.3. Star-n-Spokes Abstract TE Topology.....	28
1.3.4. Arbitrary Abstract TE Topology.....	29
1.3.5. Customized Abstract TE Topologies.....	30
1.3.6. Hierarchical Abstract TE Topologies.....	31
1.4. Merging TE Topologies Provided By Multiple Providers.....	32
1.4.1. Dealing With Multiple Abstract TE Topologies Provided By The Same Provider.....	35
1.5. Configuring Abstract TE Topologies.....	37
1.6. TE Tunnel Model.....	38
1.7. TE Tunnel/Transport Service Modeling Constructs.....	40
1.8. Transport Service Mapping.....	54
1.9. Multi-Domain Transport Service Coordination.....	55
2. Use Cases.....	59

2.1. Use Case 1. Transport service control on a single layer multi-domain transport network.....	59
2.2. Use Case 2. End-to-end TE tunnel control on a single layer multi-domain transport network.....	67
2.3. Use Case 3. Transport service control on a ODUk/Och multi-domain transport network with Ethernet access links.....	71
2.4. Use Case 4. Transport service control on a ODUk/Och multi-domain transport network with multi-function access links.....	78
2.5. Use Case 5. Real time updates of IP/MPLS layer TE link attributes that depend on supporting transport connectivity (e.g. transport SRLGs, propagation delay, etc.).....	81
2.6. Use Case 6. Virtual Network Service.....	82
3. Security Considerations.....	85
4. IANA Considerations.....	86
5. References.....	86
5.1. Normative References.....	86
5.2. Informative References.....	86
6. Acknowledgments.....	86
Appendix A. Data Examples.....	87
A.1. Use Case 1.....	87
A.1.1. Domain 1.....	87
A.1.2. Domain 2.....	94
A.1.3. Domain 3.....	100
Authors' Addresses.....	106

1. Modeling Considerations

1.1. TE Topology Model

The TE Topology Model is written in YANG modeling language. It is defined and developed by the IETF TEAS WG and is documented as "YANG Data Model for TE Topologies" [I-D.ietf-teas-yang-te-topo]. The model describes a TE network provider's Traffic Engineering data store as it is seen by a client. It allows for the provider to convey to each of its clients:

- o information on network resources available to the client in the form of one or several native TE topologies (for example, one for each layer network supported by the provider);
- o one or several abstract TE topologies, customized on per-client basis and sorted according to the provider's preference as to how the abstract TE topologies are to be used by the client;
- o updates with incremental changes happened to the previously provided abstract/native TE topology elements;

- o updates on telemetry/state information the client has expressed interest in;
- o overlay/underlay relationships between the TE topologies provided to the client (e.g. TE path computed in an underlay TE topology supporting a TE link in an overlay TE topology);
- o client/server inter-layer adaptation relationships between the TE topologies provided to the client in the form of TE inter-layer locks or transitional links;

The TE Topology Model allows a network client to:

- o (Re-)configure/negotiate abstract TE topologies provided to the client by a TE network provider, so that said abstract TE topologies optimally satisfy the client's needs, constraints and optimization criteria, based on the client's network planning, service forecasts, telemetry information extracted from the network, previous history of service provisioning and performance monitoring, etc.;
- o Obtain abstract/native TE topologies from multiple providers and lock them horizontally (inter-domain) and vertically (inter-layer) into the client's own native TE topologies;
- o Configure, with each provider the trigger, frequency and contents of the TE topology update notifications;
- o Configure, with each provider the trigger, frequency and contents of the TE topology telemetry (e.g. statistics counters) update notifications.

1.2. TE Topology Modeling Constructs

Figure 1. TE Topology

- o TE domain - a multi-layer traffic engineered network under direct and complete control of a single authority, network provider. TE domain can be described by one or more TE topologies. For example, separate TE topologies can describe each of the domain's layer networks. TE domain can hierarchically encompass/parent other (child) TE domains, and can be encompassed by its own parent.
- o TE topology - a graphical representation of a TE domain. TE topology is comprised of TE nodes (TE graph vertices) interconnected via TE links (TE graph edges).

```

/* TE topology */
augment /nw:networks/nw:network:
  /* TE topology global ID */
  +--rw provider-id?      te-types:te-global-id
  +--rw client-id?       te-types:te-global-id
  +--rw te-topology-id?   te-types:te-topology-id
  .....
  /* TE topology general parameters */
  |   +--rw preference?          uint8
  |   +--rw optimization-criterion? identityref
  .....

```



```
        /* TE topology list of TE nodes */
augment /nw:networks/nw:network/nw:node:
  +--rw te-node-id?   te-types:te-node-id
.....
        /* TE topology list of TE links */
augment /nw:networks/nw:network/nt:link:
.....
        /* TE topology list of TE link termination points */
augment /nw:networks/nw:network/nw:node/nt:termination-point:
  +--rw te-tp-id?   te-types:te-tp-id
.....
```

Figure 2. TE Node

- o TE node - an element of a TE topology (appears as a vertex on TE graph). A TE node represents one or several nodes (physical switches), or a fraction of a node. A TE node belongs to and is fully defined in exactly one TE topology. A TE node is assigned a TE topology scope-unique ID. TE node attributes include information related to the data plane aspects of the associated node(s) (e.g. TE node's connectivity matrix), as well as configuration data (such as TE node name). A given TE node can be reached on the TE graph, representing the TE topology, over one of TE links terminated by the TE node.

```

/* TE node */
augment /nw:networks/nw:network/nw:node:
  /* TE node ID */
  +--rw te-node-id?   te-types:te-node-id
  .....
  /* TE node general attributes */
  | +--rw te-node-attributes */
  .....
  /* TE node connectivity matrices */
  | +--rw connectivity-matrices
  .....
  /* TE node underlay TE topology */
  | +--rw underlay-topology {te-topology-hierarchy}?
  | +--rw network-ref?   leafref
  .....
  /* TE node information sources*/
  | +--ro information-source-entry* [information-source]
  .....
  /* TE node statistics */
  +--ro statistics
  .....
  /* TE node TTP list */
  +--rw tunnel-termination-point* [tunnel-tp-id]
  .....

```

```

/* TE link */
augment /nw:networks/nw:network/nt:link:
/* TE link bundle information */
|   +--rw (bundle-stack-level)?
|   |   +--rw bundled-links
|   |   +--rw component-links
|
.....
/* TE link general attributes */
|   +--rw te-link-attributes
|
.....
/* TE link underlay TE topology */
|   +--rw underlay! {te-topology-hierarchy}?
|   |   +--rw primary-path
|   |   +--rw backup-path* [index]
|
.....
/* TE link layer network */
|   +--rw interface-switching-capability* [switching-
capability encoding]
|
.....
/* TE link protection type */
|   |   +--rw protection-type?   uint16
|
.....
/* TE link supporting TE tunnels */
|   |   +--rw tunnels

```

```

.....
/* TE link transitional link flag */
|   +--ro is-transitional?           empty

.....
/* TE link information sources */
|   +--ro information-source?       te-info-source

.....
/* TE link statistics */
|   +--ro statistics

.....

```

- o Intra-domain TE link - TE link connecting two TE nodes within the same TE topology representing a TE network domain (e.g. L14 in Figure 1). From the point of view of the TE topology where the intra-domain TE link is defined, the TE link is close-ended, that is, both local and remote TE nodes of the link are defined in the same TE topology.
- o Inter-domain TE link - TE link connecting two border TE nodes that belong to separate TE topologies describing neighboring TE network domains (e.g. L3x in Figure 1). From the point of view of the TE topology where the inter-domain TE link is defined, the TE link is open-ended, that is, the remote TE node of the link is not defined in the TE topology where the local TE node and the TE link itself are defined.

[Note: from the point of view of a TE node terminating an inter-domain TE link there is no difference between inter-domain and access TE links]

- o Access TE link - TE link connecting a border TE node of a TE topology describing a TE network domain to a TE node of a TE topology describing a customer network site (e.g. L1x in Figure 1). From the point of view of the TE topology where the access TE link is defined, the TE link is open-ended, that is, the remote TE node of the link (t.e. TE node representing customer network element(s)) is not defined in the TE topology where the local TE node and the TE link itself are defined.

[Note: from the point of view of a TE node terminating an access TE link there is no difference between access and inter-domain TE links]

- o Dynamic TE link - a TE link that shows up in (and disappears from) a TE topology as a result of multi-layer traffic engineering. Dynamic TE link (supported by a hierarchy TE tunnel dynamically set up in a server layer network) is automatically (i.e. without explicit configuration request) added to a client layer network TE topology to augment the topology with additional flexibility to ensure successful completion of the path computation for and provisioning of a client layer network connection/LSP. For example, an ODUk hierarchy TE tunnel can support a dynamic Ethernet layer TE link to enable provisioning of an Ethernet layer connection on a network that does not have sufficient static Ethernet layer connectivity. Likewise, dynamic TE link is automatically removed from the TE topology (and its supporting hierarchy TE tunnel released) as soon as the TE link stops carrying client layer connections/LSPs.
- o TE link termination point (LTP) - a conceptual point of connection of a TE node to one of the TE links terminated by the TE node (see Figure 2a). Unlike TE link, LTP is bi-directional - an inbound TE link and an oppositely directed outbound TE link have to be connected to the TE node via the same LTP to constitute a bi-directional TE link combination.

Figure 2a. Bi-directional TE link combination (left), independent uni-directional TE links (right)

```

/* LTP */
augment /nw:networks/nw:network/nw:node/nt:termination-point:
/* LTP ID */
  +--rw te-tp-id?    te-types:te-tp-id
/* LTP network layer ID */
  | +--rw interface-switching-capability* [switching-
capability encoding]
  | | +--rw switching-capability    identityref

```

```

| | +--rw encoding                               identityref
/* LTP bandwidth information */
| | +--rw max-lsp-bandwidth* [priority]
| |   +--rw priority      uint8
| |   +--rw bandwidth?    te-bandwidth
/* LTP inter-layer locks */
| +--rw inter-layer-lock-id?                      uint32
.....

```

- o TE tunnel termination point (TTP) - an element of TE topology representing one or several potential TE tunnel termination/adaptation points (e.g. OCh layer transponder). A TTP is hosted by exactly one TE node (see Figure 2). A TTP is assigned a TE node scope-unique ID. Depending on the TE node's internal constraints, a given TTP hosted by the TE node could be accessed via one, several or all TE links originated/terminated from/by the TE node. TTP's important attributes include Local Link Connectivity List, Adaptation Client Layer List, TE inter-layer locks (see below), Unreserved Adaptation Bandwidth (announcing the TTP's remaining adaptation resources sharable between all potential client LTPs), and Property Flags (indicating miscellaneous properties of the TTP, such as capability to support 1+1 protection for a TE tunnel terminated on the TTP).

```

/* TTP */
+--rw tunnel-termination-point* [tunnel-tp-id]
/* TTP ID */
+--rw tunnel-tp-id                               binary
/* TTP layer network ID */
| +--rw switching-capability?                     identityref
| +--rw encoding?                                 identityref
/* Inter-layer-locks supported by TTP */
| +--rw inter-layer-lock-id?                      uint32
/* TTP's protection capabilities */
| +--rw protection-type?                          identityref
/* TTP's list of client layer users */
| +--rw client-layer-adaptation
.....
/* TTP's Local Link Connectivity List (LLCL) */

```

```
|  +-rw local-link-connectivities
.....

```

o Label - in the context of circuit switched layer networks identifies a particular resource on a TE link (e.g. Och wavelength, ODUk container)

```
+-:-(label)
  +-rw value?   rt-types:generalized-label
```

Figure 3. TTP Local Link Connectivity List

- o TTP basic local link connectivity list (basic LLCL) - a list of TE link/label combinations terminated by the TTP-hosting TE node (effectively the same as LTP/label pairs), which the TTP could be connected to (see Figure 3, upper left). From the point of view of a potential TE path, basic LLCL provides a list of permissible LTP/label pairs the TE path needs to start/stop on for a connection, taking the TE path, to be successfully terminated on the TTP in question.
- o TTP detailed local link connectivity list (detailed LLCL) - basic LLCL extended to provide a set of costs (such as intra-node summary TE metric, delay, SRLGs, etc.) associated with each LLCL entry (see Figure 3, upper right)

```

/* TTP LLCL */
|  +--rw local-link-connectivities
|  |  +--rw number-of-entries?          uint16
|  /* LLCL entry */
|
|  /* LLCL entry LTP */
|  |  +--rw link-tp-ref                  leafref
|
.....

/* LLC entry label range */
|  +--rw label-restriction* [inclusive-exclusive label-start]
|  |  +--rw inclusive-exclusive          enumeration
|  |  +--rw label-start                  rt-types:generalized-label
|  |  |  +--rw label-end?                rt-types:generalized-
label
|  |  |  +--rw range-bitmap?              binary
|
.....

/* LLCL entry underlay TE path(s) */
|  +--rw underlay! {te-topology-hierarchy}?
|  |  +--rw primary-path
|  |  +--rw backup-path* [index]
/* LLCL entry protection type */
|  |  +--rw protection-type?            uint16
/* LLCL entry supporting TE tunnels */
|  |  +--rw tunnels
/* LLCL entry bandwidth parameters */
|  |  +--rw max-lsp-bandwidth* [priority]
|
.....

```



```

/* LLCL entry metrics (vector of costs) */
|   +--rw te-default-metric?          uint32
|   +--rw te-delay-metric?            uint32
|   +--rw te-srlgs
|   |   +--rw value*    te-types:srlg
|   +--rw te-nsrlgs {nsrlg}?

```

```

.....
/* LLCL entry ID */
|   |   +--rw id*    uint32

```

- o TTP adaptation client layer list - a list of client layers that could be directly adopted by the TTP. This list is necessary to describe complex multi-layer (more than two layer) client-server layer hierarchies and, in particular, to identify the position of the TTP in said hierarchies.

```

/* TTP adaptation client layer list */
|   +--rw client-layer-adaptation
|   |   +--rw switching-capability* [switching-capability
encoding]
|   |   /* Client layer ID */
|   |   |   +--rw switching-capability    identityref
|   |   |   +--rw encoding                identityref
|   |   /* Adaptation bandwidth available for the client layer */
|   |   +--rw bandwidth?                  te-bandwidth

```

Figure 4. TE Node Connectivity Matrix

- o TE node basic connectivity matrix - a TE node attribute describing the TE node's switching capabilities/limitations in the form of permissible switching combinations of the TE node's LTP/label pairs (see Figure 4, upper left). From the point of view of a potential TE path arriving at the TE node at a given inbound LTP/label, the node's basic connectivity matrix describes permissible outbound LTP/label pairs for the TE path to leave the TE node.
- o TE node detailed connectivity matrix - TE node basic connectivity matrix extended to provide a set of costs (such as intra-node summary TE metric, delay, SRLGs, etc.) associated with each connectivity matrix entry (see Figure 4, upper right).

```

/* TE node connectivity matrix */
    |--rw connectivity-matrix* [id]
    |--rw id                               uint32
    |--rw from /* left LTP */

```

```

|         |  +--rw tp-ref?    leafref
|         |  +--rw to        /* right LTP */
|         |  +--rw tp-ref?    leafref
|         |  +--rw is-allowed?                boolean

/* Connectivity matrix entry label range */
|         |  +--rw label-restriction* [inclusive-exclusive
label-start]
|         |  +--rw inclusive-exclusive        enumeration
|         |  +--rw label-start                rt-
types:generalized-label
|         |  +--rw label-end?                rt-
types:generalized-label
|         |  +--rw range-bitmap?              binary

/* Connectivity matrix entry underlay TE path(s) */
|         |  +--rw underlay! {te-topology-hierarchy}?
|         |  +--rw primary-path
|         |  +--rw backup-path* [index]
/* Connectivity matrix entry protection type */
|         |  +--rw protection-type?    uint16
/* Connectivity matrix entry supporting TE tunnels */
|         |  +--rw tunnels
/* Connectivity matrix entry bandwidth parameters */
|         |  +--rw max-lsp-bandwidth* [priority]

.....
/* Connectivity matrix entry metrics (vector of costs) */
|         |  +--rw te-default-metric?    uint32
|         |  +--rw te-delay-metric?      uint32
|         |  +--rw te-srlgs
|         |  |  +--rw value*    te-types:srlg
|         |  +--rw te-nsrlgs {nsrlg}?

.....
/* Connectivity matrix entry ID */
|         |  +--rw id*    uint32

```

Figure 5. TE Path

- o TE path - an ordered list of TE node/link IDs (each possibly augmented with labels) that interconnects over a TE topology a pair of TTPs and could be used by a connection (see Figure 5). A TE path could, for example, be a product of a successful path computation performed for a given TE tunnel

```

/* TE path */

/* TE topology the path is defined in */
| | | +--rw network-ref?    leafref
/* Path type (IRO, XRO, ERO, RRO) */
| | | +--rw path-type?     identityref

/* TE path elements */
| | | +--rw path-element* [path-element-id]
| | | |   +--rw path-element-id    uint32
| | | |   +--rw index?             uint32
| | | |   +--rw (type)?
/* Numbered TE link path element */
| | | |   +--:(ip-address)
| | | | |   +--rw ip-address-hop
| | | | |   +--rw address?         inet:ip-address

```

```

/* AS number path element */
    +---rw hop-type?      te-hop-type
    +---:(as-number)
        +---rw as-number-hop
        +---rw as-number?  binary
        +---rw hop-type?   te-hop-type
/* Unnumbered TE link path element */
    +---:(unnumbered-link)
        +---rw unnumbered-hop
        +---rw te-node-id?    inet:ip-address
        +---rw tp-id?         uint32
        +---rw hop-type?      te-hop-type
/* Label path element */
    +---:(label)
        +---rw label-hop
        +---rw value?         rt-types:generalized-label
        +---rw direction?     boolean
    +---:(sid)
        +---rw sid-hop
        +---rw sid?           rt-types:generalized-label

```

- o TE path segment - a contiguous fragment of a TE path

Figure 6. TE Inter-Layer Lock

- o TE inter-layer lock - a modeling concept describing client-server layer adaptation relationships important for multi-layer traffic engineering. It is an association of M client layer LTPs and N server layer TTPs, within which data arriving at any of the client layer LTPs could be adopted onto any of the server layer TTPs. A TE inter-layer lock is identified by inter-layer lock ID, which is unique across all TE topologies provided by the same provider. The client layer LTPs and the server layer TTPs associated by a given TE inter-layer lock share the same inter-layer lock ID value.

In Figure 6 a TE inter-layer lock IL_1 associates six client layer LTPs (C_LTP_1 - C_LTP_6) with two server layer TTPs (S_TTP_1 and S_TTP_2). As mentioned, they all have the same attribute -inter-layer lock ID: IL_1, which is the only parameter/value indicating the association. A given LTP may have zero, one or more inter-layer lock IDs. In the case of multiple inter-layer lock IDs, this implies that the data arriving at the LTP can be adopted onto any of TTPs associated with all specified inter-layer locks. For example, C_LTP_1 may be attributed with two inter-layer locks- IL_1 and IL_2. This would mean that C_LTP_1 for adaptation purposes can use not just TTPs associated with inter-layer lock IL_1 (i.e. S_TTP_1 and S_TTP_2 in the Figure), but any of TTPs associated with inter-layer lock IL_2. Likewise, a given TTP may have one or more inter-layer locks, meaning that it can offer the adaptation service to any client layer LTP having an inter-layer lock matching one of its own.

LTPs and TTPs associated within the same TE inter-layer lock may be hosted by the same (hybrid, multi-layer) TE node or by multiple TE nodes defined in the same or separate TE topologies. The latter case is especially important because TE topologies of different layer networks could be modeled by separate augmentations of the basic (common to all layers) TE topology model.

```
|  +--rw inter-layer-lock-id?          uint32
```

- o Transitional link - an alternative method of modeling of client-server adaptation relationship. Transitional link is a bi-directional link connecting an LTP in a client layer to an LTP in a server layer, which is associated (via TTP's LLCL) with a server layer TTP capable of adopting of the client layer data onto a TE tunnel terminated by the TTP. Important attributes of a transitional link are local/remote LTP IDs, TE metric and available adaptation bandwidth.

Figure 7. Native and Abstract TE Topologies

- o Native TE topology - a TE topology as it is known (to full extent and unmodified) to the TE topology provider (see lower part of Figure 7.). A native TE topology might be discovered via various routing protocols and/or subscribe/publish techniques. For example, a first-level TE topology provider (such as a T-SDN Domain Controller, DC) may auto-discover its native TE topology(ies) by participating in the domain's OSPF-TE protocol instance; while a second-level TE topology provider (such as a Hierarchical T-SDN Controller, HC) normally builds its native TE topology(ies) based on TE topologies exposed by each of the subordinate, first-level TE topology providers.
- o Underlay TE topology - a TE topology that serves as a base for constructing overlay TE topologies.

- o Overlay TE topology - a TE topology constructed based on one or more underlay TE topologies. Each TE node of the overlay TE topology represents a separate underlay TE topology (that could be mapped onto an arbitrary segment of a native TE topology). Each TE link of the overlay TE topology represents, generally speaking, an arbitrary TE path in one of the underlay TE topologies. The overlay TE topology and the supporting underlay TE topologies may represent separate layer networks (e.g. OTN/ODUk and WDM/OCh respectively) or the same layer network.
- o Abstract TE topology - an overlay TE topology created by a provider to describe its network in some abstract way. An abstract TE topology contains at least one abstract TE topology element, such as TE node or TE link. An abstract TE topology is built based on contents of one or more of the provider's native TE topologies (serving as underlay(s)), the provider's policies and the client's preferences (see upper part of Figure 7).
- o Customized TE topology - a TE topology tailored for a given provider's client. A customized TE topology is usually but not always an abstract TE topology. For example, a given abstract TE topology could be exposed to a group or all provider's clients (in which case the abstract TE topology is not a customized TE topology). Likewise, a given naive TE topology could be customized for a given client (for example, by removing high delay TE links the client does not care about). So customized TE topology is not an abstract TE topology, because it does not contain abstract TE topology elements
- o TE inter-domain plug - a TE link attribute meaningful for open-ended inter-domain/access TE links. It contains a network-wide unique value (inter-domain plug ID) that identifies in the network a connectivity supporting the inter-domain/access TE link in question. It is expected that a given pair of neighboring domain TE topologies (provided by separate providers) will have each at least one open-ended inter-domain/access TE link with a TE inter-domain plug matching to one provided by its neighbor, thus allowing for a client of both domains to identify adjacent nodes in the separate neighboring TE topologies and resolve the open-ended inter-domain/access TE links by connecting them regardless of the links respective local/remote node ID/link ID attributes. Inter-domain plug IDs may be assigned and managed by a central network authority. Alternatively, inter-domain plug IDs could be dynamically auto-discovered (e.g. via LMP protocol).


```

+--rw external-domain
|   +--rw network-ref?          leafref
|   +--rw remote-te-node-id?    te-types:te-node-id
|   +--rw remote-te-link-tp-id? te-types:te-tp-id
|   +--rw plug-id?              uint32

```

1.3. Abstract TE Topology Calculation, Configuration and Maintenance

The TE Topology Model does not prescribe what and how abstract TE topologies are computed, configured, manipulated and supported by a TE network (e.g. transport network) provider. However, it is assumed that:

- o All TE topologies, native or abstract, conveyed to the same or different clients, are largely independent one from another. This implies that each TE topology, generally speaking, has an independent name space for TE node and link IDs, SRLGs, etc. (possibly overlapping with the name spaces of other TE topologies);
- o All abstract TE topologies are bound to the respective underlay native or abstract TE topologies only by the overlay/underlay relationships defined by the TE Topology Model, but, otherwise, the abstract TE topologies are decoupled from their respective underlay TE topologies.

It is envisioned that an original set of abstract TE topologies is produced by a TE network provider for each of its clients based on the provider's local configurations and/or policies, as well as the client-specific profiles. The original set of abstract TE topologies offered to a client may be accepted by the client as-is. Alternatively, the client may choose to negotiate/re-configure the abstract TE topologies, so that the latter optimally satisfy the client's needs. In particular, for each of the abstract TE topologies the client may request adding/removing TE nodes, TE links, TTPs and/or modifying re-configurable parameters of the existing components. The client may also request different optimization criteria as compared to those used for the original abstract TE topology optimization, or/and specify various topology-level constraints. The provider may accept or reject all or some abstract TE topology re-configuration requests. Hence, the abstract TE topology negotiation process may take multiple iterations before the provider and each of its clients agree upon a set of abstract TE

topologies and their contents. Furthermore, the negotiation process could be repeated over time to produce new abstract TE topologies optimal to best suit evolving circumstances.

Figure 8. Native Transport Network Domain TE Topology as an Underlay for Abstract TE Topologies

Let's assume that a native transport network domain TE topology to be as depicted in Figure 8. The popular types of abstract TE topologies based on this native TE topology as an underlay are described in the following sections.

1.3.1. Single-Node Abstract TE Topology

Figure 9. Blocking/Asymmetrical TE Node with Basic Connectivity Matrix Attribute

In Figure 9, the transport network domain is presented to a client as a one-node abstract TE topology, where the single TE node (AN1) represents the entire domain and terminates all of the inter-domain/access TE links connecting the domain to its adjacent domains (i.e. TE links L1...L8). Because AN1 represents the entire domain the node's Underlay TE Topology attribute matches the ID of one of the domain's native TE topologies (e.g. one presented in Figure 8). [Note: all or some of the underlay TE topologies a given abstract TE topology depends on could be catered to the client by the provider along with the abstract TE topology in question or upon separate request(s) issued by the client.]

One important caveat about abstract TE node AN1 is that it should be considered as an asymmetrical/blocking switch, because, generally speaking, it is not guaranteed that a suitable TE path exists between any given pair of inter-domain TE links into/out of the domain. This means from the TE Topology model point of view that there are certain limitations as to how AN1's LTPs could be interconnected inside/across the TE node. The model allows for asymmetrical/blocking switches by specifying for the associated TE nodes a non-empty basic connectivity matrix attribute describing permissible inbound-outbound TE link/label switching combinations. It is assumed that the provider's path computer can compute a set of optimal TE paths, connecting inbound TE link/label_x <=> outbound TE link/label_y combinations inside the abstract TE node over the TE node's underlay TE topology. Based on the results of such computations, AN1's

connectivity matrix can be (re-)generated and (re-)conveyed to the abstract TE topology client.

A richer version of the basic connectivity matrix is the detailed connectivity matrix. The latter not only describes permissible inbound TE link/label_x <=> TE link/label TE link/label_y switching combinations, but also provides connectivity matrix entry specific vectors of various costs/metrics (in terms of delay, bandwidth, intra-node SRLGs and summary TE metrics) that a potential TE path will accrue, should a given connectivity matrix entry be selected by the path for crossing the TE node (see Figure 10).

Figure 10. Blocking/Asymmetrical TE Node with Detailed Connectivity Matrix Attribute

1.3.2. Full Mesh Link Abstract TE Topology

Figure 11. Full Mesh Link Abstract TE Topology

In Figure 11, the transport network domain is abstracted in the following way.

- o Each of the underlay native TE topology border TE nodes (i.e., the TE nodes terminating at least one inter-domain/access TE link, such as TE nodes S3 or S11 in Figure 8) is represented in the abstract TE topology as a separate abstract TE node, matching one-for-one to the respective border TE node of the underlay TE topology. For example, S3' of the abstract TE topology represents S3 of the underlay TE topology in Figure 8. [Note that such a relationship is modeled via Supporting Node attribute of TE node S3' specifying the ID of S3, as well as the ID of the TE topology where S3 is defined (i.e. TE topology in Figure 8)]. Likewise, S9' represents S9, S11' represents S11 and so forth;

- o TE nodes S3', S5', S8', S9' and S11' are interconnected via a full mesh of abstract TE links. It is assumed that the provider's path computer can compute a set of optimal TE paths over one or more of underlay TE topologies (such as presented in Figure 8)- one for each of said abstract TE links; and the provider can set up the TE tunnels in the network supporting each of the abstract TE links, either during the abstract TE topology configuration (in the case of committed/pre-established abstract TE links), or at the time the first client's connection is placed on the abstract TE link in question (the case of uncommitted abstract TE links). [Note that so (re-)computed TE paths, as well as the IDs of respective underlay TE topologies used for their computation are normally catered to the client in the Underlay TE path attribute of the associated abstract TE links]

The configuration parameters of each of the abstract TE links (such as layer ID, bandwidth and protection requirements, preferred TE paths across the underlay TE topology for the primary and backup connections, etc.) are expected to be found in the abstract TE topology profiles/templates locally configured with the provider or pushed to the provider by the client via the policy NBI. Each of the abstract TE links may be later re-configured or removed by direct configuration requests issued by the client via TE Topology NBI. Likewise, additional abstract TE links may be requested by the client at any time.

Some possible variants/flavors of the Full Mesh Link Abstract TE Topology described above are:

- o Partial Mesh Link Abstract TE Topology (where some of the abstract TE links from the full mesh are missing);
- o Double Mesh Link Abstract TE Topology (where each pair of abstract TE nodes is connected via two diverse abstract TE links).

1.3.3. Star-n-Spokes Abstract TE Topology

Figure 12. Star-n-Spoke Abstract TE Topology

The Full Mesh Link Abstract TE Topology suffers from the n -squared problem; that is, the number of required abstract TE links is proportional to square of the number of native TE topology border TE nodes. This problem can be mitigated (i.e., the number of required abstract TE links may be significantly reduced) by adding, to the abstract TE topology, an additional abstract TE node (the star) representing one or several interconnected non-border TE nodes from the native TE topology. Abstract TE links in the Star-n-Spokes Topology connect the star with all other TE nodes of the topology (the spokes). For example, abstract TE node AN1 in Figure 12 could represent collectively TE nodes S7, S10 and S4 of the native TE topology (see Figure 8) with abstract TE links connecting AN1 with all other TE nodes in the Star-n-Spokes Abstract TE Topology in Figure 12.

In order to introduce a composite abstract TE node, (e.g. AN1 in Figure 12) representing in a given abstract TE topology an arbitrary segment of another TE topology (e.g. TE nodes S7, S12 and S4 of the TE topology in Figure 8) the TE topology provider is expected to perform the following operations:

- o Copy the TE topology segment to be represented by the abstract TE node (i.e. TE nodes S7, S10 and S4 in Figure 8, as well as the TE links interconnecting them) into a separate auxiliary TE topology (with a separate TE topology ID);

- o Set for each TE node and TE link of the auxiliary TE topology the Supporting Node/Link attribute matching the original TE topology ID, as well as the ID of the respective original TE node/link of the original TE topology. For example, if S7" of the auxiliary TE topology is a copy of S7 of the original TE topology, the Supporting Node attribute of S7" will specify the ID of the original TE topology (presented in figure 8) and the ID of S7;
- o Set for the abstract TE node AN1 the Underlay TE Topology attribute matching the auxiliary TE Topology ID

Furthermore, the Star-n-Spokes Abstract TE topology provider is expected to:

- o Compute/provision TE paths/tunnels supporting each of the abstract TE links in Figure 12 (i.e. abstract TE links connecting the spokes to the star, AN1) as described in 1.3.2;
- o Generate the AN1's Basic/Detailed Connectivity Matrix attribute based on intra-node path computations performed on the AN1's underlay (i.e. auxiliary) TE topology and describing permissible inbound TE link/label_x. outbound TE link/label_y switching combinations as described in 1.3.1

1.3.4. Arbitrary Abstract TE Topology

Figure 13. Arbitrary Abstract TE Topology

To achieve an optimal tradeoff between the number of components, the amount of information exposed by a transport network provider and the

amount of path computations required to keep said information up-to-date, the provider may present the TE network domain as an arbitrary abstract TE topology comprised of any number of abstract TE nodes interconnected by abstract TE links (see Figure 13). Each of the abstract TE nodes can represent a single or several interconnected TE nodes from the domain's underlay (native or lower level abstract) TE topology, or a fraction of an underlay TE node. [Note that each of the abstract TE nodes of the TE topology in Figure 13 is expected to be introduced and maintained by the provider following the instructions as described in 1.3.3; likewise, each of the abstract TE links of the topology is expected to be computed, provisioned and maintained as described in 1.3.2]

1.3.5. Customized Abstract TE Topologies

Figure 14. Customized Abstract TE Topology(ies)

A transport network/domain provider may serve more than one client. In such a case, the provider "slices" the network/domain resources and exposes a slice for each of the clients in the form of a customized abstract TE topology. In Figure 14, the provider serves

two clients (Blue and Red). Client Blue is provided with the Blue abstract TE topology supported by the blue TE tunnels or paths in the underlay (native) TE topology (depicted in the Figure with blue broken lines). Likewise, client Red is provided with the Red abstract TE topology supported by the red TE tunnels or paths in the underlay TE topology.

1.3.6. Hierarchical Abstract TE Topologies

Figure 15. Hierarchy of Abstract TE Topologies

As previously mentioned, an underlay TE topology for a given abstract TE topology component does not have to be one of the domain's native TE topologies - another (lower level) domain's abstract TTE topology can be used instead. This means that abstract TE topologies are hierarchical in nature.

Figure 15 provides an example of abstract TE topology hierarchy. In this Figure the blue topology is a top level abstract TE topology catered to by the provider to one of the domain's clients. One of the TE links of the blue topology - link EF - is supported by a TE path E'-M-P-Q-N-F' computed in the underlay TE topology (red topology), which happens to be domain's (lower level) abstract TE topology.. Furthermore, as shown, the TE link PQ - one of the TE links comprising the E'-M-P-Q-N-F' path - is supported by its own underlay

TE path, P'-X-Q' - computed on one of the domain's native TE topologies.

Importantly, each TE link and TE node of a given abstract TE topology has, generally speaking, its individual stack/hierarchy of underlay TE topologies.

1.4. Merging TE Topologies Provided By Multiple Providers

A client may receive TE topologies provided by multiple providers, each of which managing a separate domain of an interconnected multi-domain transport network. In order to make use of said topologies, the client is expected to merge (inter-connect) the provided TE topologies into one or more client's native TE topologies, each of which homogeneously representing the multi-domain transport network. This makes it possible for the client to select end-to-end TE paths for its TE tunnel connections traversing multiple domains.

In particular, the process of merging TE topologies includes:

- o Identifying neighboring TE domains and locking their TE topologies horizontally by connecting their inter-domain open-ended TE links;
- o Renaming TE node, link, and SRLG IDs into ones allocated from a separate name space; this is necessary because all TE topologies are considered to be, generally speaking, independent with a possibility of clashes among TE node, link or SRLG IDs. Original TE node/link IDs along with the original TE topology ID are stored in the Source attribute of the respective TE nodes/links of the merged TE topology;
- o Locking, TE topologies associated with different layer networks vertically according to provided TE inter-layer locks; this is to facilitate inter-layer path computations across multiple TE topologies provided by the same topology provider.

Figure 16. Merging Domain TE Topologies

Figure 16 illustrates the process of merging, by the client, of TE topologies provided by the client's providers.

In the Figure, each of the two providers caters to the client a TE topology (abstract or native), describing the network domain under the respective provider's control. The client, by consulting the attributes of the open-ended inter-domain/access TE links - such as TE inter-domain plugs or remote TE node/link IDs - is able to determine that:

1. the two domains are adjacent and are interconnected via three inter-domain TE links, and;
2. each domain is connected to a separate customer site, connecting the left domain in the Figure to customer devices C-11 and C-12, and the right domain to customer devices C-21, C-22 and C-23.

Therefore, the client interconnects the open-ended TE links, as shown on the upper part of the Figure.

As mentioned, one way to interconnect the open-ended inter-domain/access TE links of neighboring domains is to mandate the providers to specify remote nodeID/linkID attributes in the provided inter-domain/access TE links. This, however, may prove to be not flexible. For example, the providers may not be aware of the respective remote nodeID/linkID values. More importantly, this option does not allow for the client to mix-n-match multiple (more than one) TE topologies catered by the same providers (see the next section). Another, more flexible, option to resolve the open-ended inter-domain/access TE links is by decorating them with the TE inter-domain plug attribute. The attribute specifies inter-domain plug ID - a network-wide unique value that identifies on the network connectivity supporting a given inter-domain/access TE link. Instead of specifying remote node ID/link ID, an inter-domain/access TE link may provide a non-zero inter-domain plug ID. It is expected that two neighboring domain TE topologies (provided by separate providers) will have each at least one open-ended inter-domain/access TE link with a TE inter-domain plug matching to one provided by its neighbor. For example, the inter-domain TE link originating from node S5 of the Domain 1 TE topology (Figure 8) and the inter-domain TE link coming from node S3 of Domain2 TE topology may specify matching TE inter-domain plugs (i.e. carrying the same inter-domain plug ID). This would allow for the client to identify adjacent nodes in the separate neighboring TE topologies and resolve the inter-domain/access TE links connecting them regardless of their respective nodeIDs/linkIDs (which, as mentioned, could be allocated from independent name spaces).

Inter-domain plug IDs may be assigned and managed by a central network authority. Alternatively, inter-domain plug IDs could be dynamically auto-discovered (e.g. via LMP protocol).

Furthermore, the client renames the TE nodes, links and SRLGs offered in the abstract TE topologies by assigning to them IDs allocated from a separate name space managed by the client. Such renaming is necessary, because the two abstract TE topologies may have their own name spaces, generally speaking, independent one from another; hence, ID overlaps/clashes are possible. For example, both TE topologies have TE nodes named S7, which, after renaming, appear in the merged TE topology as S17 and S27 respectively. IDs of the original (i.e. abstract TE topology) TE nodes/links along with the ID of the abstract TE topology they belong to are stored in the Source attribute of the respective TE nodes/links of the merged TE topology. For example, the Source attribute of S27 will contain S7 and the TE topology ID of the abstract TE topology describing domain 2.

Once the merging process is complete, the client can use the merged TE topology for path computations across both domains, for example, to compute a TE path connecting C-11 to C-23.

1.4.1. Dealing With Multiple Abstract TE Topologies Provided By The Same Provider

Figure 17. Multiple Abstract TE Topologies Provided By TE Topology Providers

A given provider may expose more than one abstract TE topology to the client. For example, one abstract TE topology could be optimized based on a lowest-cost criterion, while another one could be based on best possible delay metrics, while yet another one could be based on maximum bandwidth availability for the client connections. Furthermore, the client may request all or some providers to expose additional abstract TE topologies, possibly of a different type and/or optimized differently, as compared to already-provided TE topologies. In any case, the client should be prepared for a provider to offer to the client more than one abstract TE topology.

It should be up to the client to decide how to mix-and-match multiple abstract TE topologies provided by each of the providers, as well as how to merge them into the client's native TE topologies. The client also decides how many such merged TE topologies it needs to produce and maintain. For example, in addition to the merged TE topology depicted on the upper part of Figure 16, the client may merge the abstract TE topologies received from the two providers, as shown in Figure 17, into the client's additional native TE topologies, as shown in Figure 18.

[Note: allowing for the client mix-n-matching of multiple TE topologies assumes that TE inter-domain plugs (rather than remote nodeID/linked) option is used for identifying neighboring domains and inter-domain/access TE link resolution.]

Figure 18. Multiple Native (Merged) Client's TE Topologies

It is important to keep in mind that each of the three native (merged) TE topologies could be used by the client for computing TE paths for any of the multi-domain connections. The choice as to which topology to use for a given connection depends on the connection/tunnel parameters/requirements and the topology's style and optimization criteria.

1.5. Configuring Abstract TE Topologies

When a client receives one or more abstract TE topologies from one of its providers, it may accept the topologies as-is and merge them into one or more of its own native TE topologies. Alternatively, the client may choose to request a re-configuration of one, some or all abstract TE topologies provided by the providers. Specifically, with respect to a given abstract TE topology, some of its TE nodes/links may be requested to be removed, while additional ones may be requested to be added. It is also possible that existing TE nodes/links may be asked to be re-configured. For example, a set of TE links may be requested to be disjoint from each other by configuring the same Non Sharing Risk Link Group (NSRLG) attribute for all links from the set. Such a configuration would force the provider to place TE tunnels supporting the TE links from the set onto sufficiently disjoint TE paths computed in the tunnels underlay TE topology. Furthermore, the topology-wide optimization criteria may be requested to be changed. For example, underlay TE paths supporting the abstract TE links, currently optimized to be shortest (least-cost) paths, may be requested to be re-optimized based on the minimal delay criteria. Additionally, the client may request the providers to configure entirely new abstract TE topologies and/or to remove existing ones. Furthermore, future periodic or one time additions, removals and/or re-configurations of abstract TE topology elements and/or their attributes could be (re-)scheduled by the client ahead of time.

It is the responsibility of the client to implement the logic behind the above-described abstract TE topology negotiation. It is expected that the logic is influenced by the client's local configuration/templates, policies conveyed by client's clients, input from the network planning process, telemetry processor, analytics systems and/or direct human operator commands. Figure 19 exemplifies the abstract TE topology negotiation process. As shown in the Figure, the original abstract TE topology exposed by a provider was requested to be re-configured. Specifically, one of the abstract TE links was asked to be removed, while three new ones were asked to be added to the abstract TE topology.

Figure 19. Provider. Client Abstract TE Topology Negotiation

1.6. TE Tunnel Model

The TE Tunnel Model is written in YANG modeling language. It is defined and developed by the IETF TEAS WG and is documented as "YANG Data Model for Traffic Engineering Tunnels and Interfaces" [I-D.ietf-teas-yang-te]. Among other things the model describes a TE network provider's TE Tunnel data store as it is seen and influenced by a client.

The TE Tunnel Model allows for the provider to convey to each of its clients:

- o information on TE tunnels provided to the client that are fully contained within the controlled network domain,
- o information on multi-domain TE tunnel segments across the network domain controlled by the provider;
- o information on connections/LSPs, supporting TE tunnels and TE tunnel segments;
- o updates in response to changes to the client's active TE tunnels/segments and the connections supporting them,

- o updates in response to the TE tunnel/segment telemetry/state information the client has expressed an interest in.

The TE Tunnel Model allows for a TE network client to:

- o Issue configuration requests to set up, tear down, replace, modify and manipulate end-to-end TE tunnels, as well as segments of multi-domain TE tunnels across the network controlled by the provider;
- o Request and obtain information on active TE tunnels/segments and connections supporting them;
- o Subscribe to and configure with the provider triggers, pace and contents of the TE tunnel/segment change update notifications;
- o Subscribe to and configure with the provider triggers, pace and contents of the TE tunnel/segment event notifications, such as detected alarms, faults, protection/restoration actions, etc..
- o Subscribe to and configure with the provider triggers, pace and contents of TE tunnel/segment telemetry (e.g. statistics counters) update notifications.

1.7. TE Tunnel/Transport Service Modeling Constructs

Figure 20. TE tunnel

- o TE tunnel - a connection-oriented service provided by a layer network of delivery of a client's data between source and destination tunnel termination points. A TE tunnel in a server layer network may support a link in a client layer network (e.g. OCh layer TE tunnel supporting ODU4 link). In Figure 20, a TE tunnel interconnects tunnel termination points resident on switches C-R2 and C-R3. A TE tunnel is realized via (supported by, mapped onto) one or more layer network connections/LSPs

```
/* TE tunnel */
|  +--rw tunnel* [name]
|  |  +--rw name                               leafref
```

```

| | | |--rw identifier? leafref
/* TE tunnel configuration parameters */
| | | |--rw config
| | |   |--rw name? string
| | |   |--rw type? identityref
| | |   |--rw identifier? uint16
| | |   |--rw description? string
| | |   |--rw switchcap? identityref
| | |   |--rw encoding? identityref
| | |   |--rw protection-type? identityref
| | |   |--rw admin-status? identityref
| | |   |--rw preference? uint8
| | |   |--rw reoptimize-timer? uint16
| | |   |--rw source? inet:ip-address
| | |   |--rw destination? inet:ip-address
| | |   |--rw src-tp-id? binary
| | |   |--rw dst-tp-id? binary
id | | |   |--rw topology-id? te-types:te-topology-
| | |   |--rw ignore-overload? boolean
| | |   |--rw bandwidth-generic? te-types:te-bandwidth
disjointness | | |   |--rw disjointness? te-types:te-path-
| | |   |--rw setup-priority? uint8
| | |   |--rw hold-priority? uint8
| | |   |--rw signaling-type? identityref
/* Hierarchy TE tunnel parameters */
| | |   |--rw hierarchical-link-id
| | |     |--rw local-te-node-id? te-types:te-node-id
| | |     |--rw local-te-link-tp-id? te-types:te-tp-id
| | |     |--rw remote-te-node-id? te-types:te-node-id
topology-id | | |     |--rw te-topology-id? te-types:te-
/* Bidirectional TE tunnel parameters */
| | |   |--rw bidirectional
| | |     |--rw association
| | |       |--rw id? uint16
| | |       |--rw source? inet:ip-address
| | |       |--rw global-source? inet:ip-address
| | |       |--rw type? identityref
/* TE tunnel state */
| | |   |--ro state
| | |     |--ro name? string
| | |     |--ro type? identityref
| | |     |--ro identifier? uint16

```

```

| | | +--ro oper-status? identityref
/* TE tunnel primary path and LSP container */
| | | +--rw p2p-primary-paths
| | | +--rw p2p-primary-path* [name]
| | | +--rw name
| | | /* Configuration */
leafref
| | | +--rw config
| | | +--rw name? string
| | | +--rw preference? uint8
| | | +--rw path-setup-protocol? identityref
| | | +--rw path-computation-method? identityref
| | | +--rw path-computation-server? inet:ip-
address
| | | +--rw compute-only? empty
| | | +--rw use-cspf? boolean
| | | +--rw verbatim? empty
| | | +--rw lockdown? empty
| | | +--rw named-explicit-path? leafref
| | | +--rw named-path-constraint? leafref {te-
types:named-path-constraints}?
| | | /* state */
| | | +--ro state
| | | +--ro name? string
| | | +--ro preference? uint8
| | | +--ro path-setup-protocol? identityref
| | | +--ro path-computation-method? identityref
| | | +--ro path-computation-server? inet:ip-
address
| | | +--ro compute-only? empty
| | | +--ro use-cspf? boolean
| | | +--ro verbatim? empty
| | | +--ro lockdown? empty
| | | +--ro named-explicit-path? leafref
| | | +--ro named-path-constraint? leafref
{te-types:named-path-constraints}?
| | | /* Computed path */
| | | /* Computed path properties/metrics /
| | | +--ro computed-path-properties
| | | | +--ro path-metric* [metric-type]
| | | | | +--ro metric-type identityref
| | | | | +--ro accumulative-value? uint64
| | | /* Computed path affinities */
| | | +--ro path-affinities
| | | | +--ro constraints* [usage]
| | | | | +--ro usage?
identityref

```

										+--ro (style)?	
										+---:(value)	
										+--ro value?	te-
types:admin-groups											
										+---:(named)	
[name]										+--ro affinity-names*	
										+--ro name	string
										/* Computed path SRLGs */	
										+--ro path-srlgs	
										+--ro (style)?	
										+---:(values)	
										+--ro usage?	identityref
types:srlg										+--ro values*	te-
										+---:(named)	
										+--ro constraints* [usage]	
identityref										+--ro usage	
										+--ro constraint	
										+--ro srlg-names* [name]	
										+--ro name	string
										/* Computed path sub-objects */	
										+--ro path-computed-route-objects	
										
										/* LSP (provisioned path) */	
										+--ro lsp* [source destination tunnel-id	
lsp-id extended-tunnel-id type]											
										/* LSP parameters */	
										+--ro source	leafref
										+--ro destination	leafref
										+--ro tunnel-id	leafref
										+--ro lsp-id	leafref
										+--ro extended-tunnel-id	leafref
										+--ro type	leafref
										+--ro signaling-type?	identityref
										+--rw candidate-p2p-secondary-paths	
										+--rw candidate-p2p-secondary-path*	
[secondary-path]											
										+--rw secondary-path	leafref
										+--rw config	
										+--rw secondary-path?	leafref
										+--rw priority?	uint16
identityref										+--rw path-setup-protocol?	
										+--ro state	
										+--ro secondary-path?	leafref

```

| | | |      +---ro priority?          uint16
identityref   +---ro path-setup-protocol?
| | | |      +---ro active?            boolean

/* TE tunnel secondary path and LSP container */

| | | |      +---rw p2p-secondary-paths
| | | | |    +---rw p2p-secondary-path* [name]
.....
| | | | |    +---rw name                leafref
| | | | |    +---rw config (same as for primary path )
.....
| | | | |    +---ro state (same as for primary, except for
disjointedness_state )
| | | | |    +---ro disjointness_state?        te-types:te-path-
disjointness.....
| | | | |    +---ro computed-path-properties (same as for
primary path)
.....
| | | | |    +---ro path-affinities (same as for primary
path)
.....
| | | | |    +---ro path-srlgs (same as for primary
path)
.....
| | | | |    +---ro path-computed-route-objects
.....
| | | | |    /* LSP (provisioned path) */
| | | | |    +---ro lsp (same as for the primary LSP)
.....

```

- o Tunnel termination point (TTP) - a physical device inside a given node/switch realizing a TE tunnel termination function in a given layer network, as well as the TE tunnel's adaptation function provided for client layer network(s). One example of tunnel termination point is an OCh layer transponder. [Note: Tunnel termination points are not to be confused with TE tunnel termination points, which are TE representations of physical tunnel termination points. Similar to physical switches and links of the network, such as depicted in Figure 20, being represented on a TE topology describing the network as TE nodes and TE links, (physical) tunnel termination points (TTPs) are represented as TE tunnel termination points (TE TTPs, see 1.2) hosted by the TE nodes. For example, a provisioned connection/LSP starts on a source TTP, goes through a chain of physical links and stops on a destination TTP. In contrast, TE path (e.g. result of a path computation) starts on a source TE TTP, goes through a chain of TE links and stops on a destination TE TTP.]

			---rw source?	inet:ip-address
			---rw destination?	inet:ip-address
			---rw src-tp-id?	binary
			---rw dst-tp-id?	binary

- o TE tunnel hand-off point - an access link or inter-domain link by which a multi-domain TE tunnel enters or exits a given network domain, in conjunction with a layer network resource (such as a wavelength channel or ODUk container) allocated on the access/inter-domain link for the TE tunnel.
- o TE tunnel segment - a part of a multi-domain TE tunnel that spans a given network domain and is directly and fully controlled by the domain's controller, DC. TE tunnel segment is a fragment of a multi-domain TE tunnel between
 1. the source tunnel termination point and the TE tunnel hand-off point outbound from the TE tunnel's first domain (head TE tunnel segment);
 2. inbound and outbound TE tunnel hand-off points into/from a given domain (transit TE tunnel segment);

3. inbound TE tunnel hand-off point into the TE tunnel's last domain and the destination tunnel termination point (tail TE tunnel segment);
- o Transport service - the same as TE tunnel segment
 - o Hierarchy TE tunnel - a server layer TE tunnel that supports a dynamically created TE link in the client layer network topology (e.g. see 1.2)

```

/* Hierarchy TE tunnel parameters */
      | | | | |
      | | | | | +--rw hierarchical-link-id
      | | | | | +--rw local-te-node-id?      te-types:te-node-id
      | | | | | +--rw local-te-link-tp-id?    te-types:te-tp-id
      | | | | | +--rw remote-te-node-id?      te-types:te-node-id
      | | | | | +--rw te-topology-id?         te-types:te-
topology-id

```

- o Hierarchy transport service - the first or the last segment of a multi-domain hierarchy TE tunnel
- o Dependency TE tunnel - a hierarchical TE tunnel provisioned or to be provisioned in an immediately adjacent server layer a given client layer TE tunnel depends on (i.e. carried or to be carried within)
- o Potential TE tunnel/segment - a TE tunnel/segment configured in COMPUTE_ONLY mode. For such a TE tunnel/segment TE paths to be taken by supporting connection(s) is/are computed and monitored, but the connection(s) are not provisioned

```

      | | | | | +--rw path-computation-method?  identityref
      | | | | | +--rw path-computation-server?  inet:ip-
address
      | | | | | +--rw compute-only?             empty
      | | | | | +--rw use-cspf?                 Boolean

```

Figure 20a. TE Tunnel Connections/LSPs

- o Layer network connection/connection/LSP - a layer network path supporting a TE tunnel by realizing its implied forwarding function. Said path is provisioned in a given layer network's data plane over a chain of links and cross-connected over switches terminating the links. It interconnects the supported TE tunnel's source and destination termination points (in the case of end-to-end connection) or TE tunnel's hand-off points (in the case of transport service connection) or the TE tunnel's two split-merge points (in the case of segment protection connection).

Example: ODU2 connection supporting an ODU2 TE tunnel.

				/* LSP (provisioned path) */	
				+--ro lsp* [source destination tunnel-id	
lsp-id	extended-tunnel-id	type]			
				/* LSP parameters */	
				+--ro source	leafref
				+--ro destination	leafref
				+--ro tunnel-id	leafref
				+--ro lsp-id	leafref
				+--ro extended-tunnel-id	leafref
				+--ro type	leafref
				+--ro signaling-type?	identityref
.....				
				+--ro priority?	uint16
				+--ro path-setup-protocol?	
identityref					
				+--ro active?	Boolean

- o Working connection - the primary connection of the supported TE tunnel or transport service (see Figure 20a).
- o End-to-end protection connection - a secondary end-to-end connection of the supported TE tunnel (e.g. end-to-end 1+1 protection connection, see Figure 20a).
- o Segment protection connection - a secondary connection of the supported transport service protecting the service over a given network domain (e.g. 1+1 segment protection connection, see Figure 20a)
- o Restored connection - a connection after successful network failure restoration procedures
- o Current connection - the same as restored connection
- o Nominal connection - a connection as (re-)provisioned upon a client configuration request (i.e. a connection before any automatic network failure restoration re-configurations are carried out, also a connection after restoration reversion procedures are successfully completed)
- o Unprotected TE tunnel/transport service - TE tunnel/transport service supported by a single (working/primary) connection/LSP

- o Protected TE tunnel/transport service - TE tunnel/transport service supported by one working connection/LSP and at least one protection/secondary connection/LSP
- o Restorable TE tunnel/transport service - TE tunnel/transport service with pre-configured automatic network failure restoration capabilities
- o TE tunnel/transport service automatic protection switchover - a process of switching of carrying user payload from the tunnel's/service's affected by a network failure working connection onto one of the tunnel's/service's healthy protection connection
- o TE tunnel/transport service automatic protection reversion - a process of switching of carrying user payload from the tunnel's/service's protection connection back onto the tunnel's/service's working connection after the latter was repaired from network failure
- o TE tunnel/transport service protection external command - a command, typically issued by an operator, which influences the automatic protection switchover and reversion.

External commands are defined in [ITU-T G.800] and [RFC 4427]:

- . Freeze: A temporary configuration action that prevents any switch action to be taken and as such freezes the current state.
- . Clear Freeze: An action that clears the active Freeze state.
- . Lockout of Normal: A temporary configuration action that ensures that the normal traffic is not allowed to use the protection transport entity.

As described in [ITU-T G.808], this command should be issued at both ends.

- . Clear Lockout of Normal: An action that clears the active Lockout of Normal state.
- . Lockout of Protection: A temporary configuration action that ensures that the protection transport entity is temporarily not available to transport a traffic signal (either normal or extra traffic).

- . Forced Switch: A switch action that swithes the extra traffic signal, the normal traffic signal, or the null signal to the protection transport entity, unless an equal or higher priority switch command is in effect.
 - . Manual Switch: A switch action that switches the extra traffic signal, the normal traffic signal #i, or the null signal to the protection transport entity, unless a fault condition exists on other transport entities or an equal or higher priority switch command is in effect.
 - . Exercise: An action to start testing if the APS communication is operating correctly. It is lower priority than any other state or command.
 - . Clear: An action that clears the active near-end lockout of protection, forced switch, manual switch, WTR state, or exercise command
- o TE tunnel/transport service protection Hold-off time - a configured period of time to expire between the moment of detecting of the first network failure affecting the tunnel's/service's working connection and the begining of the tunnel's/service's automatic protection switchover procedures
 - o TE tunnel/transport service protection WTR time - a configured period of time to expire between the moment of repairing the last network failure affecting the tunnel's/service's working connection and the begining of the tunnel's/service's automatic protection reversion procedures
 - o TE tunnel/transport service automatic network failure restoration - a process of replacing of the tunnel's/service's connection(s) affected by one or more network failures away from the point(s) of failue
 - o TE tunnel/transport service restoration reversion- a process of replacing of the tunnel's/service's connection(s) back onto the nominal connection paths after all network failures affecting the tunnel's/service's nominal connection(s) are repaired
 - o TE tunnel/transport service restoration Hold-off time - a configured period of time to expire between the moment of detecting of the first network failure affecting the tunnel's/service's nominal or current connection and the beginning of the automatic connection restoration procedures

- o TE tunnel/transport service restoration WTR time - a configured period of time to expire between the moment of repairing the last network failure affecting the tunnel's/service's nominal connection and the beginning of the connection automatic restoration reversion procedures
- o Configured restoration path - a TE path specified by the client to be used during the automatic network failure restoration operation on one of the TE tunnel's/transport service's nominal or current connections
- o Pre-computed restoration path - a configured restoration path to be validated by a path computer during the TE tunnel/transport service setup or client triggered modification
- o Pre-provisioned restoration path - a pre-computed restoration path to be pre-provisioned/pre-sigaled in the network (with all associated network resources allocated but not necessarily bound into cross-connects) during the TE tunnel/transport service setup or client triggered modification
- o Connection configured path - a TE path (see 1.2) over a TE topology describing a layer network/domain that specifies (loosely or strictly) the client's requirements with respect to an ordered list of network nodes, links and resources on the links a given connection should go through

```
| | +--rw explicit-route-object* [index]
| | | +--rw index leafref
| | | +--rw explicit-route-usage? identityref
(INCLUDE/EXCLUDE)
| | | +--rw index? uint32
| | | +--rw (type)?
| | | | +---:(numbered)
| | | | | +--rw numbered-hop
| | | | | +--rw address? te-types:te-tp-
id
| | | | +--rw hop-type? te-hop-type
| | | | +---:(as-number)
| | | | | +--rw as-number-hop
| | | | | +--rw as-number? binary
| | | | | +--rw hop-type? te-hop-type
| | | | +---:(unnumbered)
| | | | | +--rw unnumbered-hop
```

node-id					+++rw node-id?	te-types:te-
tp-id					+++rw link-tp-id?	te-types:te-
					+++rw hop-type?	te-hop-type
					+++:(label)	
					+++rw label-hop	
types:generalized-label					+++rw value?	rt-
					+++:(sid)	
					+++rw sid-hop	
types:generalized-label					+++rw sid?	rt-

- o Connection exclusion path - a TE path over a TE topology describing a layer network/domain that specifies the client's requirements with respect to an unordered list of network nodes, links and resources on the links to be avoided by a given connection

					+++rw route-object-exclude-always* [index]	
					+++rw index	leafref
					+++rw index?	uint32
					+++rw (type)?	
					+++:(numbered)	
					+++rw numbered-hop	
id					+++rw address?	te-types:te-tp-
					+++:(as-number)	
					+++rw as-number-hop	
					+++rw as-number?	binary
					+++:(unnumbered)	
					+++rw unnumbered-hop	
node-id					+++rw node-id?	te-types:te-
tp-id					+++rw link-tp-id?	te-types:te-
					+++:(label)	
					+++rw label-hop	
types:generalized-label					+++rw value?	rt-

```

      |           |   |   |
types:generalized-label  +--:(sid)
                        +--rw sid-hop
                        +--rw sid?   rt-

```

- o Connection computed path - a TE path over a TE topology describing a layer network/domain as computed (subject to all configured constraints and optimization criteria) for a given connection to take. Computed connection path could be thought as the TE path intended to be taken by the connection

```

/* Computed path */
/* Computed path properties/metrics /
    | | |
    | | | +---ro computed-path-properties
    | | |     +---ro path-metric* [metric-type]
    | | |         +---ro metric-type          identityref
    | | |         +---ro accumulative-value?   uint64
/* Computed path affinities */
    | | | +---ro path-affinities
    | | |     +---ro constraints* [usage]
    | | |         +---ro usage?
identityref
    | | |         +---ro (style)?
    | | |             +---:(value)
    | | |                 | +---ro value?          te-
types:admin-groups
    | | |             +---:(named)
    | | |                 +---ro affinity-names*
[name]
    | | |             +---ro name          string
/* Computed path SRLGs */
    | | | +---ro path-srlgs
    | | |     +---ro (style)?
    | | |         +---:(values)
    | | |             +---ro usage?          identityref
    | | |             +---ro values*        te-
types:srlg
    | | |         +---:(named)
    | | |             +---ro constraints* [usage]
identityref
    | | |                 +---ro usage

```



```

| | | | +--ro constraint
| | | |   +--ro srlg-names* [name]
| | | |     +--ro name      string
| | | | /* Computed path sub-objects */
| | | |   +--ro path-computed-route-objects
.....
```

- o Connection actual path - an active connection's path as provisioned in the layer network's data plane in the form of a TE path over a TE topology describing the layer network/domain

1.8. Transport Service Mapping

Figure 21. Transport Service Mapping

Let's assume that a provider has exposed to a client its network domain in the form of an abstract TE topology, as shown on the left side of Figure 21. From then on, the provider should be prepared to receive from the client, a request to set up or manipulate a transport service with TE path(s) computed for the service connection(s) based on and expressed in terms of the provided abstract TE topology (as, for example, displayed in red broken line on the right side of Figure 21). When this happens, the provider is expected to set up the TE tunnels supporting all yet uncommitted abstract TE links (e. g, TE link S3'-S8' in the Figure).

Furthermore, it is the responsibility of the provider to:

- o Perform all the necessary abstract-to-native translations for the specified TE paths (i.e. the transport service connection configured paths);

- o Provision working and protection connections supporting the transport service; as well as replace/modify/delete them in accordance with subsequent client's configuration requests;
- o Perform all the requested recovery operations upon detecting network failures affecting the transport service;
- o Notify the client about all parameter changes, events and other telemetry information the client has expressed an interest in, with respect to the transport service in question.

1.9. Multi-Domain Transport Service Coordination

A client of multiple TE network domains may need to orchestrate/coordinate its transport service setup/manipulation across some or all the domains. One example of such a client is a Hierarchical T-SDN Controller, HC, managing a connected multi-domain transport network where each of the domains is controlled by a separate Domain T-SDN Controller, DC. Said DCs are expected to expose TE Topology and TE Tunnel North Bound Interfaces, NBIs,, supported respectively by IETF TE Topology and TE Tunnel models (and their network layer specific augmentations). HC is assumed to establish client-provider relationship with each of the DCs and make use of said NBIs to extract from the domains various information (such as TE topologies and telemetry), as well as to convey instructions to coordinate across multiple domains its transport services set up and manipulation.

Figure 22. Two-Domain Transport Network

Let's consider, for example, a two-domain transport network as represented in Figure 22. Suppose that HC is requested to set up an unprotected transport service to provide connectivity between customer network elements C-R1 and C-R6. It is assumed that by the time the request has arrived, the two DCs have already provided abstract TE topologies describing their respective domains, and that HC has merged the provided TE topologies into one that homogeneously describes the entire transport network (as shown in Figure 23).

Figure 23. Two-Domain Transport Network (Abstracted View)

Consider that HC, using the merged TE topology, selected a TE path to be taken by the requested transport service connection as shown on the upper part of Figure 24.

The multi-domain transport service set up coordination includes:

- o Splitting selected for the transport service TE path(s) into segments - one set of segments per each domain involved in the service setup;
- o Issuing a configuration request to each of the involved DCs to set up the transport service across the respective domain. Note that the connection configured paths are required to be expressed in terms of respective abstract TE topologies as exposed to HC by DCs (see lower part of Figure 24).

- o Waiting for the set up complete confirmation from each of the involved DCs. In case one of the DCs reports a failure, HC is responsible to carry out the cleanup/rollback procedures by requesting all involved DCs to tear down the successfully created segments

Figure 24. Transport Service Placement Based on Abstract TE Topology

While processing the received from HC configuration request to set up the transport service, each DC is expected to carry out the transport service mapping procedures (as described in 1.8) resulting in the set up of all the necessary underlay TE tunnels, as well as one or more connections supporting the transport service. As a result, the requested transport service will be provisioned as shown in Figure 25.

The multi-domain transport service tear down coordination entails issuing to each of the involved DCs a configuration request to delete the transport service in the controlled by the DC domain. DCs are

expected in this case to release all network resources allocated for the transport service.

The multi-domain transport service modify coordination implies issuing to each of the involved DCs a configuration request to replace the transport service connections according to the newly provided paths and/or modify the connection parameters according to the newly provided configuration.

Figure 25. Multi-domain transport service is provisioned

2. Use Cases

2.1. Use Case 1. Transport service control on a single layer multi-domain transport network

Configuration (Figure 26):

- o Three-domain multi-vendor ODUk/Och transport network;
- o The domains are interconnected via ODUk inter-domain links;
- o Each of the domains is comprised of ODUk/Och network elements (switches) from a separate vendor and is controlled by a single (vendor specific) T-SDN Domain Controller (DC);
- o All DCs expose IETF TE Topology and TE Tunnel model based NBIs;
- o The transport network as a whole is controlled by a single hierarchical T-SDN controller (HC);
- o HC makes use of the NBIs to set up client-provider relationship with each of the DCs and controls via the DCs their respective network domains;
- o Three customer IP/MPLS sites are connected to the transport network via ODUk access links;
- o The customer IP/MPLS routers and the router transport ports connecting the routers to the transport network are managed autonomously and independently from the transport network.

Figure 26 Three-domain ODUk/Och transport network with ODUk access and inter-domain links

Objective: Set up/manipulate/delete a shortest delay unprotected or protected transport service to provide connectivity between customer network elements C-R2 and C-R5

1) TE Topology discovery

All DCs provide to HC respective domain ODUk layer abstract TE topologies. Let's assume that each such topology is a single-node TE topology (as described in 1.3.1, abstract TE topology of this type represents the entire domain as a single asymmetrical/blocking TE node). Let's further assume that the abstract TE nodes representing the domains are attributed with detailed connectivity matrices optimized according to the shortest delay criterion. [Note: single-node abstract TE topologies are assumed for simplicity sake. Alternatively, any DC could have provided an abstract TE topology of any type described in 1.3].

HC merges the provided TE topologies into its own native TE topology (the TE topology merging procedures are discussed in 1.4). The merged TE topology, as well as the TE topologies provided by DCs, are depicted in Figure 27. The merged TE topology homogeneously describes the entire transport network and hence is suitable for path computations across the network. Note that the dotted lines in the Figure connecting the topology access TE links with customer devices illustrate that HC in this use case has neither control nor information on the customer devices/ports and, therefore, can only provide a connectivity between the requested transport service ingress and egress access links (on assumption that the customer transport ports are provisioned independently)

Figure 27. Three-domain single layer transport network abstract TE topology

2) Transport service path computation

Using the merged TE topology (Figure 27, upper part) HC selects one or more optimal and sufficiently disjoint from each other TE path(s) for the requested transport service connection(s). Resulting TE paths for the requested end-to-end protected transport service, for example, could be as marked on the upper part of Figure 28.

It is important to keep in mind that HC's path computer is capable of performing the necessary path selection only as long as the merged TE topology provides the necessary TE visibility for the path selection, both intra-domain (e.g. by virtue of provided by the abstract TE nodes detailed connectivity matrices) and inter-domain (because of provided inter-domain TE link attributes). In case one or more DCs is/are not capable of or willing to provide the detailed connectivity matrices (that is, DCs expose the respective domains as black boxes - unconstrained TE nodes terminating the inter-domain TE links), HC will not be able to select the end-to-end TE path(s) for the requested transport service on its own. In such a case HC may opt for making use of the Path Computation NBI, exposed by the DCs to explore/evaluate intra-domain TE path availability in real time. IETF TE Tunnel model supports the Path Computation NBI by allowing for the configuration of transport services in COMPUTE_ONLY mode. In this mode the provider is expected to compute TE paths for a requested transport service connections and return the paths in the request's response without triggering the connection provisioning in the network.

Consider, for example, the case when none of the DCs has provided the detailed connectivity matrix attribute for the abstract TE nodes representing the respective domain. In such a case HC may:

1. Request the ingress domain DC (i.e. DC1) to compute intra-domain TE paths connecting the ingress access TE link (i.e. the link facing C-R2) with each of the inter-domain TE links (i.e. links connecting Domain 1 to Domain 2 and Domain 3 respectively);
2. Grow the TE paths returned by DC1 in (1) over the respective outbound inter-domain TE links;
3. Request the neighboring DC(s) (e.g. DC3) to compute all intra-domain TE paths connecting across the domain all inbound into the domain inter-domain TE links reached by the path growing process in (2) with all other (outbound) domain's inter-domain TE links;

4. Augment the TE paths produced in step (2) with the TE paths determined in step (3);
5. Repeat steps (2), (3) and (4) until the resulting TE paths reach the egress domain (i.e. Domain 2);
6. Request the egress domain DC (i.e. DC2) to grow each of the TE paths across the domain to connect them to the egress access TE link (i.e. the link facing C-R5);
7. Select one (or more) most optimal and sufficiently disjoint from each other TE path(s) from the list produced in step (6).

[Note: The transport service path selection method based on Path Computation NBIs exposed by DCs does not scale well and the more domains comprise the network and the more inter-domain links interconnect them, the worse the method works. Realistically, this approach will not work sufficiently well for the networks with more than 3 domains]

Figure 28. TE paths computed for the protected transport service

3) Transport service setup coordination

HC carries out the multi-domain transport service setup coordination as described in 1.9. In particular, HC splits the computed TE path(s) into 3 sets of TE path segments - one set per domain (as shown on the lower part of Figure 28), and issues a TE tunnel configuration request to each of the DCs to set up the requested transport service across the domain under the DC's control. The primary (and secondary) connection explicit path(s) is/are specified in the requests in terms of respective domain abstract TE topologies.

While processing the configuration request, each DC performs the transport service mapping (as described in 1.8). In particular, the DC translates the specified explicit path(s) from abstract into native TE topology terms, sets up supporting underlay TE tunnels

(e.g. OCh TE tunnels), and, then, allocates required ODUk containers on the selected links and provisions the ODUk cross-connects on the switches terminating the links.

If the setup is successfully completed in all three domains, the transport service connection(s) will be provisioned as depicted in Figure 29. If one of the DCs fails to set up its part, all successfully provisioned segments will be asked by HC to be released.

4) Transport service teardown coordination

HC issues to each of DCs a configuration request to release the transport service over the controlled domain, as well as the server layer TE tunnels supporting dynamically created links.

Figure 29. Transport service is provisioned

2.2. Use Case 2. End-to-end TE tunnel control on a single layer multi-domain transport network

Configuration (Figure 26): the same as in use case 1, except that HC in this use case controls customer devices/ports by extracting information from and pushing configuration to the customer site SDN controller(s) managing the customer devices directly.

Objective: Set up//delete an unprotected shortest delay TE tunnel interconnecting end-to-end C-R2 and C-R5

1) TE Topology discovery

As in use case 1 all DCs provide to HC domain ODUk layer abstract TE topologies. Additionally in this use the three customer site controllers expose the TE Topology and Tunnel model based NBIs to HC. Using the TE Topology NBI each customer controller provides to HC the respective customer site domain abstract TE topology. Customer site abstract TE topologies contain abstract TE nodes representing the devices which are directly connected to the transport network. Said abstract TE nodes host TE tunnel termination points, TTPs, representing the ports over which the customer devices are connected to the transport network, and terminate access TE links the TTPs are accessible from (see Figure 30).

Figure 30. Abstract TE topologies provided by all network domains and customer sites

HC merges the provided topologies into its own native TE Topology (the TE topology merging procedures are discussed in 1.4). The merged TE topology is depicted in Figure 31. It homogeneously describes end-to-end not only the entire transport network, but also the customer sites connected to the network and hence is suitable for TE tunnel end to end path computations.

Figure 31. Abstract TE topology describing transport network and connected to it customer sites

2) TE tunnel path computation

Using the merged TE topology (Figure 31) HC selects an optimal TE path for the requested TE tunnel connecting end-to-end the specified TE tunnel termination points, TTPs. The resulting TE path, for example, could be as marked on the upper part of Figure 32.

Figure 32. TE path computed for the TE tunnel

3) TE tunnel setup coordination

HC carries out the multi-domain TE tunnel setup coordination as described for use case 1, except that in this use case HC additionally initiates and controls the setup of the TE tunnel's head and tail segments on the respective customer sites. Note that the customer site controllers behave exactly as transport network domain DCs. In particular, they receive issued by HC configuration requests to set up the TE tunnel's head and tail segments respectively. While processing the requests the customer site controllers perform the necessary provisioning of the TE tunnel's source and destination termination points, as well as of the local sides of the selected

access links. If all segments are successfully provisioned on customer sites and network domains, the TE tunnel connection will be provisioned as marked in Figure 33.

4) TE tunnel teardown coordination

HC issues to each of DCs and customer site controllers a configuration request to release respective segments of the TE tunnel, as well as the server layer TE tunnels supporting dynamically created links.

Figure 33. TE tunnel is provisioned

2.3. Use Case 3. Transport service control on a ODUk/Och multi-domain transport network with Ethernet access links

Configuration (Figure 34): the same as in use case 1, except that all access links in this use case are Ethernet layer links (depicted as

blue lines in the Figure), while all inter-domain links remain to be ODUk layer links.

Figure 34. Three-domain ODUk/Och transport network with Ethernet layer access links

Objective: Set up/delete an unprotected shortest delay transport service supporting connectivity between C-R2 and C-R5

1) TE Topology discovery

In order to make possible for the necessary in this use case multi-layer path computation, each DC exposes to HC two (ODUk layer and Ethernet layer) abstract TE topologies, Additionally, the lower layer (ODUk) TE nodes announce hosted by them TE tunnel termination points, TTPs, capable of adopting the payload carried over the Ethernet layer access links, From the TE Topology model point of view this means that said TTPs are attributed with TE inter-layer locks

matching ones attributed to Ethernet TE links (i.e. TE links provided within Ethernet layer abstract TE topologies).

Ethernet and ODUk layer single node abstract TE topologies catered to HC by each of the DCs are presented in Figure 35.

HC merges the provided TE topologies into its own native TE Topology (the merging procedures are described in 1.4). Importantly in this case HC locks the provided TE topologies not only horizontally, but vertically as well, thus producing a two-layer TE topology homogenously describing both layers of the entire transport network, as well as the client-server layer adaptation relationships between the two layers. This makes the merged TE topology suitable for multi-layer/inter-layer multi-domain transport service path computations. The merged TE topology is presented in Figure 36.

Figure 35. ODUk and Ethernet layer abstract TE topologies exposed by DCs

Figure 36. Two-layer three-domain transport network abstract TE topology

2) Transport service path computation

Using the merged TE topology (Figure 36) HC selects an optimal TE path for the requested transport service.

Note that if HC's path computer considered only Ethernet layer TE nodes and links, the path computation would fail. This is because the Ethernet layer TE nodes (i.e. D1-e, D2-e and D3-e in the Figure) are disconnected from each other. However, the inter-layer associations (in the form of the TE inter-layer links) make possible for the path computer to select TE path(s) in the lower (ODUk) layer that can be used to set up hierarchy TE tunnel(s) supporting additional dynamic TE link(s) in the upper (Ethernet) layer in order for the requested transport service path computation to succeed.

Let's assume that the resulting TE path is as marked in Figure 37. The red line in the Figure marks the TE path selected for the ODUk layer hierarchy TE tunnel supporting the required Ethernet layer dynamic TE link.

Figure 37. Multi-layer TE path computed for the transport service

3) Transport service setup coordination

HC sets up the requested Ethernet layer transport service in two stages. First, it coordinates the end-to-end setup of the ODUk layer hierarchy TE tunnel between the selected TTPs. If this operation succeeds, a new Ethernet layer dynamic TE link (blue line connecting TE nodes D1-e and D2-e in Figure 38) is automatically added to the merged abstract TE topology. Importantly, as a part of the hierarchy transport service setup both DC1 and DC 2 add a new open-ended Ethernet layer inter-domain dynamic TE link to their respective abstract TE topologies. Second, HC coordinates the setup of the requested (Ethernet layer) transport service. The required TE path for the second stage is marked as fat blue line in the Figure. Note that DC3 controlling domain 3 is only involved in the first stage, but is oblivious to the second stage.

Figure 38. A new Ethernet layer TE link supported by ODUk layer TE tunnel is added to the provided and merged abstract TE topologies

IF all involved DCs confirm successful setup completion, the requested transport service, as well as the supporting server layer hierarchy TE tunnel, will be provisioned as depicted in Figure 39. If one of the DCs fails to set up its segment in either of the layers, all successfully provisioned segments will be requested by HC to be released.

Figure 39. Ethernet transport service and supporting ODUk TE tunnel are provisioned

4) Transport service teardown coordination

First, HC issues to DC1 and DC2 a configuration request to release the Ethernet layer transport service in the respective domains. After that, all three DCs are requested to release the segments of the supporting ODUk layer hierarchy TE tunnel. While processing the request DC1 and DC2 also remove the dynamic Ethernet layer TE links supported by the respective hierarchy TE tunnel's segments, thus the

network's abstract TE topologies are reverted back to the state as shown in Figures 35 and 36.

2.4. Use Case 4. Transport service control on a ODUk/Och multi-domain transport network with multi-function access links

Configuration (Figure 40): the same as in use case 3, except that all access links in this use case are multi-function links (depicted in the Figure as blue compound lines). Let's assume that, depending on configuration, the multi-function access links in this use case can carry either Ethernet or SDH/STM16 layer payload.

Objective: Set up/delete an unprotected shortest delay SDH/STM16 layer transport service interconnecting C-R2 and C-R5

Figure 40. Three-domain ODUk/Och transport network with multi-function access links

1) TE Topology discovery

The TE Topology model considers multi-function links as parallel mutually exclusive TE links each belonging to a separate layer network. For this use case each DC exposes to HC three (ODUk-, Ethernet- and SDH/STM16-layer) abstract TE topologies (generally speaking, one abstract TE topology per each layer network supported by at least one access or inter-domain link). Like in use case 3, the lower layer (ODUk) TE nodes announce hosted by them TE tunnel termination points, TTPs, capable in this case of adopting Ethernet, SDH/STM16 or both layer payloads. The TTPs are attributed with TE inter-layer links matching ones specified for Ethernet and/or SDH/STM16 TE links.

Ethernet, SDH/STM16 and ODUk layer single-node abstract TE topologies catered to HC by each of the DCs are presented in Figure 41.

HC merges the provided topologies into its own native TE Topology (the merging procedures are described in 1.4). As in use case 3 HC locks the provided TE topologies not only horizontally (i.e. between domains), but vertically (between layers) as well, thus producing a three-layer TE topology homogenously describing the three layers of the entire transport network, as well as the client-server layer adaptation relationships between the layers. This makes the merged TE topology suitable for multi-layer/inter-layer multi-domain transport service path computations. The merged TE topology is presented in Figure 42.

Figure 41. ODUk, Ethernet and SDH/STM16 layer abstract TE topologies exposed by DCs

Figure 42. Three-layer three-domain transport network abstract TE topology

2) Transport service path computation

Using the merged TE topology (Figure 42) HC's path computer selects a TE path for the requested transport service. For example, for the SDH/STM16 layer unprotected transport service the resulting TE path could be determined as marked in Figure 43.

Figure 43. Multi-layer TE path computed for SDH/STM16 layer transport service

3) Transport service setup coordination

Same as in use case 3.

4) Transport service teardown coordination

Same as in use case 3.

2.5. Use Case 5. Real time updates of IP/MPLS layer TE link attributes that depend on supporting transport connectivity (e.g. transport SRLGs, propagation delay, etc.)

Configuration (Figure 26): the same as in use case 1,

Objective: A transport service interconnecting transport ports of two IP routers across a transport network is likely to serve a link in IP/MPLS layer network, which is usually controlled by a client of the

transport network, such as IP/MPLS Controller. Performance of TE applications (e.g. path computer) running on the IP/MPLS Controller depends on the accuracy of IP/MPLS layer TE link attributes. Some of these attributes can change over time and are known real-time only to a transport network controller, such as HC. Examples of said attributes are transport SRLGs, propagation delay metric, protection capacities and status, etc. The objective of this use case is to ensure up-to-date state of said attributes in the IP/MPLS Controller's internal TED via necessary updates provided in a timely manner by the controller (e.g. HC) managing transport connectivity supporting IP/MPLS layer links.

Realization:

- o HC exposes and supports IETF TE Topology and TE Tunnel model based NBIs (the same NBIs that are exposed by DCs serving HC);
- o IP/MPLS Controller makes use of the exposed NBIs to set up the respective client-provider relationships with HC;
- o IP/MPLS Controller uses the TE Tunnel NBI to configure with HC a transport service interconnecting transport ports of a pair of IP routers desired to be adjacent in the IP/MPLS layer network. The TE Tunnel model allows for specifying in the transport service configuration request the TE topology and link IDs of the IP/MPLS TE link the requested transport service will be serving;
- o IP/MPLS Controller uses the TE Topology NBI to subscribe with HC on the IP/MPLS TE link notifications with respect to changes in the TE link's attributes, such as SRLGs, propagation delay, protection capabilities/status, etc.;
- o HC uses the TE Topology NBI to convey the requested notifications when HC learns the attributes IP/MPLS has expressed interest in or detects any changes since previous notifications (for example, due to network failure restoration/reversion procedures happened to the transport connectivity that supports the failure affected IP/MPLS links)

2.6. Use Case 6. Virtual Network Service

Configuration (Figure 26): the same as in use case 1,

Objective: Set up two Virtual Networks for the client, with Virtual Network 1 interconnecting customer IP routers C-R1, C-R7 and C-R4 over a single-node abstract TE topology, and Virtual Network 2

interconnecting customer IP routers C-R2, C-R3, C-R8, C-R5 and C-R6 over a full mesh link abstract TE topology as depicted in Figure 44.

[Note: A client of a transport network may want to limit the transport network connectivity of a particular type and quality within distinct subsets of its network elements interconnected across the transport network. Furthermore, a given transport network may serve more than one client. In this case some or all clients may want to ensure the availability of transport network resources in case dynamic (re-)connecting of their network elements across the transport network is envisioned. In all such cases a client may want to set up one or more Virtual Networks over provided transport network]

1) Virtual Network setup

From the client's point of view a Virtual Network setup includes the following procedures:

- o Identifying the Virtual Network membership - a subset of the client's network elements/ports to be interconnected over the abstract TE topology configured for the Virtual Network. Note that from the transport network provider's point of view this effectively determines the list of abstract TE topology's open-ended access TE links;
- o Deciding on the Virtual Network's abstract TE topology type (e.g. single-node vs. link mesh), optimization criterion (e.g. shortest delay vs. smallest cost), bandwidth, link disjointedness, adaptation capabilities and other requirements/constraints, as well as, whether the TE tunnels supporting the abstract TE topology need to be pre-established or established on demand (i.e. when respective abstract TE topology elements are selected for a client transport service);
- o Using the IETF TE Topology model based NBI exposed by the transport network controller (i.e. HC), configure the Virtual Network's abstract TE topology. Let's assume that in this use case the abstract TE topology for Virtual Network 1 is configured as a single-node abstract TE topology (see section 1.3.1) with the abstract TE node's detailed connectivity matrix optimized according to the shortest delay criteria. Likewise, the abstract TE topology for Virtual Network 2 is configured as a full-mesh link abstract TE topology (see section 1.3.2) optimized according to the smallest cost criteria with each of the abstract TE links to be supported by pre-established end-to-end protected TE tunnels.

[Note: Virtual Network's abstract TE topology (re-)configuration/negotiation process is no different from one that happens, for example, between HC and its providers, DCs, and is described in section 1.5]

Figure 44. Virtual Networks provided for a transport network client

2) Using Virtual Network

Recall that use case 1 was about setting up a transport service interconnecting customer network elements C-R2 and C-R5 across the transport network. With the Virtual Network 2 in place, the client could have used the Virtual Network's TE topology to select a TE path for the service. The TE Tunnel model based NBI allows for the client to specify the Virtual Network's TE topology ID, as well, as the selected TE path (for example, as marked in Figure 45) as a configured path attribute in the transport service configuration request to ensure that the intended transport network resources are used for the service.

Figure 45. Transport service TE path is selected on Virtual Network's TE topology

3. Security Considerations

This document does not define networking protocols and data, hence are not directly responsible for security risks.

4. IANA Considerations

This document has no actions for IANA.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[I-D.ietf-teas-yang-te-topo]

Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo-15 (work in progress), February 2018.

[I-D.ietf-teas-yang-te]

Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-13 (work in progress), March 2018.

5.2. Informative References

[RFC2702] Awduche, D., "Requirements for Traffic Engineering Over MPLS", RFC 2702, September 1999.

6. Acknowledgments

TBD.

Appendix A.

Data Examples

This section contains examples of an instance data in the JSON encoding [RFC7951].

A.1. Use Case 1

In the use case described in Section 2.1. , there are three provider network domains, each of them is represented as an abstract TE topology. The JSON encoded example data configurations for the three domains are:

A.1.1. Domain 1

```
{
  "networks": {
    "network": [
      {
        "network-types": {
          "te-topology": {}
        },
        "network-id": "otn-domain1-abs",
        "provider-id": 201,
        "client-id": 300,
        "te-topology-id": "te-topology:otn-domain1-abs",
        "node": [
          {
            "node-id": "D1",
            "te-node-id": "2.0.1.1",
            "te": {
              "te-node-attributes": {
                "domain-id" : 1,
                "is-abstract": [null],
                "underlay-topology": "domain1-och",
                "connectivity-matrices": {
                  "is-allowed": true,
                  "path-constraints": {
                    "bandwidth-generic": {
                      "te-bandwidth": {
                        "otn": [
                          {
                            "rate-type": "odul",
                            "counter": 2
                          }
                        ]
                      }
                    }
                  }
                }
              }
            }
          ]
        }
      ]
    }
  }
}
```

```
    }  
  ]  
}  
}  
}  
"connectivity-matrix": [  
  {  
    "id": 10302,  
    "from": "1-0-3",  
    "to": "1-2-0"  
  },  
  {  
    "id": 10203,  
    "from": "1-0-2",  
    "to": "1-3-0"  
  },  
  {  
    "id": 10311,  
    "from": "1-0-3",  
    "to": "1-11-0"  
  },  
  {  
    "id": 11103,  
    "from": "1-0-11",  
    "to": "1-3-0"  
  },  
  {  
    "id": 10903,  
    "from": "1-0-9",  
    "to": "1-3-0"  
  },  
  {  
    "id": 10309,  
    "from": "1-0-3",  
    "to": "1-9-0"  
  },  
  {  
    "id": 10910,  
    "from": "1-0-9",  
    "to": "1-10-0"  
  },  
],
```

```
    {
      "id": 11009,
      "from": "1-0-10",
      "to": "1-9-0"
    },
    {
      "id": 20910,
      "from": "1-1-9",
      "to": "1-10-0"
    },
    {
      "id": 21009,
      "from": "1-0-10",
      "to": "1-9-1"
    },
    {
      "id": 20911,
      "from": "1-1-9",
      "to": "1-11-0"
    },
    {
      "id": 21109,
      "from": "1-0-11",
      "to": "1-9-1"
    }
  ]
}
},
"termination-point": [
  {
    "tp-id": "1-0-3",
    "te-tp-id": 10003
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
]
```

```
    },
    {
      "tp-id": "1-3-0",
      "te-tp-id": 10300
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    }
  ],
  {
    "tp-id": "1-0-9",
    "te-tp-id": 10009
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
],
{
  "tp-id": "1-9-0",
  "te-tp-id": 10900
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
}
],
{
  "tp-id": "1-1-9",
  "te-tp-id": 10109
  "te": {
```

```
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  },
  {
    "tp-id": "1-9-1",
    "te-tp-id": 10901
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-0-2",
    "te-tp-id": 10002
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-2-0",
    "te-tp-id": 10200
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
}
```

```
    ]
  },
  {
    "tp-id": "1-0-10",
    "te-tp-id": 10010
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-10-0",
    "te-tp-id": 11000
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-0-11",
    "te-tp-id": 10011
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-11-0",
```


A.1.1.2. Domain 2

```
{
  "networks": {
    "network": [
      {
        "network-types": {
          "te-topology": {}
        },
        "network-id": "otn-domain2-abs",
        "provider-id": 202,
        "client-id": 300,
        "te-topology-id": "te-topology:otn-domain2-abs",
        "node": [
          {
            "node-id": "D2",
            "te-node-id": "2.0.2.2",
            "te": {
              "te-node-attributes": {
                "is-abstract": [null],
                "underlay-topology": "domain2-och",
                "connectivity-matrices": {
                  "is-allowed": true,
                  "path-constraints": {
                    "bandwidth-generic": {
                      "te-bandwidth": {
                        "otn": [
                          {
                            "rate-type": "odul",
                            "counter": 2
                          }
                        ]
                      }
                    }
                  }
                }
              }
            }
          ]
        }
      }
    ]
  }
}
```

```
{
  "id": 12521,
  "from": "1-0-25",
  "to": "1-21-0"
},
{
  "id": 12128,
  "from": "1-0-21",
  "to": "1-28-0"
},
{
  "id": 12821,
  "from": "1-0-28",
  "to": "1-21-0"
},
{
  "id": 12231,
  "from": "1-0-22",
  "to": "1-31-0"
},
{
  "id": 13122,
  "from": "1-0-31",
  "to": "1-22-0"
},
{
  "id": 22228,
  "from": "1-1-22",
  "to": "1-28-0"
},
{
  "id": 22822,
  "from": "1-0-28",
  "to": "1-22-1"
},
{
  "id": 12528,
  "from": "1-0-25",
  "to": "1-28-0"
},
{
```

```
        "id": 12825,
        "from": "1-0-28",
        "to": "1-25-0"
      }
    ]
  }
},
"termination-point": [
  {
    "tp-id": "1-0-21",
    "te-tp-id": 10021
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-21-0",
    "te-tp-id": 12100
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-0-22",
    "te-tp-id": 10022
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
]
```

```

    }
  ]
}
},
{
  "tp-id": "1-22-0",
  "te-tp-id": 12200
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "1-1-22",
  "te-tp-id": 10122
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "1-22-1",
  "te-tp-id": 12201
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
},
{

```

```
"tp-id": "1-0-25",
"te-tp-id": 10025
"te": {
  "interface-switching-capability": [
    {
      "switching-capability": "switching-otn",
      "encoding": "lsp-encoding-oduk"
    }
  ]
}
},
{
  "tp-id": "1-25-0",
  "te-tp-id": 12500
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "1-1-25",
  "te-tp-id": 10125
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "1-25-1",
  "te-tp-id": 12501
  "te": {
    "interface-switching-capability": [
      {
```

```
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  },
  {
    "tp-id": "1-0-28",
    "te-tp-id": 10028
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-28-0",
    "te-tp-id": 12800
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-0-31",
    "te-tp-id": 10031
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
}
```

```

    },
    {
      "tp-id": "1-31-0",
      "te-tp-id": 13100
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    }
  ]
}

```

A.1.3. Domain 3

```
{
  "networks": {
    "network": [
      {
        "network-types": {
          "te-topology": {}
        },
        "network-id": "otn-domain3-abs",
        "provider-id": 203,
        "client-id": 300,
        "te-topology-id": "te-topology:otn-domain3-abs",
        "node": [
          {
            "node-id": "D3",
            "te-node-id": "2.0.3.3",
            "te": {
              "te-node-attributes": {
                "is-abstract": [null],

```

```
"underlay-topology": "domain3-och",
"connectivity-matrices": {
  "is-allowed": true,
  "path-constraints": {
    "bandwidth-generic": {
      "te-bandwidth": {
        "otn": [
          {
            "rate-type": "odul",
            "counter": 2
          }
        ]
      }
    }
  }
}
"connectivity-matrix": [
  {
    "id": 13638,
    "from": "1-0-38",
    "to": "1-38-0"
  },
  {
    "id": 13836,
    "from": "1-0-38",
    "to": "1-36-0"
  },
  {
    "id": 13639,
    "from": "1-0-36",
    "to": "1-39-0"
  },
  {
    "id": 13936,
    "from": "1-0-39",
    "to": "1-36-0"
  },
  {
    "id": 23636,
    "from": "1-0-36",
    "to": "1-36-1"
  },
]
```



```

        {
            "id": 33636,
            "from": "1-1-36",
            "to": "1-36-0"
        },
        {
            "id": 13739,
            "from": "1-0-37",
            "to": "1-39-0"
        },
        {
            "id": 13937,
            "from": "1-0-39",
            "to": "1-37-0"
        },
        {
            "id": 23737,
            "from": "1-0-37",
            "to": "1-37-1"
        },
        {
            "id": 33737,
            "from": "1-1-37",
            "to": "1-37-0"
        }
    ]
}
},
"termination-point": [
    {
        "tp-id": "1-0-36",
        "te-tp-id": 10036
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-otn",
                    "encoding": "lsp-encoding-oduk"
                }
            ]
        }
    }
]
}

```

```
    },
    {
      "tp-id": "1-36-0",
      "te-tp-id": 13600
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    }
  ],
  {
    "tp-id": "1-0-37",
    "te-tp-id": 10037
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
],
{
  "tp-id": "1-37-0",
  "te-tp-id": 13700
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
}
],
{
  "tp-id": "1-1-37",
  "te-tp-id": 10137
  "te": {
```

```
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      },
    },
    {
      "tp-id": "1-37-1",
      "te-tp-id": 13701
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    },
    {
      "tp-id": "1-0-39",
      "te-tp-id": 10039
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    },
    {
      "tp-id": "1-39-0",
      "te-tp-id": 13900
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    }
  ]
}
```

```
    ]
  },
  {
    "tp-id": "1-0-36",
    "te-tp-id": 10036
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-36-0",
    "te-tp-id": 13600
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-0-38",
    "te-tp-id": 10038
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-38-0",
```

```
"te-tp-id": 13800
"te": {
  "interface-switching-capability": [
    {
      "switching-capability": "switching-otn",
      "encoding": "lsp-encoding-oduk"
    }
  ]
}
```

Authors' Addresses

Igor Bryskin
Huawei Technologies
Email: Igor.Bryskin@huawei.com

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

Tarek Saad
Cisco Systems Inc
Email: tsaad@cisco.com

Xufeng Liu
Jabil
Email: Xufeng_Liu@jabil.com

TEAS Working Group
Internet Draft
Intended status: Standard Track
Expires: September 2018

Italo Busi (Ed.)
Huawei
Sergio Belotti (Ed.)
Nokia
Victor Lopez
Oscar Gonzalez de Dios
Telefonica
Anurag Sharma
Google
Yan Shi
China Unicom
Ricard Vilalta
CTTC
Karthik Sethuraman
NEC

March 5, 2018

Yang model for requesting Path Computation
draft-ietf-teas-yang-path-computation-01.txt

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on September 5, 2016.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

There are scenarios, typically in a hierarchical SDN context, in which an orchestrator may not have detailed information to be able to perform an end-to-end path computation and would need to request lower layer/domain controllers to calculate some (partial) feasible paths.

Multiple protocol solutions can be used for communication between different controller hierarchical levels. This document assumes that the controllers are communicating using YANG-based protocols (e.g., NETCONF or RESTCONF).

Based on this assumption this document proposes a YANG model for a path computation request that an higher controller can exploit to retrieve the needed information, complementing his topology knowledge, to make his E2E path computation feasible.

The draft proposes a stateless RPC which complements the stateful solution defined in [TE-TUNNEL].

Moreover this document describes some use cases where a path computation request, via YANG-based protocols (e.g., NETCONF or RESTCONF), can be needed.

Table of Contents

1. Introduction.....	3
1.1. Terminology.....	5
2. Use Cases.....	5
2.1. Packet/Optical Integration.....	5
2.2. Multi-domain TE Networks.....	8
2.3. Data center interconnections.....	10
3. Motivations.....	12
3.1. Motivation for a YANG Model.....	12
3.1.1. Benefits of common data models.....	12
3.1.2. Benefits of a single interface.....	12
3.1.3. Extensibility.....	13
3.2. Interactions with TE Topology.....	14
3.2.1. TE Topology Aggregation.....	14
3.2.2. TE Topology Abstraction.....	18
3.2.3. Complementary use of TE topology and path computation.....	19
3.3. Stateless and Stateful Path Computation.....	21
4. Path Computation and Optimization for multiple paths.....	22
5. YANG Model for requesting Path Computation.....	23
5.1. Synchronization of multiple path computation requests.....	24
5.2. Returned metric values.....	25
6. YANG model for stateless TE path computation.....	27
6.1. YANG Tree.....	27
6.2. YANG Module.....	35
7. Security Considerations.....	44
8. IANA Considerations.....	45
9. References.....	45
9.1. Normative References.....	45
9.2. Informative References.....	46
10. Acknowledgments.....	46
Appendix A. Examples of dimensioning the "detailed connectivity matrix".....	47

1. Introduction

There are scenarios, typically in a hierarchical SDN context, in which an orchestrator may not have detailed information to be able to perform an end-to-end path computation and would need to request lower layer/domain controllers to calculate some (partial) feasible paths.

When we are thinking to this type of scenarios we have in mind specific level of interfaces on which this request can be applied.

We can reference ABNO Control Interface [RFC7491] in which an Application Service Coordinator can request ABNO controller to take in charge path calculation (see Figure 1 in the RFC) and/or ACTN [ACTN-frame], where controller hierarchy is defined, the need for path computation arises on both interfaces CMI (interface between Customer Network Controller(CNC) and Multi Domain Service Coordinator (MDSC)) and/or MPI (interface between MSDC-PNC). [ACTN-Info] describes an information model for the Path Computation request.

Multiple protocol solutions can be used for communication between different controller hierarchical levels. This document assumes that the controllers are communicating using YANG-based protocols (e.g., NETCONF or RESTCONF).

Path Computation Elements, Controllers and Orchestrators perform their operations based on Traffic Engineering Databases (TED). Such TEDs can be described, in a technology agnostic way, with the YANG Data Model for TE Topologies [TE-TOPO]. Furthermore, the technology specific details of the TED are modeled in the augmented TE topology models (e.g. [OTN-TOPO] for OTN ODU technologies).

The availability of such topology models allows providing the TED using YANG-based protocols (e.g., NETCONF or RESTCONF). Furthermore, it enables a PCE/Controller performing the necessary abstractions or modifications and offering this customized topology to another PCE/Controller or high level orchestrator.

Note: This document does not assume that an orchestrator/coordinator always implements a "PCE" functionality, as defined in [RFC4655].

The tunnels that can be provided over the networks described with the topology models can be also set-up, deleted and modified via YANG-based protocols (e.g., NETCONF or RESTCONF) using the TE-Tunnel Yang model [TE-TUNNEL].

This document proposes a YANG model for a path computation request defined as a stateless RPC, which complements the stateful solution defined in [TE-TUNNEL].

Moreover, this document describes some use cases where a path computation request, via YANG-based protocols (e.g., NETCONF or RESTCONF), can be needed.

1.1. Terminology

TED: The traffic engineering database is a collection of all TE information about all TE nodes and TE links in a given network.

PCE: A Path Computation Element (PCE) is an entity that is capable of computing a network path or route based on a network graph, and of applying computational constraints during the computation. The PCE entity is an application that can be located within a network node or component, on an out-of-network server, etc. For example, a PCE would be able to compute the path of a TE LSP by operating on the TED and considering bandwidth and other constraints applicable to the TE LSP service request. [RFC4655]

2. Use Cases

This section presents different use cases, where an orchestrator needs to request underlying SDN controllers for path computation.

The presented uses cases have been grouped, depending on the different underlying topologies: a) IP-Optical integration; b) Multi-domain Traffic Engineered (TE) Networks; and c) Data center interconnections.

2.1. Packet/Optical Integration

In this use case, an Optical network is used to provide connectivity to some nodes of a Packet network (see Figure 1).

A possible example could be the case where an Optical network provides connectivity to same IP routers of an IP network.

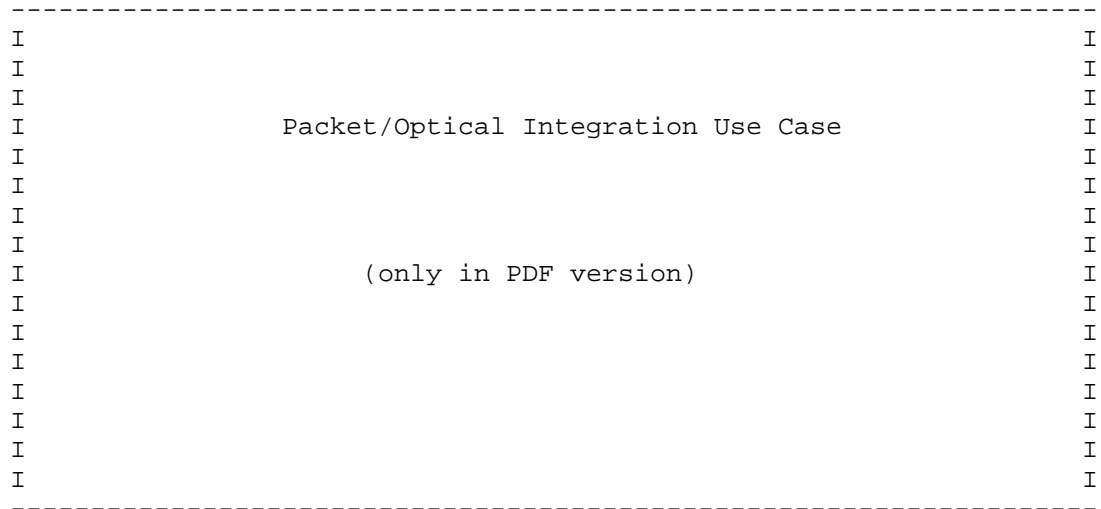


Figure 1 - Packet/Optical Integration Use Case

Figure 1 as well as Figure 2 below only show a partial view of the packet network connectivity, before additional packet connectivity is provided by the Optical network.

It is assumed that the Optical network controller provides to the packet/optical coordinator an abstracted view of the Optical network. A possible abstraction shall be representing the optical network as one "virtual node" with "virtual ports" connected to the access links.

It is also assumed that Packet network controller can provide the packet/optical coordinator the information it needs to setup connectivity between packet nodes through the Optical network (e.g., the access links).

The path computation request helps the coordinator to know the real connections that can be provided by the optical network.

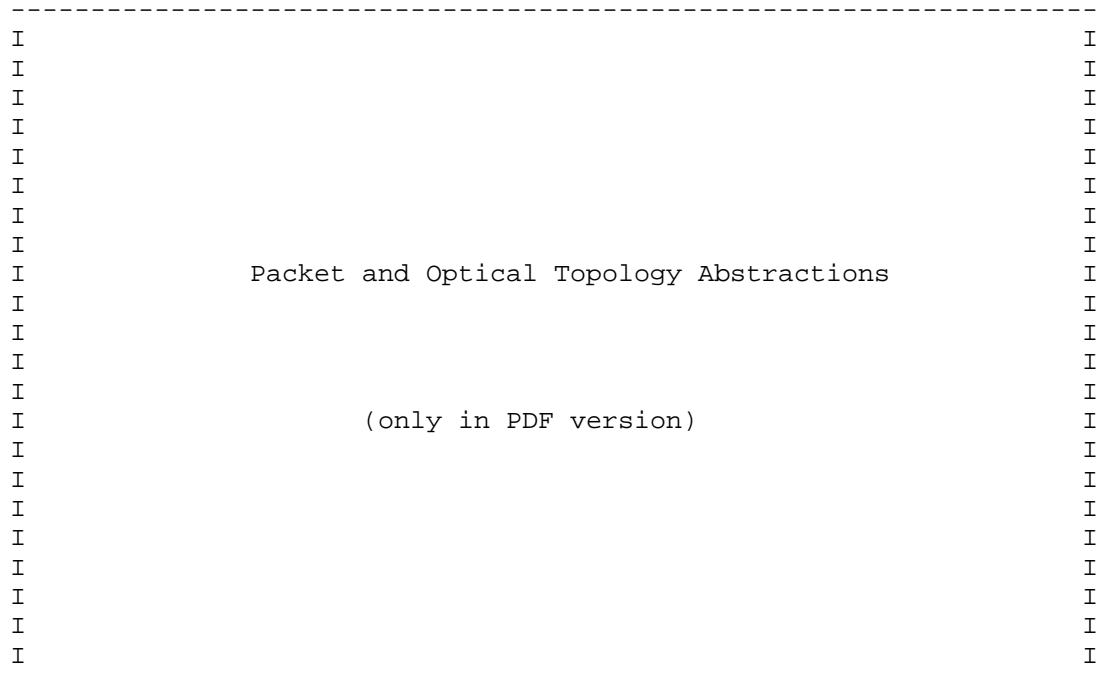


Figure 2 - Packet and Optical Topology Abstractions

In this use case, the coordinator needs to setup an optimal underlying path for an IP link between R1 and R2.

As depicted in Figure 2, the coordinator has only an "abstracted view" of the physical network, and it does not know the feasibility or the cost of the possible optical paths (e.g., VP1-VP4 and VP2-VP5), which depend from the current status of the physical resources within the optical network and on vendor-specific optical attributes.

The coordinator can request the underlying Optical domain controller to compute a set of potential optimal paths, taking into account optical constraints. Then, based on its own constraints, policy and knowledge (e.g. cost of the access links), it can choose which one of these potential paths to use to setup the optimal e2e path crossing optical network.

```
-----  
I                                     I  
I           Packet/Optical Path Computation Example           I  
I                                     I  
I                                     I  
I                                     I  
I                                     I  
I                                     I  
I                                     I  
I           (only in PDF version)           I  
I                                     I  
-----
```

Figure 3 - Packet/Optical Path Computation Example

For example, in Figure 3, the Coordinator can request the Optical network controller to compute the paths between VP1-VP4 and VP2-VP5 and then decide to setup the optimal end-to-end path using the VP2-VP5 Optical path even this is not the optimal path from the Optical domain perspective.

Considering the dynamicity of the connectivity constraints of an Optical domain, it is possible that a path computed by the Optical network controller when requested by the Coordinator is no longer valid/available when the Coordinator requests it to be setup up.

It is worth noting that with the approach proposed in this document, the likelihood for this issue to happen can be quite small since the time window between the path computation request and the path setup request should be quite short (especially if compared with the time that would be needed to update the information of a very detailed abstract connectivity matrix).

If this risk is still not acceptable, the Orchestrator may also optionally request the Optical domain controller not only to compute the path but also to keep track of its resources (e.g., these resources can be reserved to avoid being used by any other connection). In this case, some mechanism (e.g., a timeout) needs to be defined to avoid having stranded resources within the Optical domain.

2.2. Multi-domain TE Networks

In this use case there are two TE domains which are interconnected together by multiple inter-domains links.

A possible example could be a multi-domain optical network.

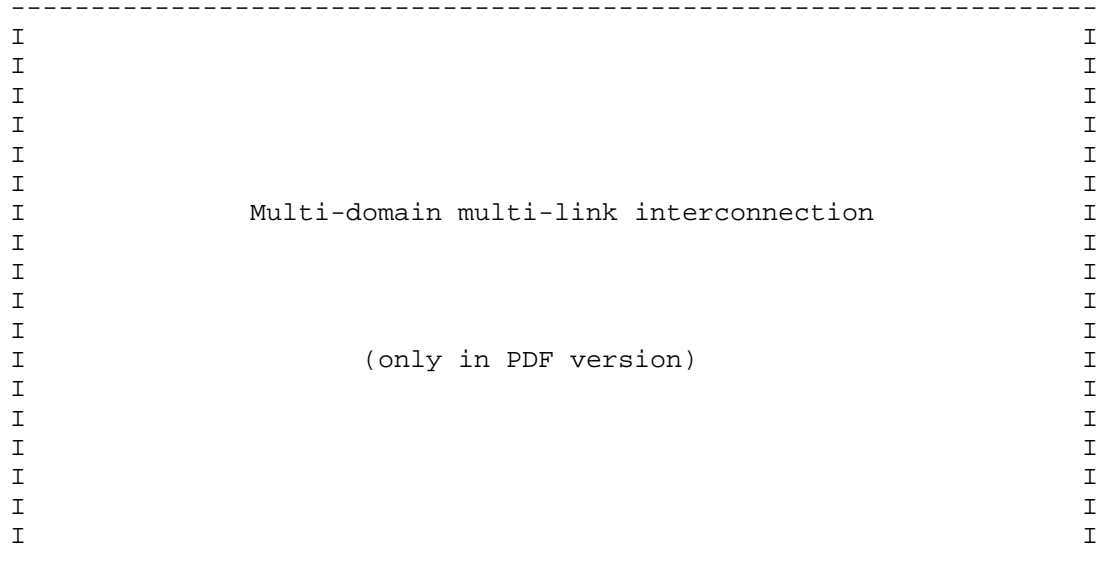


Figure 4 - Multi-domain multi-link interconnection

In order to setup an end-to-end multi-domain TE path (e.g., between nodes A and H), the orchestrator needs to know the feasibility or the cost of the possible TE paths within the two TE domains, which depend from the current status of the physical resources within each TE network. This is more challenging in case of optical networks because the optimal paths depend also on vendor-specific optical attributes (which may be different in the two domains if they are provided by different vendors).

In order to setup a multi-domain TE path (e.g., between nodes A and H), Orchestrator can request the TE domain controllers to compute a set of intra-domain optimal paths and take decisions based on the information received. For example:

- o The Orchestrator asks TE domain controllers to provide set of paths between A-C, A-D, E-H and F-H

- o TE domain controllers return a set of feasible paths with the associated costs: the path A-C is not part of this set (in optical networks, it is typical to have some paths not being feasible due to optical constraints that are known only by the optical domain controller)
- o The Orchestrator will select the path A- D-F- H since it is the only feasible multi-domain path and then request the TE domain controllers to setup the A-D and F-H intra-domain paths
- o If there are multiple feasible paths, the Orchestrator can select the optimal path knowing the cost of the intra-domain paths (provided by the TE domain controllers) and the cost of the inter-domain links (known by the Orchestrator)

This approach may have some scalability issues when the number of TE domains is quite big (e.g. 20).

In this case, it would be worthwhile using the abstract TE topology information provided by the domain controllers to limit the number of potential optimal end-to-end paths and then request path computation to fewer domain controllers in order to decide what the optimal path within this limited set is.

For more details, see section 3.2.3.

2.3. Data center interconnections

In these use case, there is a TE domain which is used to provide connectivity between data centers which are connected with the TE domain using access links.

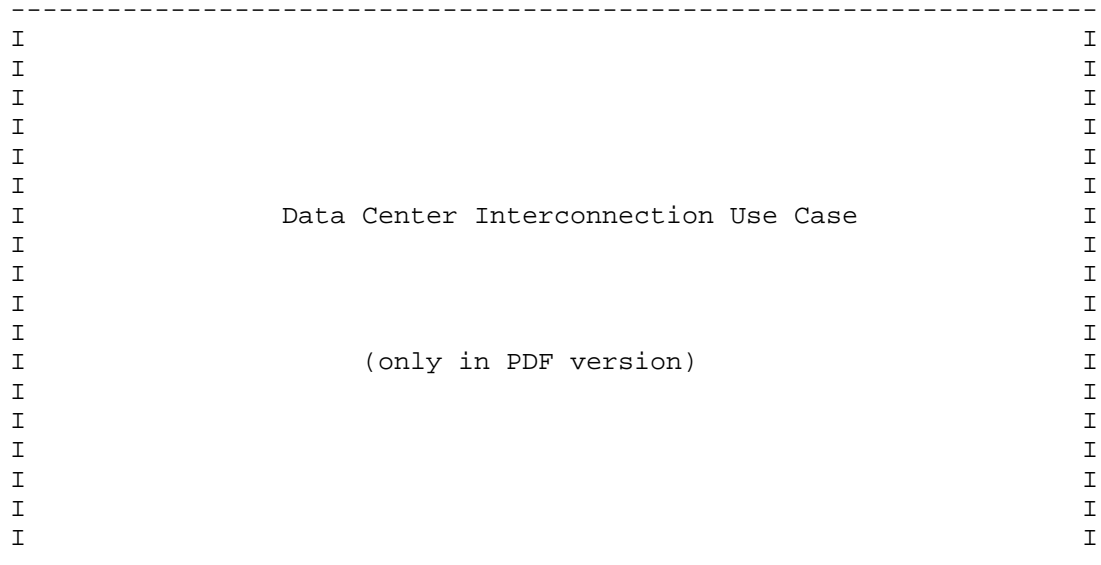


Figure 5 - Data Center Interconnection Use Case

In this use case, there is need to transfer data from Data Center 1 (DC1) to either DC2 or DC3 (e.g. workload migration).

The optimal decision depends both on the cost of the TE path (DC1-DC2 or DC1-DC3) and of the data center resources within DC2 or DC3.

The Cloud Orchestrator needs to make a decision for optimal connection based on TE Network constraints and data centers resources. It may not be able to make this decision because it has only an abstract view of the TE network (as in use case in 2.1).

The cloud orchestrator can request to the TE domain controller to compute the cost of the possible TE paths (e.g., DC1-DC2 and DC1-DC3) and to the DC controller to provide the information it needs about the required data center resources within DC2 and DC3 and then it can take the decision about the optimal solution based on this information and its policy.

3. Motivations

This section provides the motivation for the YANG model defined in this document.

Section 3.1 describes the motivation for a YANG model to request path computation.

Section 3.2 describes the motivation for a YANG model which complements the TE Topology YANG model defined in [TE-TOPO].

Section 3.3 describes the motivation for a stateless YANG RPC which complements the TE Tunnel YANG model defined in [TE-TUNNEL].

3.1. Motivation for a YANG Model

3.1.1. Benefits of common data models

Path computation requests are closely aligned with the YANG data models that provide (abstract) TE topology information, i.e., [TE-TOPO] as well as that are used to configure and manage TE Tunnels, i.e., [TE-TUNNEL]. Therefore, there is no need for an error-prone mapping or correlation of information. For instance, there is benefit in using the same endpoint identifiers in path computation requests and in the topology modeling. Also, the attributes used in path computation constraints use the same data models. As a result, there are many benefits in aligning path computation requests with YANG models for TE topology information and TE Tunnels configuration and management.

3.1.2. Benefits of a single interface

A typical use case for path computation requests is the interface between an orchestrator and a domain controller. The system integration effort is typically lower if a single, consistent interface is used between such systems, i.e., one data modeling language (i.e., YANG) and a common protocol (e.g., NETCONF or RESTCONF).

Practical benefits of using a single, consistent interface include:

1. Simple authentication and authorization: The interface between different components has to be secured. If different protocols have different security mechanisms, ensuring a common access control model may result in overhead. For instance, there may

be a need to deal with different security mechanisms, e.g., different credentials or keys. This can result in increased integration effort.

2. Consistency: Keeping data consistent over multiple different interfaces or protocols is not trivial. For instance, the sequence of actions can matter in certain use cases, or transaction semantics could be desired. While ensuring consistency within one protocol can already be challenging, it is typically cumbersome to achieve that across different protocols.
3. Testing: System integration requires comprehensive testing, including corner cases. The more different technologies are involved, the more difficult it is to run comprehensive test cases and ensure proper integration.
4. Middle-box friendliness: Provider and consumer of path computation requests may be located in different networks, and middle-boxes such as firewalls, NATs, or load balancers may be deployed. In such environments it is simpler to deploy a single protocol. Also, it may be easier to debug connectivity problems.
5. Tooling reuse: Implementers may want to implement path computation requests with tools and libraries that already exist in controllers and/or orchestrators, e.g., leveraging the rapidly growing eco-system for YANG tooling.

3.1.3. Extensibility

Path computation is only a subset of the typical functionality of a controller. In many use cases, issuing path computation requests comes along with the need to access other functionality on the same system. In addition to obtaining TE topology, for instance also configuration of services (setup/modification/deletion) may be required, as well as:

1. Receiving notifications for topology changes as well as integration with fault management
2. Performance management such as retrieving monitoring and telemetry data
3. Service assurance, e.g., by triggering OAM functionality
4. Other fulfilment and provisioning actions beyond tunnels and services, such as changing QoS configurations

YANG is a very extensible and flexible data modeling language that can be used for all these use cases.

The YANG model for path computation requests seamlessly complements with [TE-TOP0] and [TE-TUNNEL] in the use cases where YANG-based protocols (e.g., NETCONF or RESTCONF) are used.

3.2. Interactions with TE Topology

The use cases described in section 2 have been described assuming that the topology view exported by each underlying SDN controller to the orchestrator is aggregated using the "virtual node model", defined in [RFC7926].

TE Topology information, e.g., as provided by [TE-TOP0], could in theory be used by an underlying SDN controllers to provide TE information to the orchestrator thus allowing a PCE available within the Orchestrator to perform multi-domain path computation by its own, without requesting path computations to the underlying SDN controllers.

In case the Orchestrator does not implement a PCE function, as discussed in section 1, it could not perform path computation based on TE Topology information and would instead need to request path computation to the underlying controllers to get the information it needs to compute the optimal end-to-end path.

This section analyzes the need for an orchestrator to request underlying SDN controllers for path computation even in case the Orchestrator implements a PCE functionality, as well as how the TE Topology information and the path computation can be complementary.

In nutshell, there is a scalability trade-off between providing all the TE information needed by PCE, when implemented by the Orchestrator, to take optimal path computation decisions by its own versus requesting the Orchestrator to ask to too many underlying SDN Domain Controllers a set of feasible optimal intra-domain TE paths.

3.2.1. TE Topology Aggregation

Using the TE Topology model, as defined in [TE-TOP0], the underlying SDN controller can export the whole TE domain as a single abstract TE node with a "detailed connectivity matrix", which extends the "connectivity matrix", defined in [RFC7446], with specific TE attributes (e.g., delay, SRLGs and summary TE metrics).

The information provided by the "detailed abstract connectivity matrix" would be equivalent to the information that should be provided by "virtual link model" as defined in [RFC7926].

For example, in the Packet/Optical integration use case, described in section 2.1, the Optical network controller can make the information shown in Figure 3 available to the Coordinator as part of the TE Topology information and the Coordinator could use this information to calculate by its own the optimal path between R1 and R2, without requesting any additional information to the Optical network Controller.

However, there is a tradeoff between accuracy (i.e., providing "all" the information that might be needed by the PCE available to Orchestrator) and scalability, to be considered when designing the amount of information to provide within the "detailed abstract connectivity matrix".

Figure 6 below shows another example, similar to Figure 3, where there are two possible Optical paths between VP1 and VP4 with different properties (e.g., available bandwidth and cost).

```

-----
I                                     I
I           IP+Optical Path Computation Example           I
I                   with multiple choices                   I
I                                                         I
I                                                         I
I                                                         I
I                   (only in PDF version)                   I
I                                                         I
I                                                         I
-----

```

Figure 6 - Packet/Optical Path Computation Example with multiple choices

Reporting all the information, as in Figure 6, using the "detailed abstract connectivity matrix", is quite challenging from a scalability perspective. The amount of this information is not just based on number of end points (which would scale as N-square), but also on many other parameters, including client rate, user constraints / policies for the service, e.g. max latency < N ms, max cost, etc., exclusion policies to route around busy links, min OSNR

margin, max preFEC BER etc. All these constraints could be different based on connectivity requirements.

Examples of how the "detailed connectivity matrix" can be dimensioned are described in Appendix A.

It is also worth noting that the "connectivity matrix" has been originally defined in WSON, [RFC7446] to report the connectivity constraints of a physical node within the WDM network: the information it contains is pretty "static" and therefore, once taken and stored in the TE data base, it can be always being considered valid and up-to-date in path computation request.

Using the "connectivity matrix" with an abstract node to abstract the information regarding the connectivity constraints of an Optical domain, would make this information more "dynamic" since the connectivity constraints of an Optical domain can change over time because some optical paths that are feasible at a given time may become unfeasible at a later time when e.g., another optical path is established. The information in the "detailed abstract connectivity matrix" is even more dynamic since the establishment of another optical path may change some of the parameters (e.g., delay or available bandwidth) in the "detailed abstract connectivity matrix" while not changing the feasibility of the path.

"Connectivity matrix" is sometimes confused with optical reach table that contain multiple (e.g. k-shortest) regen-free reachable paths for every A-Z node combination in the network. Optical reach tables can be calculated offline, utilizing vendor optical design and planning tools, and periodically uploaded to the Controller: these optical path reach tables are fairly static. However, to get the connectivity matrix, between any two sites, either a regen free path can be used, if one is available, or multiple regen free paths are concatenated to get from src to dest, which can be a very large combination. Additionally, when the optical path within optical domain needs to be computed, it can result in different paths based on input objective, constraints, and network conditions. In summary, even though "optical reachability table" is fairly static, which regen free paths to build the connectivity matrix between any source and destination is very dynamic, and is done using very sophisticated routing algorithms.

There is therefore the need to keep the information in the "connectivity matrix" updated which means that there another tradeoff between the accuracy (i.e., providing "all" the information

that might be needed by the Orchestrator's PCE) and having up-to-date information. The more the information is provided and the longer it takes to keep it up-to-date which increases the likelihood that the Orchestrator's PCE computes paths using not updated information.

It seems therefore quite challenging to have a "detailed abstract connectivity matrix" that provides accurate, scalable and updated information to allow the Orchestrator's PCE to take optimal decisions by its own.

If the information in the "detailed abstract connectivity matrix" is not complete/accurate, we can have the following drawbacks considering for example the case in Figure 6:

- o If only the VP1-VP4 path with available bandwidth of 2 Gb/s and cost 50 is reported, the Orchestrator's PCE will fail to compute a 5 Gb/s path between routers R1 and R2, although this would be feasible;
- o If only the VP1-VP4 path with available bandwidth of 10 Gb/s and cost 60 is reported, the Orchestrator's PCE will compute, as optimal, the 1 Gb/s path between R1 and R2 going through the VP2-VP5 path within the Optical domain while the optimal path would actually be the one going through the VP1-VP4 sub-path (with cost 50) within the Optical domain.

Instead, using the approach proposed in this document, the Orchestrator, when it needs to setup an end-to-end path, it can request the Optical domain controller to compute a set of optimal paths (e.g., for VP1-VP4 and VP2-VP5) and take decisions based on the information received:

- o When setting up a 5 Gb/s path between routers R1 and R2, the Optical domain controller may report only the VP1-VP4 path as the only feasible path: the Orchestrator can successfully setup the end-to-end path passing through this Optical path;
- o When setting up a 1 Gb/s path between routers R1 and R2, the Optical domain controller (knowing that the path requires only 1 Gb/s) can report both the VP1-VP4 path, with cost 50, and the VP2-VP5 path, with cost 65. The Orchestrator can then compute the optimal path which is passing through the VP1-VP4 sub-path (with cost 50) within the Optical domain.

3.2.2. TE Topology Abstraction

Using the TE Topology model, as defined in [TE-TOPO], the underlying SDN controller can export an abstract TE Topology, composed by a set of TE nodes and TE links, which are abstracting the topology controlled by each domain controller.

Considering the example in Figure 4, the TE domain controller 1 can export a TE Topology encompassing the TE nodes A, B, C and D and the TE Link interconnecting them. In a similar way, TE domain controller 2 can export a TE Topology encompassing the TE nodes E, F, G and H and the TE Link interconnecting them.

In this example, for simplicity reasons, each abstract TE node maps with each physical node, but this is not necessary.

In order to setup a multi-domain TE path (e.g., between nodes A and H), the Orchestrator can compute by its own an optimal end-to-end path based on the abstract TE topology information provided by the domain controllers. For example:

- o Orchestrator's PCE, based on its own information, can compute the optimal multi-domain path being A-B-C-E-G-H, and then request the TE domain controllers to setup the A-B-C and E-G-H intra-domain paths
- o But, during path setup, the domain controller may find out that A-B-C intra-domain path is not feasible (as discussed in section 2.2, in optical networks it is typical to have some paths not being feasible due to optical constraints that are known only by the optical domain controller), while only the path A-B-D is feasible
- o So what the hierarchical controller computed is not good and need to re-start the path computation from scratch

As discussed in section 3.2.1, providing more extensive abstract information from the TE domain controllers to the multi-domain Orchestrator may lead to scalability problems.

In a sense this is similar to the problem of routing and wavelength assignment within an Optical domain. It is possible to do first routing (step 1) and then wavelength assignment (step 2), but the chances of ending up with a good path is low. Alternatively, it is possible to do combined routing and wavelength assignment, which is

known to be a more optimal and effective way for Optical path setup. Similarly, it is possible to first compute an abstract end-to-end path within the multi-domain Orchestrator (step 1) and then compute an intra-domain path within each Optical domain (step 2), but there are more chances not to find a path or to get a suboptimal path that performing per-domain path computation and then stitch them.

3.2.3. Complementary use of TE topology and path computation

As discussed in section 2.2, there are some scalability issues with path computation requests in a multi-domain TE network with many TE domains, in terms of the number of requests to send to the TE domain controllers. It would therefore be worthwhile using the TE topology information provided by the domain controllers to limit the number of requests.

An example can be described considering the multi-domain abstract topology shown in Figure 7. In this example, an end-to-end TE path between domains A and F needs to be setup. The transit domain should be selected between domains B, C, D and E.

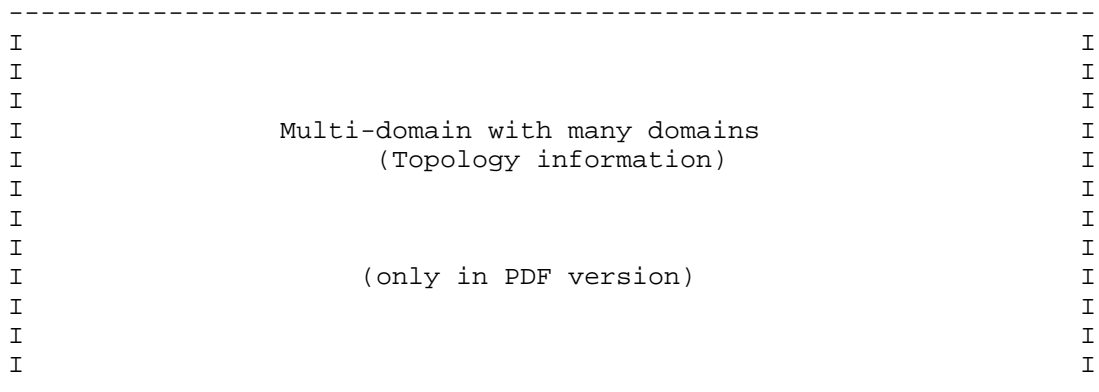


Figure 7 - Multi-domain with many domains (Topology information)

The actual cost of each intra-domain path is not known a priori from the abstract topology information. The Orchestrator only knows, from the TE topology provided by the underlying domain controllers, the feasibility of some intra-domain paths and some upper-bound and/or lower-bound cost information. With this information, together with

the cost of inter-domain links, the Orchestrator can understand by its own that:

- o Domain B cannot be selected as the path connecting domains A and E is not feasible;
- o Domain E cannot be selected as a transit domain since it is known from the abstract topology information provided by domain controllers that the cost of the multi-domain path A-E-F (which is 100, in the best case) will be always be higher than the cost of the multi-domain paths A-D-F (which is 90, in the worst case) and A-E-F (which is 80, in the worst case)

Therefore, the Orchestrator can understand by its own that the optimal multi-domain path could be either A-D-F or A-E-F but it cannot know which one of the two possible options actually provides the optimal end-to-end path.

The Orchestrator can therefore request path computation only to the TE domain controllers A, D, E and F (and not to all the possible TE domain controllers).

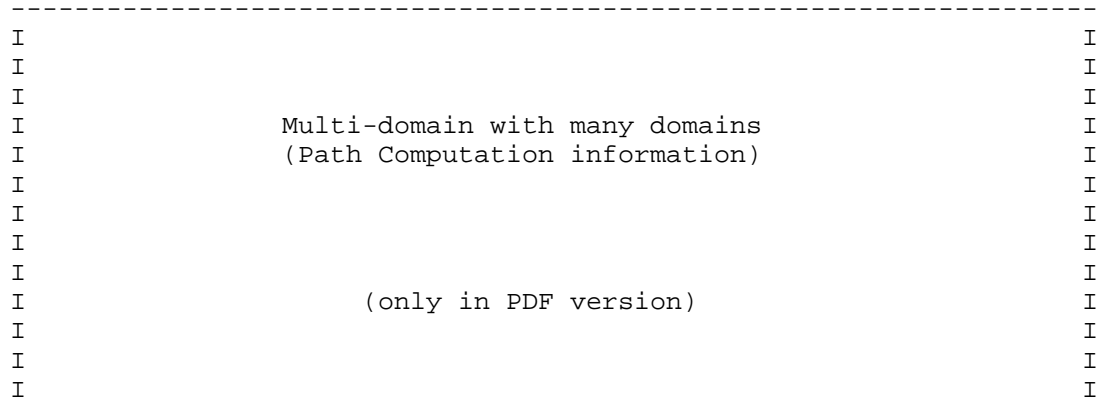


Figure 8 - Multi-domain with many domains (Path Computation information)

Based on these requests, the Orchestrator can know the actual cost of each intra-domain path which belongs to potential optimal end-to-end paths, as shown in Figure 8, and then compute the optimal

end-to-end path (e.g., A-D-F, having total cost of 50, instead of A-C-F having a total cost of 70).

3.3. Stateless and Stateful Path Computation

The TE Tunnel YANG model, defined in [TE-TUNNEL], can support the need to request path computation.

It is possible to request path computation by configuring a "compute-only" TE tunnel and retrieving the computed path(s) in the LSP(s) Record-Route Object (RRO) list as described in section 3.3.1 of [TE-TUNNEL].

This is a stateful solution since the state of each created "compute-only" TE tunnel needs to be maintained and updated, when underlying network conditions change.

It is very useful to provide options for both stateless and stateful path computation mechanisms. It is suggested to use stateless mechanisms as much as possible and to rely on stateful path computation when really needed.

Stateless RPC allows requesting path computation using a simple atomic operation and it is the natural option/choice, especially with stateless PCE.

Since the operation is stateless, there is no guarantee that the returned path would still be available when path setup is requested: this is not a major issue in case the time between path computation and path setup is short.

The RPC response must be provided synchronously and, if collaborative computations are time consuming, it may not be possible to immediate reply to client.

In this case, the client can define a maximum time it can wait for the reply, such that if the computation does not complete in time, the server will abort the path computation and reply to the client with an error. It may be possible that the server has tighter timing constraints than the client: in this case the path computation is aborted earlier than the time specified by the client.

Note - The RPC response issue (slow RPC server) is not specific to the path computation RPC case so, it may be worthwhile, evaluating

whether a more generic solution applicable to any YANG RPC can be used instead.

In case the stateless solution is not sufficient, a stateful solution, based on "compute-only" TE tunnel, could be used to support asynchronous operations and/or to get notifications in case the computed path has been changed.

It is worth noting that also the stateful solution, although increasing the likelihood that the computed path is available at path setup, it does not guaranteed that because notifications may not be reliable or delivered on time.

The stateful path computation has also the following drawbacks:

- o Several messages required for any path computation
- o Requires persistent storage in the provider controller
- o Need for garbage collection for stranded paths
- o Process burden to detect changes on the computed paths in order to provide notifications update

4. Path Computation and Optimization for multiple paths

There are use cases, where it is advantageous to request path computation for a set of paths, through a network or through a network domain, using a single request [RFC5440].

This would reduce the protocol overhead to send multiple requests.

In the context of a typical multi-domain TE network, there could be multiple choices for the ingress/egress points of a domain and the Orchestrator needs to request path computation between all the ingress/egress pairs to select the best pair. For example, in the example of section 2.2, the Orchestrator needs to request the TE network controller 1 to compute the A-C and the A-D paths and to the TE network controller 2 to compute the E-H and the F-H paths.

It is also possible that the Orchestrator receives a request to setup a group of multiple end to end connections. The orchestrator needs to request each TE domain controller to compute multiple paths, one (or more) for each end to end connection.

There are also scenarios where it can be needed to request path computation for a set of paths in a synchronized fashion.

One example could be computing multiple diverse paths. Computing a set of diverse paths in a not-synchronized fashion, leads to a high probability of not being able to satisfy all request. In this case, a sub-optimal primary path that could be protected by a diversely routed secondary path should be computed instead of an optimal primary path that could not be protected.

There are also scenarios where it is needed to request optimizing a set of paths using objective functions that apply to the whole set of paths, see [RFC5541], e.g. to minimize the sum of the costs of all the computed paths in the set.

5. YANG Model for requesting Path Computation

This document define a YANG stateless RPC to request path computation as an "augmentation" of tunnel-rpc, defined in [TE-TUNNEL]. This model provides the RPC input attributes that are needed to request path computation and the RPC output attributes that are needed to report the computed paths.

```
augment /te:tunnels-rpc/te:input/te:tunnel-info:
  +---- path-request* [request-id]
  .....

augment /te:tunnels-rpc/te:output/te:result:
  +--ro response* [response-id]
  +--ro response-id      uint32
  +--ro (response-type)?
    +--:(no-path-case)
    |   +--ro no-path!
    +--:(path-case)
      +--ro computed-path
        +--ro path-id?          yang-types:uuid
        +--ro path-properties
        .....
```

This model extensively re-uses the grouping defined in [TE-TUNNEL] to ensure maximal syntax and semantics commonality.

5.1. Synchronization of multiple path computation requests

The YANG model permits to synchronize a set of multiple path requests (identified by specific request-id) all related to a "svec" container emulating the syntax of "SVEC" PCEP object [RFC 5440].

```

+---- synchronization* [synchronization-id]
  +---- synchronization-id    uint32
  +---- svec
    | +---- relaxable?          boolean
    | +---- link-diverse?      boolean
    | +---- node-diverse?      boolean
    | +---- srlg-diverse?      boolean
    | +---- request-id-number* uint32
  +---- svec-constraints
    | +---- path-metric-bound* [metric-type]
    |   +---- metric-type      identityref
    |   +---- upper-bound?     uint64
  +---- path-srlgs
    | +---- usage?             identityref
    | +---- values*            srlg
  +---- exclude-objects
  .....
  +---- optimizations
    +---- (algorithm)?
      +--:(metric)
        | +---- optimization-metric* [metric-type]
        |   +---- metric-type      identityref
        |   +---- weight?          uint8
      +--:(objective-function)
        +---- objective-function
          +---- objective-function-type? identityref

```

The model, in addition to the metric types, defined in [TE-TUNNEL], which can be applied to each individual path request, defines additional specific metrics types that apply to a set of synchronized requests, as referenced in [RFC5541].

```

identity svec-metric-type {
  description
    "Base identity for svec metric type";
}

```

```
identity svec-metric-cumul-te {
  base svec-metric-type;
  description
    "TE cumulative path metric";
}

identity svec-metric-cumul-igp {
  base svec-metric-type;
  description
    "IGP cumulative path metric";
}

identity svec-metric-cumul-hop {
  base svec-metric-type;
  description
    "Hop cumulative path metric";
}

identity svec-metric-aggregate-bandwidth-consumption {
  base svec-metric-type;
  description
    "Cumulative bandwith consumption of the set of synchronized
paths";
}

identity svec-metric-load-of-the-most-loaded-link {
  base svec-metric-type;
  description
    "Load of the most loaded link";
}
```

5.2. Returned metric values

This YANG model provides a way to return the values of the metrics computed by the path computation in the output of RPC, together with other important information (e.g. srlg, affinities, explicit route), emulating the syntax of the "C" flag of the "METRIC" PCEP object [RFC 5440]:

```
augment /te:tunnels-rpc/te:output/te:result:
```

```

+--ro response* [response-id]
+--ro response-id      uint32
+--ro (response-type)?
+--:(no-path-case)
|   +--ro no-path!
+--:(path-case)
+--ro pathCompService
+--ro path-id?          yang-types:uuid
+--ro path-properties
+--ro path-metric* [metric-type]
|   +--ro metric-type      identityref
|   +--ro accumulative-value? uint64
+--ro path-affinities
|   +--ro constraint* [usage]
|       +--ro usage      identityref
|       +--ro value?     admin-groups
+--ro path-srlgs
|   +--ro usage?          identityref
|   +--ro values*         srlg
+--ro path-route-objects
.....

```

It also allows to request which metric should returned in the input of RPC:

```

augment /te:tunnels-rpc/te:input/te:tunnel-info:
+---- path-request* [request-id]
|   +---- request-id      uint32
|   .....
|   +---- requested-metrics* [metric-type]
|       +---- metric-type  identityref
|       .....

```

This feature is essential for using a stateless path computation in a multi-domain TE network as described in section 2.2. In this case, the metrics returned by a path computation requested to a given TE network controller must be used by the Orchestrator to compute the best end-to-end path. If they are missing the Orchestrator cannot compare different paths calculated by the TE network controllers and choose the best one for the optimal e2e path.

6. YANG model for stateless TE path computation

6.1. YANG Tree

Figure 9 below shows the tree diagram of the YANG model defined in module ietf-te-path-computation.yang.

```

module: ietf-te-path-computation
  +--rw paths
    +--ro path* [path-id]
      +--ro path-id          yang-types:uuid
      +--ro path-properties
        +--ro path-metric* [metric-type]
          | +--ro metric-type      identityref
          | +--ro accumulative-value? uint64
        +--ro path-affinities
          | +--ro constraint* [usage]
          |   +--ro usage      identityref
          |   +--ro value?    admin-groups
        +--ro path-srlgs
          | +--ro usage?      identityref
          | +--ro values*    srlg
        +--ro path-route-objects
          +--ro path-route-object* [index]
            +--ro index          uint32
            +--ro (type)?
              +--:(numbered)
                | +--ro numbered-hop
                |   +--ro address?    te-types:te-tp-id
                |   +--ro hop-type?   te-hop-type
                |   +--ro direction? te-link-direction
              +--:(as-number)
                | +--ro as-number-hop
                |   +--ro as-number?  binary
                |   +--ro hop-type?   te-hop-type
              +--:(unnumbered)
                | +--ro unnumbered-hop
                |   +--ro node-id?    te-types:te-node-id
                |   +--ro link-tp-id? te-types:te-tp-id
                |   +--ro hop-type?   te-hop-type

```

```

|         +--ro direction?      te-link-direction
+---:(label)
|         +--ro label-hop
|         +--ro te-label
|         +--ro (technology)?
|         |   +---:(generic)
|         |   +--ro generic?      rt-
types:generalized-label
|         +--ro direction?      te-label-direction
augment /te:tunnels-rpc/te:input/te:tunnel-info:
+----- path-request* [request-id]
|   +----- request-id          uint32
|   +----- source?             inet:ip-address
|   +----- destination?        inet:ip-address
|   +----- src-tp-id?          binary
|   +----- dst-tp-id?          binary
|   +----- bidirectional
|   |   +----- association
|   |   |   +----- id?         uint16
|   |   |   +----- source?     inet:ip-address
|   |   |   +----- global-source? inet:ip-address
|   |   |   +----- type?       identityref
|   |   |   +----- provisioning? identityref
|   +----- explicit-route-objects
|   |   +----- route-object-exclude-always* [index]
|   |   |   +----- index        uint32
|   |   |   +----- (type)?
|   |   |   |   +---:(numbered)
|   |   |   |   |   +----- numbered-hop
|   |   |   |   |   |   +----- address?      te-types:te-tp-id
|   |   |   |   |   |   +----- hop-type?     te-hop-type
|   |   |   |   |   |   +----- direction?    te-link-direction
|   |   |   |   +---:(as-number)
|   |   |   |   |   +----- as-number-hop
|   |   |   |   |   |   +----- as-number?     binary
|   |   |   |   |   |   +----- hop-type?     te-hop-type
|   |   |   |   +---:(unnumbered)
|   |   |   |   |   +----- unnumbered-hop
|   |   |   |   |   |   +----- node-id?      te-types:te-node-id

```

```

|         +---- link-tp-id?    te-types:te-tp-id
|         +---- hop-type?     te-hop-type
|         +---- direction?    te-link-direction
|         +---:(label)
|         +---- label-hop
|         +---- te-label
|         +---- (technology)?
|         |         +---:(generic)
|         |         +---- generic?      rt-
types:generalized-label
|         +---- direction?    te-label-direction
+---- route-object-include-exclude* [index]
+---- explicit-route-usage?   identityref
+---- index                   uint32
+---- (type)?
+---:(numbered)
|         +---- numbered-hop
|         +---- address?      te-types:te-tp-id
|         +---- hop-type?     te-hop-type
|         +---- direction?    te-link-direction
+---:(as-number)
|         +---- as-number-hop
|         +---- as-number?    binary
|         +---- hop-type?     te-hop-type
+---:(unnumbered)
|         +---- unnumbered-hop
|         +---- node-id?      te-types:te-node-id
|         +---- link-tp-id?   te-types:te-tp-id
|         +---- hop-type?     te-hop-type
|         +---- direction?    te-link-direction
+---:(label)
|         +---- label-hop
|         +---- te-label
|         +---- (technology)?
|         |         +---:(generic)
|         |         +---- generic?      rt-
types:generalized-label
|         +---- direction?    te-label-direction
+---- path-constraints

```

```

+---- te-bandwidth
|   +---- (technology)?
|       +---:(generic)
|           +---- generic?    te-bandwidth
+---- setup-priority?        uint8
+---- hold-priority?         uint8
+---- signaling-type?        identityref
+---- disjointness?          te-types:te-path-disjointness
+---- path-metric-bounds
|   +---- path-metric-bound* [metric-type]
|       +---- metric-type    identityref
|       +---- upper-bound?   uint64
+---- path-affinities
|   +---- constraint* [usage]
|       +---- usage          identityref
|       +---- value?         admin-groups
+---- path-srlgs
|   +---- usage?             identityref
|   +---- values*           srlg
+---- optimizations
|   +---- (algorithm)?
|       +---:(metric) {path-optimization-metric}?
|           +---- optimization-metric* [metric-type]
|               +---- metric-type
identityref
|   +---- weight?            uint8
|   +---- explicit-route-exclude-objects
|       +---- route-object-exclude-object* [index]
|           +---- index          uint32
|           +---- (type)?
|               +---:(numbered)
|                   +---- numbered-hop
|                       +---- address?    te-types:te-tp-
id
|   +---- hop-type?          te-hop-type
|   +---- direction?         te-link-
direction
|   +---:(as-number)
|       +---- as-number-hop

```

						+---- as-number?	binary
						+---- hop-type?	te-hop-type
						---:(unnumbered)	
						+---- unnumbered-hop	
node-id						+---- node-id?	te-types:te-
tp-id						+---- link-tp-id?	te-types:te-
						+---- hop-type?	te-hop-type
direction						+---- direction?	te-link-
						---:(label)	
						+---- label-hop	
						+---- te-label	
						+---- (technology)?	
						+---:(generic)	
types:generalized-label						+---- generic?	rt-
						+---- direction?	te-label-
direction							
						+---- explicit-route-include-objects	
						+---- route-object-include-object* [index]	
						+---- index	uint32
						+---- (type)?	
						---:(numbered)	
						+---- numbered-hop	
id						+---- address?	te-types:te-tp-
						+---- hop-type?	te-hop-type
direction						+---- direction?	te-link-
						---:(as-number)	
						+---- as-number-hop	
						+---- as-number?	binary
						+---- hop-type?	te-hop-type
						---:(unnumbered)	
						+---- unnumbered-hop	
node-id						+---- node-id?	te-types:te-

```

tp-id | | | | | +---- link-tp-id? te-types:te-
| | | | | +---- hop-type? te-hop-type
| | | | | +---- direction? te-link-
direction
| | | | | +---:(label)
| | | | | +---- label-hop
| | | | | +---- te-label
| | | | | +---- (technology)?
| | | | | | +---:(generic)
| | | | | | +---- generic? rt-
types:generalized-label
| | | | | +---- direction? te-label-
direction
| | | | | +---- tiebreakers
| | | | | | +---- tiebreaker* [tiebreaker-type]
| | | | | | +---- tiebreaker-type identityref
| | | | | +---:(objective-function) {path-optimization-objective-
function}?
| | | | | +---- objective-function
| | | | | | +---- objective-function-type? identityref
| | | | | +---- requested-metrics* [metric-type]
| | | | | | +---- metric-type identityref
+---- synchronization* [synchronization-id]
+---- synchronization-id uint32
+---- svec
| +---- relaxable? boolean
| +---- link-diverse? boolean
| +---- node-diverse? boolean
| +---- srlg-diverse? boolean
| +---- request-id-number* uint32
+---- svec-constraints
| +---- path-metric-bound* [metric-type]
| | +---- metric-type identityref
| | +---- upper-bound? uint64
+---- path-srlgs
| +---- usage? identityref
| +---- values* srlg
+---- exclude-objects

```

```

+---- excludes* [index]
+---- index          uint32
+---- (type)?
+--:(numbered)
+---- numbered-hop
+---- address?       te-types:te-tp-id
+---- hop-type?      te-hop-type
+---- direction?     te-link-direction
+--:(as-number)
+---- as-number-hop
+---- as-number?     binary
+---- hop-type?      te-hop-type
+--:(unnumbered)
+---- unnumbered-hop
+---- node-id?       te-types:te-node-id
+---- link-tp-id?    te-types:te-tp-id
+---- hop-type?      te-hop-type
+---- direction?     te-link-direction
+--:(label)
+---- label-hop
+---- te-label
+---- (technology)?
+---- |(generic)
+---- |generic?      rt-
types:generalized-label
+---- direction?     te-label-direction
+---- optimizations
+---- (algorithm)?
+--:(metric)
+---- optimization-metric* [metric-type]
+---- metric-type     identityref
+---- weight?         uint8
+--:(objective-function)
+---- objective-function
+---- objective-function-type? identityref
augment /te:tunnels-rpc/te:output/te:result:
+--ro response* [response-id]
+--ro response-id     uint32
+--ro (response-type)?

```

```

+---:(no-path-case)
|   +---ro no-path!
+---:(path-case)
+---ro computed-path
+---ro path-id?          yang-types:uuid
+---ro path-properties
+---ro path-metric* [metric-type]
|   +---ro metric-type      identityref
|   +---ro accumulative-value?  uint64
+---ro path-affinities
|   +---ro constraint* [usage]
|   +---ro usage            identityref
|   +---ro value?          admin-groups
+---ro path-srlgs
|   +---ro usage?          identityref
|   +---ro values*        srlg
+---ro path-route-objects
+---ro path-route-object* [index]
+---ro index              uint32
+---ro (type)?
+---:(numbered)
|   +---ro numbered-hop
|   +---ro address?        te-types:te-tp-
id
|   +---ro hop-type?       te-hop-type
direction
|   +---ro direction?      te-link-
+---:(as-number)
|   +---ro as-number-hop
|   +---ro as-number?      binary
|   +---ro hop-type?       te-hop-type
+---:(unnumbered)
|   +---ro unnumbered-hop
|   +---ro node-id?        te-types:te-
node-id
|   +---ro link-tp-id?     te-types:te-
tp-id
|   +---ro hop-type?       te-hop-type

```



```

direction
|      +--ro direction?    te-link-
+--:(label)
  +--ro label-hop
    +--ro te-label
      +--ro (technology)?
        | +--:(generic)
        |   +--ro generic?    rt-
types:generalized-label
direction
      +--ro direction?    te-label-

```

Figure 9

- TE path computation YANG

G tree

6.2. YANG Module

```

<CODE BEGINS>file "ietf-te-path-computation@2018-03-02.yang"
module iETF-te-path-computation {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-path-computation";
  // replace with IANA namespace when assigned

  prefix "tepc";

  import iETF-inet-types {
    prefix "inet";
  }

  import iETF-yang-types {
    prefix "yang-types";
  }

  import iETF-te {
    prefix "te";
  }

  import iETF-te-types {
    prefix "te-types";
  }

```

```
organization
  "Traffic Engineering Architecture and Signaling (TEAS)
  Working Group";

contact
  "WG Web:    <http://tools.ietf.org/wg/teas/>
  WG List:    <mailto:teas@ietf.org>

  WG Chair: Lou Berger
              <mailto:lberger@labn.net>

  WG Chair: Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  ";

description "YANG model for stateless TE path computation";

revision "2018-03-02" {
  description "Revision to fix issues #22, 29, 33 and 39";
  reference "YANG model for stateless TE path computation";
}

/*
 * Features
 */

feature stateless-path-computation {
  description
    "This feature indicates that the system supports
    stateless path computation.";
}

/*
 * Groupings
 */

grouping path-info {
```

```
    leaf path-id {
      type yang-types:uuid;
      config false;
      description "path-id ref.";
    }
    uses te-types:generic-path-properties;
    description "Path computation output information";
  }

  grouping end-points {
    leaf source {
      type inet:ip-address;
      description "TE tunnel source address.";
    }
    leaf destination {
      type inet:ip-address;
      description "P2P tunnel destination address";
    }
    leaf src-tp-id {
      type binary;
      description "TE tunnel source termination point identifier.";
    }
    leaf dst-tp-id {
      type binary;
      description "TE tunnel destination termination point
identifier.";
    }
    description "Path Computation End Points grouping.";
  }

  grouping requested-metrics-info {
    description "requested metric";
    list requested-metrics {
      key 'metric-type';
      description "list of requested metrics";
      leaf metric-type {
        type identityref {
          base te-types:path-metric-type;
        }
      }
    }
  }
```

```
        description "the requested metric";
    }
}

identity svec-metric-type {
    description
        "Base identity for svec metric type";
}

identity svec-metric-cumul-te {
    base svec-metric-type;
    description
        "TE cumulative path metric";
}

identity svec-metric-cumul-igp {
    base svec-metric-type;
    description
        "IGP cumulative path metric";
}

identity svec-metric-cumul-hop {
    base svec-metric-type;
    description
        "Hop cumulative path metric";
}

identity svec-metric-aggregate-bandwidth-consumption {
    base svec-metric-type;
    description
        "Cumulative bandwith consumption of the set of synchronized
paths";
}

identity svec-metric-load-of-the-most-loaded-link {
    base svec-metric-type;
    description
        "Load of the most loaded link";
}
```

```
    }

    grouping svec-metrics-bounds_config {
      description "TE path metric bounds grouping for computing a set
of
      synchronized requests";
      leaf metric-type {
        type identityref {
          base svec-metric-type;
        }
        description "TE path metric type usable for computing a set of
          synchronized requests";
      }
      leaf upper-bound {
        type uint64;
        description "Upper bound on end-to-end svec path metric";
      }
    }

    grouping svec-metrics-optimization_config {
      description "TE path metric bounds grouping for computing a set
of
      synchronized requests";
      leaf metric-type {
        type identityref {
          base svec-metric-type;
        }
        description "TE path metric type usable for computing a set of
          synchronized requests";
      }
      leaf weight {
        type uint8;
        description "Metric normalization weight";
      }
    }

    grouping svec-exclude {
      description "List of resources to be excluded by all the paths
        in the SVEC";
```

```
    container exclude-objects {
      description "resources to be excluded";
      list excludes {
        key index;
        description
          "List of explicit route objects to always exclude
           from synchronized path computation";
        uses te-types:explicit-route-hop;
      }
    }
  }

  grouping synchronization-constraints {
    description "Global constraints applicable to synchronized
      path computation";
    container svec-constraints {
      description "global svec constraints";
      list path-metric-bound {
        key metric-type;
        description "list of bound metrics";
        uses svec-metrics-bounds_config;
      }
    }
    uses te-types:generic-path-srlgs;
    uses svec-exclude;
  }

  grouping synchronization-optimization {
    description "Synchronized request optimization";
    container optimizations {
      description
        "The objective function container that includes
         attributes to impose when computing a synchronized set of
        paths";

      choice algorithm {
        description "Optimizations algorithm.";
        case metric {
          list optimization-metric {
```

```

        key "metric-type";
        description "svec path metric type";
        uses svec-metrics-optimization_config;
    }
}
case objective-function {
    container objective-function {
        description
            "The objective function container that includes
            attributes to impose when computing a TE path";
        uses te-types:path-objective-function_config;
    }
}
}
}

grouping synchronization-info {
    description "Information for sync";
    list synchronization {
        key "synchronization-id";
        description "sync list";
        leaf synchronization-id {
            type uint32;
            description "index";
        }
    }
    container svec {
        description
            "Synchronization VECtor";
        leaf relaxable {
            type boolean;
            default true;
            description
                "If this leaf is true, path computation process is free
to ignore svec content.
                otherwise it must take into account this svec.";
        }
        leaf link-diverse {
            type boolean;

```

```
        default false;
        description "link-diverse";
    }
    leaf node-diverse {
        type boolean;
        default false;
        description "node-diverse";
    }
    leaf srlg-diverse {
        type boolean;
        default false;
        description "srlg-diverse";
    }
    leaf-list request-id-number {
        type uint32;
        description "This list reports the set of M path
computation requests that must be synchronized.";
    }
}
uses synchronization-constraints;
uses synchronization-optimization;
}

grouping no-path-info {
    description "no-path-info";
    container no-path {
        presence "Response without path information, due to failure
performing the path computation";
        description "if path computation cannot identify a path,
rpc returns no path.";
    }
}

/*
 * Root container
 */
container paths {
```



```
list path {
  key "path-id";
  config false;
  uses path-info;
  description "List of previous computed paths.";
}
description "Root container for path-computation";
}

/**
 * AUGMENTS TO TE RPC
 */

augment "/te:tunnels-rpc/te:input/te:tunnel-info" {
  description "statelessComputeP2PPath input";
  list path-request {
    key "request-id";
    description "request-list";
    leaf request-id {
      type uint32;
      mandatory true;
      description "Each path computation request is uniquely
identified by the request-id-number.
      It must be present also in rpcs.";
    }
    uses end-points;
    uses te:bidir-assoc-properties;
    uses te-types:path-route-objects;
    uses te-types:generic-path-constraints;
    uses te-types:generic-path-optimization;
    uses requested-metrics-info;
  }
  uses synchronization-info;
}

augment "/te:tunnels-rpc/te:output/te:result" {
  description "statelessComputeP2PPath output";
  list response {
    key response-id;
```

```

config false;
description "response";
leaf response-id {
    type uint32;
    description
        "The list key that has to reuse request-id-number.";
}
choice response-type {
    config false;
    description "response-type";
    case no-path-case {
        uses no-path-info;
    }
    case path-case {
        container computed-path {
            uses path-info;
            description "Path computation service.";
        }
    }
}
}
}
}
}
<CODE ENDS>
```

Figure 10 - TE path computation YANG module

7. Security Considerations

This document describes use cases of requesting Path Computation using YANG models, which could be used at the ABNO Control Interface [RFC7491] and/or between controllers in ACTN [ACTN-frame]. As such, it does not introduce any new security considerations compared to the ones related to YANG specification, ABNO specification and ACTN Framework defined in [RFC6020], [RFC7950], [RFC7491] and [ACTN-frame].

This document also defines common data types using the YANG data modeling language. The definitions themselves have no security impact on the Internet, but the usage of these definitions in concrete YANG modules might have. The security considerations

spelled out in the YANG specification [RFC6020] apply for this document as well.

8. IANA Considerations

This section is for further study: to be completed when the YANG model is more stable.

9. References

9.1. Normative References

- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC7139] Zhang, F. et al., "GMPLS Signaling Extensions for Control of Evolving G.709 Optical Transport Networks", RFC 7139, March 2014.
- [RFC7491] Farrel, A., King, D., "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, March 2015.
- [RFC7926] Farrel, A. et al., "Problem Statement and Architecture for Information Exchange Between Interconnected Traffic Engineered Networks", RFC 7926, July 2016.
- [RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.
- [TE-TOPO] Liu, X. et al., "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo, work in progress.
- [TE-TUNNEL] Saad, T. et al., "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.
- [ACTN-Frame] Ceccarelli, D., Lee, Y. et al., "Framework for Abstraction and Control of Traffic Engineered Networks" draft-ietf-actn-framework, work in progress.
- [ITU-T G.709-2016] ITU-T Recommendation G.709 (06/16), "Interface for the optical transport network", June 2016

9.2. Informative References

- [RFC4655] Farrel, A. et al., "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.
- [RFC5541] Le Roux, JL. et al., " Encoding of Objective Functions in the Path Computation Element Communication Protocol (PCEP)", RFC 5541, June 2009.
- [RFC7446] Lee, Y. et al., "Routing and Wavelength Assignment Information Model for Wavelength Switched Optical Networks", RFC 7446, February 2015.
- [OTN-TOPO] Zheng, H. et al., "A YANG Data Model for Optical Transport Network Topology", draft-ietf-ccamp-otn-topo-yang, work in progress.
- [ACTN-Info] Lee, Y., Belotti, S., Dhody, D., Ceccarelli, D., "Information Model for Abstraction and Control of Transport Networks", draft-leebelotti-actn-info, work in progress.
- [PCEP-Service-Aware] Dhody, D. et al., "Extensions to the Path Computation Element Communication Protocol (PCEP) to compute service aware Label Switched Path (LSP)", draft-ietf-pce-pcep-service-aware, work in progress.

10. Acknowledgments

The authors would like to thank Igor Bryskin and Xian Zhang for participating in discussions and providing valuable insights.

The authors would like to thank the authors of the TE Tunnel YANG model [TE-TUNNEL], in particular Igor Bryskin, Vishnu Pavan Beeram, Tarek Saad and Xufeng Liu, for their inputs to the discussions and support in having consistency between the Path Computation and TE Tunnel YANG models.

This document was prepared using 2-Word-v2.0.template.dot.

Appendix A. Examples of dimensioning the "detailed connectivity matrix"

In the following table, a list of the possible constraints, associated with their potential cardinality, is reported.

The maximum number of potential connections to be computed and reported is, in first approximation, the multiplication of all of them.

Constraint	Cardinality
End points	$N(N-1)/2$ if connections are bidirectional (OTN and WDM), $N(N-1)$ for unidirectional connections.
Bandwidth	<p>In WDM networks, bandwidth values are expressed in GHz.</p> <p>On fixed-grid WDM networks, the central frequencies are on a 50GHz grid and the channel width of the transmitters are typically 50GHz such that each central frequency can be used, i.e., adjacent channels can be placed next to each other in terms of central frequencies.</p> <p>On flex-grid WDM networks, the central frequencies are on a 6.25GHz grid and the channel width of the transmitters can be multiples of 12.5GHz.</p> <p>For fixed-grid WDM networks typically there is only one possible bandwidth value (i.e., 50GHz) while for flex-grid WDM networks typically there are 4 possible bandwidth values (e.g., 37.5GHz, 50GHz, 62.5GHz, 75GHz).</p> <p>In OTN (ODU) networks, bandwidth values are expressed as pairs of ODU type and, in case of ODUFlex, ODU rate in bytes/sec as described in section 5 of [RFC7139].</p> <p>For "fixed" ODUk types, 6 possible bandwidth values are possible (i.e., ODU0, ODU1, ODU2, ODU2e, ODU3, ODU4).</p> <p>For ODUFlex(GFP), up to 80 different bandwidth values can be specified, as defined in Table 7-8 of [ITU-T G.709-2016].</p> <p>For other ODUFlex types, like ODUFlex(CBR), the number of possible bandwidth values depends on the rates of the</p>

clients that could be mapped over these ODUFlex types, as shown in Table 7.2 of [ITU-T G.709-2016], which in theory could be a countinuum of values. However, since different ODUFlex bandwidths that use the same number of TSs on each link along the path are equivalent for path computation purposes, up to 120 different bandwidth ranges can be specified.

Ideas to reduce the number of ODUFlex bandwidth values in the detailed connectivity matrix, to less than 100, are for further study.

Bandwidth specification for ODUCn is currently for further study but it is expected that other bandwidth values can be specified as integer multiples of 100Gb/s.

In IP we have bandwidth values in bytes/sec. In principle, this is a countinuum of values, but in practice we can identify a set of bandwidth ranges, where any bandwidth value inside the same range produces the same path.

The number of such ranges is the cardinality, which depends on the topology, available bandwidth and status of the network. Simulations (Note: reference paper submitted for publication) show that values for medium size topologies (around 50-150 nodes) are in the range 4-7 (5 on average) for each end points couple.

Metrics IGP, TE and hop number are the basic objective metrics defined so far. There are also the 2 objective functions defined in [RFC5541]: Minimum Load Path (MLP) and Maximum Residual Bandwidth Path (MBP). Assuming that one only metric or objective function can be optimized at once, the total cardinality here is 5.

With [PCEP-Service-Aware], a number of additional metrics are defined, including Path Delay metric, Path Delay Variation metric and Path Loss metric, both for point-to-point and point-to-multipoint paths. This increases the cardinality to 8.

Bounds Each metric can be associated with a bound in order to find a path having a total value of that metric lower than the given bound. This has a potentially very high cardinality (as any value for the bound is allowed). In

practice there is a maximum value of the bound (the one with the maximum value of the associated metric) which results always in the same path, and a range approach like for bandwidth in IP should produce also in this case the cardinality. Assuming to have a cardinality similar to the one of the bandwidth (let say 5 on average) we should have 6 (IGP, TE, hop, path delay, path delay variation and path loss; we don't consider here the two objective functions of [RFC5541] as they are conceived only for optimization)*5 = 30 cardinality.

Technology

constraints For further study

Priority We have 8 values for setup priority, which is used in path computation to route a path using free resources and, where no free resources are available, resources used by LSPs having a lower holding priority.

Local prot It's possible to ask for a local protected service, where all the links used by the path are protected with fast reroute (this is only for IP networks, but line protection schemas are available on the other technologies as well). This adds an alternative path computation, so the cardinality of this constraint is 2.

Administrative

Colors Administrative colors (aka affinities) are typically assigned to links but when topology abstraction is used affinity information can also appear in the detailed connectivity matrix.

There are 32 bits available for the affinities. Links can be tagged with any combination of these bits, and path computation can be constrained to include or exclude any or all of them. The relevant cardinality is 3 (include-any, exclude-any, include-all) times 2^{32} possible values. However, the number of possible values used in real networks is quite small.

Included Resources

A path computation request can be associated to an ordered set of network resources (links, nodes) to be included along the computed path. This constraint would

have a huge cardinality as in principle any combination of network resources is possible. However, as far as the Orchestrator doesn't know details about the internal topology of the domain, it shouldn't include this type of constraint at all (see more details below).

Excluded Resources

A path computation request can be associated to a set of network resources (links, nodes, SRLGs) to be excluded from the computed path. Like for included resources, this constraint has a potentially very high cardinality, but, once again, it can't be actually used by the Orchestrator, if it's not aware of the domain topology (see more details below).

As discussed above, the Orchestrator can specify include or exclude resources depending on the abstract topology information that the domain controller exposes:

- o In case the domain controller exposes the entire domain as a single abstract TE node with his own external terminations and connectivity matrix (whose size we are estimating), no other topological details are available, therefore the size of the connectivity matrix only depends on the combination of the constraints that the Orchestrator can use in a path computation request to the domain controller. These constraints cannot refer to any details of the internal topology of the domain, as those details are not known to the Orchestrator and so they do not impact size of connectivity matrix exported.
- o Instead in case the domain controller exposes a topology including more than one abstract TE nodes and TE links, and their attributes (e.g. SRLGs, affinities for the links), the Orchestrator knows these details and therefore could compute a path across the domain referring to them in the constraints. The connectivity matrixes to be estimated here are the ones relevant to the abstract TE nodes exported to the Orchestrator. These connectivity matrixes and therefore their sizes, while cannot depend on the other abstract TE nodes and TE links, which are external to the given abstract node, could depend to SRLGs (and other attributes, like affinities) which could be present also in the portion of the topology represented by the abstract nodes, and therefore contribute to the size of the related connectivity matrix.

We also don't consider here the possibility to ask for more than one path in diversity or for point-to-multi-point paths, which are for further study.

Considering for example an IP domain without considering SRLG and affinities, we have an estimated number of paths depending on these estimated cardinalities:

Endpoints = $N(N-1)$, Bandwidth = 5, Metrics = 6, Bounds = 20,
Priority = 8, Local prot = 2

The number of paths to be pre-computed by each IP domain is therefore $24960 * N(N-1)$ where N is the number of domain access points.

This means that with just 4 access points we have nearly 300000 paths to compute, advertise and maintain (if a change happens in the domain, due to a fault, or just the deployment of new traffic, a substantial number of paths need to be recomputed and the relevant changes advertised to the upper controller).

This seems quite challenging. In fact, if we assume a mean length of 1K for the json describing a path (a quite conservative estimate), reporting 300000 paths means transferring and then parsing more than 300 Mbytes for each domain. If we assume that 20% (to be checked) of this paths change when a new deployment of traffic occurs, we have 60 Mbytes of transfer for each domain traversed by a new end-to-end path. If a network has, let say, 20 domains (we want to estimate the load for a non-trivial domain setup) in the beginning a total initial transfer of 6Gigs is needed, and eventually, assuming 4-5 domains are involved in mean during a path deployment we could have 240-300 Mbytes of changes advertised to the higher order controller.

Further bare-bone solutions can be investigated, removing some more options, if this is considered not acceptable; in conclusion, it seems that an approach based only on connectivity matrix is hardly feasible, and could be applicable only to small networks with a limited meshing degree between domains and renouncing to a number of path computation features.

Contributors

Dieter Beller
Nokia
Email: dieter.beller@nokia.com

Gianmarco Bruno
Ericsson
Email: gianmarco.bruno@ericsson.com

Francesco Lazzeri
Ericsson
Email: francesco.lazzeri@ericsson.com

Young Lee
Huawei
Email: leeyoung@huawei.com

Carlo Perocchio
Ericsson
Email: carlo.perocchio@ericsson.com

Authors' Addresses

Italo Busi (Editor)
Huawei
Email: italo.busi@huawei.com

Sergio Belotti (Editor)
Nokia
Email: sergio.belotti@nokia.com

Victor Lopez
Telefonica
Email: victor.lopezalvarez@telefonica.com

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

Anurag Sharma
Google
Email: ansha@google.com

Yan Shi
China Unicom
Email: shiyan49@chinaunicom.cn

Ricard Vilalta
CTTC
Email: ricard.vilalta@cttc.es

Karthik Sethuraman
NEC
Email: karthik.sethuraman@necam.com

Michael Scharf
Nokia
Email: michael.scharf@nokia.com

Daniele Ceccarelli
Ericsson
Email: daniele.ceccarelli@ericsson.com

TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2018

V. Beeram
Juniper Networks
T. Saad, Ed.
R. Gandhi
Cisco Systems, Inc.
X. Liu
Jabil
I. Bryskin
Huawei Technologies
H. Shah
Ciena
October 29, 2017

A YANG Data Model for Resource Reservation Protocol (RSVP)
draft-ietf-teas-yang-rsvp-08

Abstract

This document defines a YANG data model for the configuration and management of RSVP Protocol. The model covers the building blocks of the RSVP protocol that can be augmented and used by other RSVP extension models such as RVSP extensions to Traffic-Engineering (RSVP-TE). The model covers the configuration, operational state, remote procedural calls, and event notifications data.

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 May 2, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
1.2. Tree Diagram	3
1.3. Prefixes in Data Node Names	4
2. Design Considerations	5
2.1. Module Hierarchy	5
2.2. State Data Organization	6
2.3. Configuration Inheritance	6
3. Model Organization	7
3.1. RSVP Base YANG Model	7
3.1.1. Global Data	8
3.1.2. Interface Data	8
3.1.3. Neighbor Data	9
3.1.4. Session Data	9
3.1.5. Tree Diagram	9
3.1.6. YANG Module	13
3.2. RSVP Extended YANG Model	31
3.2.1. Tree Diagram	32
3.2.2. YANG Module	33
4. IANA Considerations	44
5. Security Considerations	44
6. Acknowledgement	44
7. Contributors	45
8. References	45
8.1. Normative References	45
8.2. Informative References	46
Authors' Addresses	47

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of

implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage the RSVP protocol [RFC2205]. This model covers RSVP protocol building blocks that can be augmented and used by other RSVP extension models- such as for signaling RSVP-TE MPLS (or other technology specific) Label Switched Paths (LSP)s.

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

1.2. Tree Diagram

A simplified graphical representation of the data model is presented in each section of the model. The following notations are used for the YANG model data tree representation.

<status> <flags> <name> <opts> <type>

<status> is one of:

- + for current
- x for deprecated
- o for obsolete

<flags> is one of:

- rw for read-write configuration data
- ro for read-only non-configuration data
- x for execution rpcs
- n for notifications

<name> is the name of the node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>

<opts> is one of:

- ? for an optional leaf or node
- ! for a presence container
- * for a leaf-list or list

Brackets [<keys>] for a list's keys

Curly braces {<condition>} for optional feature that make node conditional

Colon : for marking case nodes

Ellipses ("...") subtree contents not shown

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

<type> is the name of the type for leafs and leaf-lists.

1.3. 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
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
rt-type	ietf-routing-types	XX
key-chain	ietf-key-chain	XX

Table 1: Prefixes and corresponding YANG modules

2. Design Considerations

2.1. Module Hierarchy

The RSVP base YANG module augments the "control-plane-protocol" list in ietf-routing [RFC8022] module with specific RSVP parameters in an "rsvp" container. It also defines an extension identity "rsvp" of base "rt:routing-protocol" to identify the RSVP protocol.

During modeling discussion, some RSVP features are categorized as core to the functionality of the protocol, and hence, are supported by all vendors claiming the support for RSVP. These features' configuration and state were grouped in the RSVP base module.

Other extended RSVP features are categorized as either optional or providing additional knobs to provide better tune basic functionality of the RSVP protocol. The support for extended RSVP features by all vendors was considered optional. Such features were grouped in a separate RSVP extended module.

The augmentation of the RSVP model by other models (e.g. RSVP-TE for MPLS or other technologies) are considered outside the scope of this document and discussed in separate document(s).

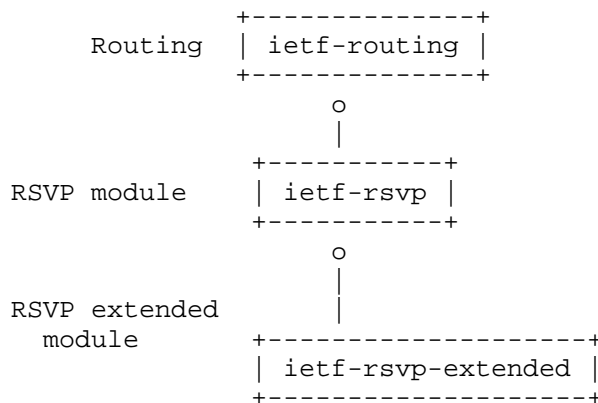


Figure 1: Relationship of RSVP and RSVP extended modules with other protocol modules

The RSVP base model does not aim to be feature complete. The primary intent is to cover a set of standard core features (listed below) that are commonly in use.

- o Authentication ([RFC2747])
- o Refresh Reduction ([RFC2961])
- o Hellos ([RFC3209])
- o Graceful Restart ([RFC3473], [RFC5063])

The extended RSVP YANG model covers non-basic configuration(s) for RSVP feature(s) as well as optional RSVP feature that are not a must for basic RSVP operation.

2.2. State Data Organization

The Network Management Datastore Architecture (NMDA) [I-D.dsdt-nmda-guidelines] addresses the "OpState" that was discussed in the IETF. As per NMDA guidelines for new models and models that are not concerned with the operational state of configuration information, this revision of the draft adopts the NMDA proposal for configuration and state data of this model.

2.3. Configuration Inheritance

The defined data model supports configuration inheritance for neighbors, and interfaces. Data elements defined in the main container (e.g. the container that encompasses the list of

interfaces, or neighbors) are assumed to apply equally to all elements of the list, unless overridden explicitly for a certain element (e.g. interface). Vendors are expected to augment the above container(s) to provide the list of inheritance command for their implementations.

3. Model Organization

This document divides the RSVP model into two modules: the RSVP base and extended. Each module covers the configuration, state, notification and RPCs of data. The relationship between the different modules is depicted in Figure 1.

3.1. RSVP Base YANG Model

This section describes the RSVP base YANG data model. The container "rsvp" is the top level container in this data model. The presence of this container enables the RSVP protocol functionality.

Data for such state is contained under the respective "state" sub-container of the intended object (e.g. interface) as shown in Figure 2.

```
module: ietf-rsvp
  +--rw rsvp!
    +--rw globals
      +-- rw config
        <<intended configuration>>
      .
      +-- ro state
        <<applied configuration>>
        <<derived state associated with the tunnel>>
      .
    +--rw interfaces
      +-- rw config
        <<intended configuration>>
      .
      +-- ro state
        <<applied configuration>>
        <<derived state associated with the tunnel>>
      .
    +--rw neighbors
      +-- rw config
        <<intended configuration>>
      .
      +-- ro state
```

```

        <<applied configuration>>
        <<derived state associated with the tunnel>>
    .
    .
+--rw sessions
    +-- rw config
        <<intended configuration>>
    .
    +-- ro state
        <<applied configuration>>
        <<derived state associated with the tunnel>>
    .
rpcs:
    +--x global-rpc
    +--x interfaces-rpc
    +--x neighbors-rpc
    +--x sessions-rpc
notifications:
    +--n global-notif
    +--n interfaces-notif
    +--n neighbors-notif
    +--n sessions-notif

```

Figure 2: RSVP high-level tree model view

The following subsections provide overview of the parts of the model pertaining to configuration and state data.

Configuration and state data are organized into those applicable globally (node scope), per interfaces, per neighbors, or per session.

3.1.1. Global Data

This branch of the data model covers global configuration and states that control RSVP protocol behavior.

3.1.2. Interface Data

This branch of the data model covers configuration and state elements relevant to one or all RSVP interfaces. Any data configuration applied at the "interfaces" container level are equally applicable to all interfaces - unless overridden by explicit configuration under a specific interface.

3.1.3. Neighbor Data

This branch of the data model covers configuration of elements relevant to RSVP neighbors. This would be discussed in detail in future revisions.

3.1.4. Session Data

This branch of the data model covers configuration of elements relevant to RSVP sessions. This would be discussed in detail in future revisions.

3.1.5. Tree Diagram

```

module: ietf-rsvp
augment
  /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
    +--rw rsvp!
      +--rw globals
        +--rw sessions
          +--ro session* [local-index]
            +--ro local-index    -> ../state/local-index
            +--ro state
              +--ro local-index?      uint64
              +--ro destination-port?  inet:port-number
              +--ro source?            inet:ip-address
              +--ro destination?       inet:ip-address
              +--ro session-name?      string
              +--ro session-state?     enumeration
              +--ro session-type?      identityref
              +--ro psbs
                +--ro psb*
                  +--ro source-port?  inet:port-number
                  +--ro expires-in?    uint32
              +--ro rsbs
                +--ro rsb*
                  +--ro source-port?  inet:port-number
                  +--ro reservation-style? identityref
                  +--ro expires-in?    uint32
          +--rw statistics
            +--ro state
              +--ro messages
                +--ro ack-sent?          yang:counter64
                +--ro ack-received?      yang:counter64
                +--ro bundle-sent?       yang:counter64
                +--ro bundle-received?   yang:counter64
                +--ro hello-sent?        yang:counter64
                +--ro hello-received?    yang:counter64

```

```

| | | +--ro integrity-challenge-sent?      yang:counter64
| | | +--ro integrity-challenge-received?  yang:counter64
| | | +--ro integrity-response-sent?      yang:counter64
| | | +--ro integrity-response-received?  yang:counter64
| | | +--ro notify-sent?                  yang:counter64
| | | +--ro notify-received?              yang:counter64
| | | +--ro path-sent?                    yang:counter64
| | | +--ro path-received?                yang:counter64
| | | +--ro path-err-sent?                yang:counter64
| | | +--ro path-err-received?            yang:counter64
| | | +--ro path-tear-sent?                yang:counter64
| | | +--ro path-tear-received?            yang:counter64
| | | +--ro resv-sent?                     yang:counter64
| | | +--ro resv-received?                 yang:counter64
| | | +--ro resv-confirm-sent?             yang:counter64
| | | +--ro resv-confirm-received?         yang:counter64
| | | +--ro resv-err-sent?                 yang:counter64
| | | +--ro resv-err-received?             yang:counter64
| | | +--ro resv-tear-sent?                yang:counter64
| | | +--ro resv-tear-received?            yang:counter64
| | | +--ro summary-refresh-sent?          yang:counter64
| | | +--ro summary-refresh-received?      yang:counter64
| | | +--ro unknown-messages-received?    yang:counter64
| | | +--ro packets
| | | | +--ro sent?                       yang:counter64
| | | | +--ro received?                   yang:counter64
| | | +--ro errors
| | | | +--ro authenticate?               yang:counter64
| | | | +--ro checksum?                   yang:counter64
| | | | +--ro packet-len?                 yang:counter64
| | | +--rw graceful-restart
| | | | +--rw enabled?                     boolean
+--rw interfaces
| +--rw refresh-reduction
| | +--rw enabled?                         boolean
+--rw hellos
| +--rw enabled?                           boolean
+--rw authentication
| +--rw enabled?                           boolean
| +--rw authentication-key?                string
| +--rw crypto-algorithm                   identityref
+--rw statistics
| +--ro state
| | +--ro messages
| | | +--ro ack-sent?                      yang:counter64
| | | +--ro ack-received?                  yang:counter64
| | | +--ro bundle-sent?                   yang:counter64
| | | +--ro bundle-received?               yang:counter64

```

```

| | | | | +--ro hello-sent? yang:counter64
| | | | | +--ro hello-received? yang:counter64
| | | | | +--ro integrity-challenge-sent? yang:counter64
| | | | | +--ro integrity-challenge-received? yang:counter64
| | | | | +--ro integrity-response-sent? yang:counter64
| | | | | +--ro integrity-response-received? yang:counter64
| | | | | +--ro notify-sent? yang:counter64
| | | | | +--ro notify-received? yang:counter64
| | | | | +--ro path-sent? yang:counter64
| | | | | +--ro path-received? yang:counter64
| | | | | +--ro path-err-sent? yang:counter64
| | | | | +--ro path-err-received? yang:counter64
| | | | | +--ro path-tear-sent? yang:counter64
| | | | | +--ro path-tear-received? yang:counter64
| | | | | +--ro resv-sent? yang:counter64
| | | | | +--ro resv-received? yang:counter64
| | | | | +--ro resv-confirm-sent? yang:counter64
| | | | | +--ro resv-confirm-received? yang:counter64
| | | | | +--ro resv-err-sent? yang:counter64
| | | | | +--ro resv-err-received? yang:counter64
| | | | | +--ro resv-tear-sent? yang:counter64
| | | | | +--ro resv-tear-received? yang:counter64
| | | | | +--ro summary-refresh-sent? yang:counter64
| | | | | +--ro summary-refresh-received? yang:counter64
| | | | | +--ro unknown-messages-received? yang:counter64
| | | | +--ro packets
| | | | | +--ro sent? yang:counter64
| | | | | +--ro received? yang:counter64
| | | | +--ro errors
| | | | | +--ro authenticate? yang:counter64
| | | | | +--ro checksum? yang:counter64
| | | | | +--ro packet-len? yang:counter64
| | | +--rw interface* [interface]
| | | | +--rw interface if:interface-ref
| | | | +--rw refresh-reduction
| | | | | +--rw enabled? boolean
| | | | +--rw hellos
| | | | | +--rw enabled? boolean
| | | | +--rw authentication
| | | | | +--rw enabled? boolean
| | | | | +--rw authentication-key? string
| | | | | +--rw crypto-algorithm identityref
| | | | +--rw statistics
| | | | | +--ro state
| | | | | | +--ro messages
| | | | | | | +--ro ack-sent?
yang:counter64
| | | | | | | +--ro ack-received?

```

```
yang:counter64
|
| +--ro bundle-sent?
yang:counter64
|
| +--ro bundle-received?
yang:counter64
|
| +--ro hello-sent?
yang:counter64
|
| +--ro hello-received?
yang:counter64
|
| +--ro integrity-challenge-sent?
yang:counter64
|
| +--ro integrity-challenge-received?
yang:counter64
|
| +--ro integrity-response-sent?
yang:counter64
|
| +--ro integrity-response-received?
yang:counter64
|
| +--ro notify-sent?
yang:counter64
|
| +--ro notify-received?
yang:counter64
|
| +--ro path-sent?
yang:counter64
|
| +--ro path-received?
yang:counter64
|
| +--ro path-err-sent?
yang:counter64
|
| +--ro path-err-received?
yang:counter64
|
| +--ro path-tear-sent?
yang:counter64
|
| +--ro path-tear-received?
yang:counter64
|
| +--ro resv-sent?
yang:counter64
|
| +--ro resv-received?
yang:counter64
|
| +--ro resv-confirm-sent?
yang:counter64
|
| +--ro resv-confirm-received?
yang:counter64
|
| +--ro resv-err-sent?
yang:counter64
|
| +--ro resv-err-received?
yang:counter64
|
| +--ro resv-tear-sent?
yang:counter64
|
| +--ro resv-tear-received?
```



```

yang:counter64
|
|   +---ro summary-refresh-sent?
yang:counter64
|
|   +---ro summary-refresh-received?
yang:counter64
|
|   +---ro unknown-messages-received?
yang:counter64
|
|   +---ro packets
|   |
|   |   +---ro sent?          yang:counter64
|   |   +---ro received?     yang:counter64
|   |
|   |   +---ro errors
|   |   |
|   |   |   +---ro authenticate?  yang:counter64
|   |   |   +---ro checksum?      yang:counter64
|   |   |   +---ro packet-len?    yang:counter64
|   |
|   +---rw neighbors
|   |
|   |   +---rw neighbor* [address]
|   |   |
|   |   |   +---rw address      inet:ip-address
|   |   |   +---ro state
|   |   |   |
|   |   |   |   +---ro address?          inet:ip-address
|   |   |   |   +---ro epoch?            uint32
|   |   |   |   +---ro expiry-time?      uint32
|   |   |   |   +---ro graceful-restart
|   |   |   |   |
|   |   |   |   |   +---ro enabled?      boolean
|   |   |   |   |   +---ro local-restart-time?  uint32
|   |   |   |   |   +---ro local-recovery-time?  uint32
|   |   |   |   |   +---ro neighbor-restart-time?  uint32
|   |   |   |   |   +---ro neighbor-recovery-time?  uint32
|   |   |   |   +---ro helper-mode
|   |   |   |   |
|   |   |   |   |   +---ro enabled?      boolean
|   |   |   |   |   +---ro max-helper-restart-time?  uint32
|   |   |   |   |   +---ro max-helper-recovery-time?  uint32
|   |   |   |   |   +---ro neighbor-restart-time-remaining?  uint32
|   |   |   |   |   +---ro neighbor-recovery-time-remaining?  uint32
|   |   |   +---ro hello-status?      enumeration
|   |   +---ro interface?              if:interface-ref
|   |   +---ro neighbor-state?          enumeration
|   |   +---ro refresh-reduction-capable?  boolean
|   |   +---ro restart-count?            yang:counter32
|   |   +---ro restart-time?             yang:date-and-time

```

Figure 3: RSVP model tree diagram

3.1.6. YANG Module

```

<CODE BEGINS> file "ietf-rsvp@2017-10-29.yang"
module ietf-rsvp {

    namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp";

```

```
/* Replace with IANA when assigned */
prefix "rsvp";

import ietf-interfaces {
  prefix "if";
}

import ietf-inet-types {
  prefix inet;
}

import ietf-yang-types {
  prefix "yang";
}

import ietf-routing {
  prefix "rt";
}

import ietf-key-chain {
  prefix "key-chain";
}

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>

  WG Chair:   Lou Berger
              <mailto:lberger@labn.net>

  WG Chair:   Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Tarek Saad
              <mailto:tsaad@cisco.com>

  Editor:     Rakesh Gandhi
              <mailto:rgandhi@cisco.com>

  Editor:     Himanshu Shah
              <mailto:hshah@ciena.com>
```

Editor: Xufeng Liu
<mailto:Xufeng_Liu@jabil.com>

Editor: Xia Chen
<mailto:jescia.chenxia@huawei.com>

Editor: Raqib Jones
<mailto:raqib@Brocade.com>

Editor: Bin Wen
<mailto:Bin_Wen@cable.comcast.com>;

```
description
  "This module contains the RSVP YANG data model.";

revision "2017-10-29" {
  description "Latest revision of RSVP yang module.";
  reference "RFC2205";
}

identity rsvp {
  base "rt:routing-protocol";
  description "RSVP protocol";
}

identity rsvp-session-type {
  description "Base RSVP session type";
}

identity rsvp-session-ipv4 {
  base rsvp-session-type;
  description "RSVP IPv4 session type";
}

identity rsvp-session-ipv6 {
  base rsvp-session-type;
  description "RSVP IPv4 session type";
}

identity reservation-style {
  description "Base identity for reservation style";
}

identity reservation-wildcard-filter {
  base reservation-style;
  description "Wildcard-Filter (WF) Style";
  reference "RFC2205";
}
```

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

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

grouping graceful-restart_config {
  description
    "Base configuration parameters relating to RSVP
    Graceful-Restart";
  leaf enabled {
    type boolean;
    description
      "'true' if RSVP Graceful Restart is enabled.
      'false' if RSVP Graceful Restart is disabled.";
  }
}

grouping graceful-restart {
  description
    "RSVP graceful restart parameters grouping";
  container graceful-restart {
    description
      "RSVP graceful restart parameters container";
    uses graceful-restart_config;
  }
}

grouping refresh-reduction_config {
  description
    "Configuration parameters relating to RSVP
    refresh reduction";

  leaf enabled {
    type boolean;
    description
      "'true' if RSVP Refresh Reduction is enabled.
      'false' if RSVP Refresh Reduction is disabled.";
  }
}

grouping refresh-reduction {
```

```
    description
      "Top level grouping for RSVP refresh reduction
      parameters";
    container refresh-reduction {
      description
        "Top level container for RSVP refresh reduction
        parameters";
      uses refresh-reduction_config;
    }
  }

  grouping authentication_config {
    description
      "Configuration parameters relating to RSVP
      authentication";
    leaf enabled {
      type boolean;
      description
        "'true' if RSVP Authentication is enabled.
        'false' if RSVP Authentication is disabled.";
    }
    leaf authentication-key {
      type string;
      description
        "An authentication key string";
      reference
        "RFC 2747: RSVP Cryptographic Authentication";
    }
    leaf crypto-algorithm {
      type identityref {
        base key-chain:crypto-algorithm;
      }
      mandatory true;
      description
        "Cryptographic algorithm associated with key.";
    }
  }

  grouping authentication {
    description
      "Top level grouping for RSVP authentication parameters";
    container authentication {
      description
        "Top level container for RSVP authentication
        parameters";
      uses authentication_config;
    }
  }
```

```
grouping hellos_config {
  description
    "Configuration parameters relating to RSVP
    hellos";
  leaf enabled {
    type boolean;
    description
      "'true' if RSVP Hello is enabled.
      'false' if RSVP Hello is disabled.";
  }
}

grouping hellos {
  description
    "Top level grouping for RSVP hellos parameters";
  container hellos {
    description
      "Top level container for RSVP hello parameters";
    uses hellos_config;
  }
}

grouping signaling-parameters_config {
  description
    "Configuration parameters relating to RSVP
    signaling";
}

grouping signaling-parameters {
  description
    "Top level grouping for RSVP signaling parameters";
  uses signaling-parameters_config;
}

grouping session-attributes_state {
  description
    "Top level grouping for RSVP session properties";
  leaf local-index {
    type uint64;
    description
      "The index used to identify the RSVP session
      on the local network element. This index is
      generated by the device and is unique only
      to the local network element.";
  }
  leaf destination-port {
    type inet:port-number;
    description "RSVP destination port";
  }
}
```

```
    reference "RFC2205";
  }
  leaf source {
    type inet:ip-address;
    description "RSVP source address";
    reference "RFC2205";
  }
  leaf destination {
    type inet:ip-address;
    description "RSVP destination address";
    reference "RFC2205";
  }
  leaf session-name {
    type string;
    description
      "The signaled name of this RSVP session.";
  }
  leaf session-state {
    type enumeration {
      enum "up" {
        description
          "RSVP session is up";
      }
      enum "down" {
        description
          "RSVP session is down";
      }
    }
    description
      "Enumeration of RSVP session states";
  }
  leaf session-type {
    type identityref {
      base rsvp-session-type;
    }
    description "RSVP session type";
  }
  container psbs {
    description "Path State Block container";
    list psb {
      description "List of path state blocks";
      leaf source-port {
        type inet:port-number;
        description "RSVP source port";
        reference "RFC2205";
      }
      leaf expires-in {
        type uint32;
      }
    }
  }
```

```
        units seconds;
        description "Time to reservation expiry (in seconds)";
    }
}
}
container rsbs {
    description "Reservation State Block container";
    list rsb {
        description "List of reservation state blocks";
        leaf source-port {
            type inet:port-number;
            description "RSVP source port";
            reference "RFC2205";
        }
        leaf reservation-style {
            type identityref {
                base reservation-style;
            }
            description "RSVP reservation style";
        }
        leaf expires-in {
            type uint32;
            units seconds;
            description "Time to reservation expiry (in seconds)";
        }
    }
}

grouping neighbor-attributes {
    description
        "Top level grouping for RSVP neighbor properties";
    leaf address {
        type inet:ip-address;
        description
            "Address of RSVP neighbor";
    }
    container state {
        config false;
        description
            "State information associated with RSVP
            neighbor properties";
        uses neighbor-derived_state;
    }
}

grouping packets_state {
    description
```



```
    "Packet statistics grouping";
  container packets {
    description
      "Packet statistics container";
    leaf sent {
      type yang:counter64;
      description
        "Packet sent count";
    }

    leaf received {
      type yang:counter64;
      description
        "Packet received count";
    }
  }
}

grouping protocol_state {
  description
    "RSVP protocol statistics grouping";
  container messages {
    description
      "RSVP protocol statistics container";
    leaf ack-sent {
      type yang:counter64;
      description
        "Hello sent count";
    }

    leaf ack-received {
      type yang:counter64;
      description
        "Hello received count";
    }

    leaf bundle-sent {
      type yang:counter64;
      description
        "Bundle sent count";
    }

    leaf bundle-received {
      type yang:counter64;
      description
        "Bundle received count";
    }
  }
}
```

```
leaf hello-sent {
  type yang:counter64;
  description
    "Hello sent count";
}

leaf hello-received {
  type yang:counter64;
  description
    "Hello received count";
}

leaf integrity-challenge-sent {
  type yang:counter64;
  description
    "Integrity Challenge sent count";
}

leaf integrity-challenge-received {
  type yang:counter64;
  description
    "Integrity Challenge received count";
}

leaf integrity-response-sent {
  type yang:counter64;
  description
    "Integrity Response sent count";
}

leaf integrity-response-received {
  type yang:counter64;
  description
    "Integrity Response received count";
}

leaf notify-sent {
  type yang:counter64;
  description
    "Notify sent count";
}

leaf notify-received {
  type yang:counter64;
  description
    "Notify received count";
}
```

```
leaf path-sent {
  type yang:counter64;
  description
    "Path sent count";
}

leaf path-received {
  type yang:counter64;
  description
    "Path received count";
}

leaf path-err-sent {
  type yang:counter64;
  description
    "Path error sent count";
}

leaf path-err-received {
  type yang:counter64;
  description
    "Path error received count";
}

leaf path-tear-sent {
  type yang:counter64;
  description
    "Path tear sent count";
}

leaf path-tear-received {
  type yang:counter64;
  description
    "Path tear received count";
}

leaf resv-sent {
  type yang:counter64;
  description
    "Resv sent count";
}

leaf resv-received {
  type yang:counter64;
  description
    "Resv received count";
}
```

```
leaf resv-confirm-sent {
  type yang:counter64;
  description
    "Confirm sent count";
}

leaf resv-confirm-received {
  type yang:counter64;
  description
    "Confirm received count";
}

leaf resv-err-sent {
  type yang:counter64;
  description
    "Resv error sent count";
}

leaf resv-err-received {
  type yang:counter64;
  description
    "Resv error received count";
}

leaf resv-tear-sent {
  type yang:counter64;
  description
    "Resv tear sent count";
}

leaf resv-tear-received {
  type yang:counter64;
  description
    "Resv tear received count";
}

leaf summary-refresh-sent {
  type yang:counter64;
  description
    "Summary refresh sent count";
}

leaf summary-refresh-received {
  type yang:counter64;
  description
    "Summary refresh received count";
}
```

```
    leaf unknown-messages-received {
      type yang:counter64;
      description
        "Unknown packet received count";
    }
  }
}

grouping errors_state {
  description
    "Error statistics state grouping";
  container errors {
    description
      "Error statistics state container";
    leaf authenticate {
      type yang:counter64;
      description
        "The total number of packets received with an
        authentication failure.";
    }

    leaf checksum {
      type yang:counter64;
      description
        "The total number of packets received with an invalid
        checksum value.";
    }

    leaf packet-len {
      type yang:counter64;
      description
        "The total number of packets received with an invalid
        packet length.";
    }
  }
}

grouping statistics_state {
  description "RSVP statistic attributes.";
  container statistics {
    description
      "statistics state container";
    container state {
      config false;
      description
        "State information associated with RSVP
        hello parameters";
      uses protocol_state;
    }
  }
}
```

```
        uses packets_state;
        uses errors_state;
    }
}

grouping neighbor-derived_state {
  description
    "Derived state at neighbor level.";

  leaf address {
    type inet:ip-address;
    description
      "Address of RSVP neighbor";
  }

  leaf epoch {
    type uint32;
    description
      "Neighbor epoch.";
  }

  leaf expiry-time {
    type uint32;
    units seconds;
    description
      "Neighbor expiry time after which the neighbor state
       is purged if no states associated with it";
  }

  container graceful-restart {
    description
      "Graceful restart information.";

    leaf enabled {
      type boolean;
      description
        "'true' if graceful restart is enabled for the
         neighbor.";
    }

    leaf local-restart-time {
      type uint32;
      units seconds;
      description
        "Local node restart time";
    }
  }
}
```

```
leaf local-recovery-time {
    type uint32;
    units seconds;
    description
        "Local node recover time";
}

leaf neighbor-restart-time {
    type uint32;
    units seconds;
    description
        "Neighbor restart time";
}

leaf neighbor-recovery-time {
    type uint32;
    units seconds;
    description
        "Neighbor recover time";
}

container helper-mode {
    description
        "Helper mode information ";

    leaf enabled {
        type boolean;
        description
            "'true' if helper mode is enabled.";
    }

    leaf max-helper-restart-time {
        type uint32;
        units seconds;
        description
            "The time the router or switch waits after it
            discovers that a neighboring router has gone down
            before it declares the neighbor down";
    }

    leaf max-helper-recovery-time {
        type uint32;
        units seconds;
        description
            "The amount of time the router retains the state of its
            RSVP neighbors while they undergo a graceful restart";
    }
}
```

```
    leaf neighbor-restart-time-remaining {
      type uint32;
      units seconds;
      description
        "Number of seconds remaining for neighbor to send
        Hello message after restart.";
    }

    leaf neighbor-recovery-time-remaining {
      type uint32;
      units seconds;
      description
        "Number of seconds remaining for neighbor to
        refresh.";
    }
  } // helper-mode
} // graceful-restart

leaf hello-status {
  type enumeration {
    enum "enabled" {
      description
        "Enabled";
    }
    enum "disabled" {
      description
        "Disabled";
    }
    enum "restarting" {
      description
        "Restarting";
    }
  }
  description
    "Hello status";
}

leaf interface {
  type if:interface-ref;
  description
    "Interface where RSVP neighbor was detected";
}

leaf neighbor-state {
  type enumeration {
    enum "up" {
      description
        "up";
    }
  }
}
```



```
    }
    enum "down" {
      description
        "down";
    }
    enum "hello-disable" {
      description
        "hello-disable";
    }
    enum "restarting" {
      description
        "restarting";
    }
  }
  description
    "Neighbor state";
}

leaf refresh-reduction-capable {
  type boolean;
  description
    "enables all RSVP refresh reduction message
    bundling, RSVP message ID, reliable message delivery
    and summary refresh";
  reference
    "RFC 2961 RSVP Refresh Overhead Reduction
    Extensions";
}

leaf restart-count {
  type yang:counter32;
  description
    "Number of times this neighbor restart";
}

leaf restart-time {
  type yang:date-and-time;
  description
    "Last restart time of the neighbor";
}
}

grouping global-attributes {
  description
    "Top level grouping for RSVP global properties";
  container sessions {
    description
      "RSVP sessions container";
  }
}
```

```
list session {
  key "local-index";
  config false;
  description
    "List of RSVP sessions";

  leaf local-index {
    type leafref {
      path "../state/local-index";
    }
    description
      "Reference to the local index for the RSVP
      session";
  }
  container state {
    config false;
    description
      "State information associated with RSVP
      session parameters";
    uses session-attributes_state;
  }
}
uses statistics_state;
}

grouping intf-attributes {
  description
    "Top level grouping for RSVP interface properties";
  uses signaling-parameters;
  uses refresh-reduction;
  uses hellos;
  uses authentication;
  uses statistics_state;
}

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol" {
  when "rt:type = 'rsvp:rsvp'" {
    description
      "This augment is only valid when routing protocol
      instance type is RSVP.";
  }
  description
    "RSVP protocol augmentation";
  container rsvp {
    presence "Enable RSVP feature";
    description "RSVP feature container";
  }
}
```

```
    container globals {
      description "RSVP global properties.";
      uses global-attributes;
      uses graceful-restart;
    }

    container interfaces {
      description
        "RSVP interfaces container";
      uses intf-attributes;

      list interface {
        key "interface";
        description
          "RSVP interfaces.";
        leaf interface {
          type if:interface-ref;
          description
            "RSVP interface.";
        }
        uses intf-attributes;
      }
    }

    container neighbors {
      description "RSVP neighbors container";
      list neighbor {
        key "address";
        description "List of RSVP neighbors";
        uses neighbor-attributes;
      }
    }
  }
}
<CODE ENDS>
```

3.2. RSVP Extended YANG Model

The RSVP extended YANG model covers optional or non-core RSVP feature(s). It also covers feature(s) that are not necessarily supported by all vendors, and hence, guarded with "if-feature" checks.

3.2.1. Tree Diagram

Figure 4 shows the YANG tree representation for configuration and state data that is augmenting the RSVP basic module:

```

module: ietf-rsvp-extended
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
    rsvp:rsvp/rsvp:globals/rsvp:graceful-restart:
      +--rw restart-time?      uint32
      +--rw recovery-time?    uint32
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
    rsvp:rsvp/rsvp:globals/rsvp:statistics/rsvp:state/rsvp:packets:
      +---ro discontinuity-time?  yang:date-and-time
      +---ro out-dropped?         yang:counter64
      +---ro in-dropped?         yang:counter64
      +---ro out-error?          yang:counter64
      +---ro in-error?          yang:counter64
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
    rsvp:rsvp/rsvp:globals/rsvp:statistics/rsvp:state/rsvp:messages:
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
    rsvp:rsvp/rsvp:globals/rsvp:statistics/rsvp:state/rsvp:errors:
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
    rsvp:rsvp/rsvp:interfaces:
      +--rw refresh-interval?      uint32
      +--rw refresh-misses?       uint32
      +--rw checksum?             boolean
      +--rw patherr-state-removal? empty
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
    rsvp:rsvp/rsvp:interfaces/rsvp:refresh-reduction:
      +--rw bundle-message-max-size?  uint32
      +--rw reliable-ack-hold-time?   uint32
      +--rw reliable-ack-max-size?   uint32
      +--rw reliable-retransmit-time? uint32
      +--rw reliable-srefresh?       empty
      +--rw summary-max-size?        uint32
  augment
    /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
    rsvp:rsvp/rsvp:interfaces/rsvp:hellos:
      +--rw interface-based?  empty
      +--rw hello-interval?   uint32
      +--rw hello-misses?     uint32
  augment

```

```

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:interfaces/rsvp:authentication:
  +--rw lifetime?      uint32
  +--rw window-size?   uint32
  +--rw challenge?     empty
  +--rw retransmits?   uint32
  +--rw key-chain?     key-chain:key-chain-ref
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:interfaces/rsvp:interface:
  +--rw refresh-interval?   uint32
  +--rw refresh-misses?    uint32
  +--rw checksum?          boolean
  +--rw patherr-state-removal? empty
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:interfaces/rsvp:interface/rsvp:refresh-reduction:
  +--rw bundle-message-max-size?   uint32
  +--rw reliable-ack-hold-time?     uint32
  +--rw reliable-ack-max-size?     uint32
  +--rw reliable-retransmit-time?   uint32
  +--rw reliable-srefresh?         empty
  +--rw summary-max-size?          uint32
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:interfaces/rsvp:interface/rsvp:hellos:
  +--rw interface-based?   empty
  +--rw hello-interval?    uint32
  +--rw hello-misses?      uint32
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:interfaces/rsvp:interface/rsvp:authentication:
  +--rw lifetime?      uint32
  +--rw window-size?   uint32
  +--rw challenge?     empty
  +--rw retransmits?   uint32
  +--rw key-chain?     key-chain:key-chain-ref

```

Figure 4: RSVP extended model tree diagram

3.2.2. YANG Module

Figure 5 shows the RSVP extended YANG module:

```

<CODE BEGINS> file "ietf-rsvp-extended@2017-10-29.yang"
module ietf-rsvp-extended {

  namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp-extended";

```

```
prefix "rsvp-ext";

import ietf-rsvp {
  prefix "rsvp";
}

import ietf-routing {
  prefix "rt";
}

import ietf-yang-types {
  prefix "yang";
}

import ietf-key-chain {
  prefix "key-chain";
}

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>

  WG Chair:   Lou Berger
              <mailto:lberger@labn.net>

  WG Chair:   Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Tarek Saad
              <mailto:tsaad@cisco.com>

  Editor:     Rakesh Gandhi
              <mailto:rgandhi@cisco.com>

  Editor:     Himanshu Shah
              <mailto:hshah@ciena.com>

  Editor:     Xufeng Liu
              <mailto:Xufeng_Liu@jabil.com>
```

Editor: Xia Chen
<mailto:jescia.chenxia@huawei.com>

Editor: Raqib Jones
<mailto:raqib@Brocade.com>

Editor: Bin Wen
<mailto:Bin_Wen@cable.comcast.com>;

```
description
  "This module contains the Extended RSVP YANG data model.";

revision "2017-10-29" {
  description "Latest revision of RSVP extended yang module.";
  reference "RFC2205";
}

/* RSVP features */
feature authentication {
  description
    "Indicates support for RSVP authentication";
}

feature error-statistics {
  description
    "Indicates support for error statistics";
}

feature global-statistics {
  description
    "Indicates support for global statistics";
}

feature graceful-restart {
  description
    "Indicates support for RSVP graceful restart";
}

feature hellos {
  description
    "Indicates support for RSVP hellos (RFC3209).";
}

feature notify {
  description
    "Indicates support for RSVP notify message (RFC3473).";
}
```

```
feature refresh-reduction {
  description
    "Indicates support for RSVP refresh reduction
    (RFC2961).";
}

feature refresh-reduction-extended {
  description
    "Indicates support for RSVP refresh reduction
    (RFC2961).";
}

feature per-interface-statistics {
  description
    "Indicates support for per interface statistics";
}

grouping graceful-restart-extended_config {
  description
    "Configuration parameters relating to RSVP
    Graceful-Restart";
  leaf restart-time {
    type uint32;
    units seconds;
    description
      "Graceful restart time (seconds).";
    reference
      "RFC 5495: Description of the Resource
      Reservation Protocol - Traffic-Engineered
      (RSVP-TE) Graceful Restart Procedures";
  }
  leaf recovery-time {
    type uint32;
    units seconds;
    description
      "RSVP state recovery time";
  }
}

grouping authentication-extended_config {
  description
    "Configuration parameters relating to RSVP
    authentication";
  leaf lifetime {
    type uint32 {
      range "30..86400";
    }
    units seconds;
  }
}
```



```
    description
      "Life time for each security association";
    reference
      "RFC 2747: RSVP Cryptographic
      Authentication";
  }
  leaf window-size {
    type uint32 {
      range "1..64";
    }
    description
      "Window-size to limit number of out-of-order
      messages.";
    reference
      "RFC 2747: RSVP Cryptographic
      Authentication";
  }
  leaf challenge {
    type empty;
    description
      "Enable challenge messages.";
    reference
      "RFC 2747: RSVP Cryptographic
      Authentication";
  }
  leaf retransmits {
    type uint32 {
      range "1..10000";
    }
    description
      "Number of retransmits when messages are
      dropped.";
    reference
      "RFC 2747: RSVP Cryptographic
      Authentication";
  }
  leaf key-chain {
    type key-chain:key-chain-ref;
    description
      "Key chain name to authenticate RSVP
      signaling messages.";
    reference
      "RFC 2747: RSVP Cryptographic
      Authentication";
  }
}

grouping hellos-extended_config {
```

```
description
  "Configuration parameters relating to RSVP
  hellos";
leaf interface-based {
  type empty;
  description
    "Enable interface-based Hello adjacency if present.";
}
leaf hello-interval {
  type uint32;
  units milliseconds;
  description
    "Configure interval between successive Hello
    messages in milliseconds.";
  reference
    "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels.
    RFC 5495: Description of the Resource
    Reservation Protocol - Traffic-Engineered
    (RSVP-TE) Graceful Restart Procedures";
}
leaf hello-misses {
  type uint32 {
    range "1..10";
  }
  description
    "Configure max number of consecutive missed
    Hello messages.";
  reference
    "RFC 3209: RSVP-TE: Extensions to RSVP for
    LSP Tunnels RFC 5495: Description of the
    Resource Reservation Protocol - Traffic-
    Engineered (RSVP-TE) Graceful Restart
    Procedures";
}
}

grouping signaling-parameters-extended_config {
  description
    "Configuration parameters relating to RSVP
    signaling";
  leaf refresh-interval {
    type uint32;
    description
      "Set interval between successive refreshes";
  }
  leaf refresh-misses {
    type uint32;
    description
```

```
        "Set max number of consecutive missed
        messages for state expiry";
    }
    leaf checksum {
        type boolean;
        description
            "Enable RSVP message checksum computation";
    }
    leaf patherr-state-removal {
        type empty;
        description
            "State-Removal flag in Path Error message
            if present.";
    }
}

grouping refresh-reduction-extended_config {
    description
        "Configuration parameters relating to RSVP
        refresh reduction";

    leaf bundle-message-max-size {
        type uint32 {
            range "512..65000";
        }
        description
            "Configure maximum size (bytes) of a
            single RSVP Bundle message.";
    }
    leaf reliable-ack-hold-time {
        type uint32;
        units milliseconds;
        description
            "Configure hold time in milliseconds for
            sending RSVP ACK message(s).";
    }
    leaf reliable-ack-max-size {
        type uint32;
        description
            "Configure max size of a single RSVP ACK
            message.";
    }
    leaf reliable-retransmit-time {
        type uint32;
        units milliseconds;
        description
            "Configure min delay in milliseconds to
            wait for an ACK before a retransmit.";
    }
}
```

```
    }
    leaf reliable-srefresh {
      type empty;
      description
        "Configure use of reliable messaging for
        summary refresh if present.";
    }
    leaf summary-max-size {
      type uint32 {
        range "20..65000";
      }
      description
        "Configure max size (bytes) of a single
        RSVP summary refresh message.";
    }
  }
}

grouping packets-extended_state {
  description
    "Packet statistics.";
  leaf discontinuity-time {
    type yang:date-and-time;
    description
      "The time on the most recent occasion at which any one
      or more of the statistic counters suffered a
      discontinuity. If no such discontinuities have occurred
      since the last re-initialization of the local
      management subsystem, then this node contains the time
      the local management subsystem re-initialized itself.";
  }
  leaf out-dropped {
    type yang:counter64;
    description
      "Out packet drop count";
  }
  leaf in-dropped {
    type yang:counter64;
    description
      "In packet drop count";
  }
  leaf out-error {
    type yang:counter64;
    description
      "Out packet error count";
  }
}
```

```
    leaf in-error {
      type yang:counter64;
      description
        "In packet rx error count";
    }
  }

  grouping protocol-extended_state {
    description
      "RSVP protocol statistics.";
  }

  grouping errors-extended_state {
    description
      "Error statistics.";
  }

  grouping extended_state {
    description "RSVP statistic attributes.";
    uses packets-extended_state;
    uses protocol-extended_state;
    uses errors-extended_state;
  }

  /**
   * RSVP extensions augmentations
   */

  /* RSVP globals graceful restart*/
  augment "/rt:routing/rt:control-plane-protocols/" +
    "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
    "rsvp:graceful-restart" {
    description
      "RSVP globals configuration extensions";
    uses graceful-restart-extended_config;
  }

  /* RSVP statistics augmentation */
  augment "/rt:routing/rt:control-plane-protocols/" +
    "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
    "rsvp:statistics/rsvp:state/rsvp:packets" {
    description
      "RSVP packet stats extensions";
    uses packets-extended_state;
  }
  augment "/rt:routing/rt:control-plane-protocols/" +
    "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
    "rsvp:statistics/rsvp:state/rsvp:messages" {
```

```
    description
      "RSVP protocol message stats extensions";
    uses protocol-extended_state;
  }
  augment "/rt:routing/rt:control-plane-protocols/" +
    "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/" +
    "rsvp:statistics/rsvp:state/rsvp:errors" {
    description
      "RSVP errors stats extensions";
    uses errors-extended_state;
  }

  /**
   * RSVP all interfaces extensions
   */

  /* RSVP interface signaling extensions */
  augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces" {
    description
      "RSVP signaling all interfaces configuration extensions";
    uses signaling-parameters-extended_config;
  }

  /* RSVP refresh reduction extension */
  augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
    + "rsvp:refresh-reduction" {
    description
      "RSVP refresh-reduction all interface configuration
      extensions";
    uses refresh-reduction-extended_config;
  }

  /* RSVP hellos extension */
  augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
    + "rsvp:hellos" {
    description
      "RSVP hello all interfaces configuration extensions";
    uses hellos-extended_config;
  }

  /* RSVP authentication extension */
  augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/"
    + "rsvp:authentication" {
    description
```

```
    "RSVP authentication all interfaces configuration extensions";
    uses authentication-extended_config;
}

/**
 * RSVP interface extensions
 */

/* RSVP interface signaling extensions */
augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
    "rsvp:interface" {
    description
        "RSVP signaling interface configuration extensions";
    uses signaling-parameters-extended_config;
}

/* RSVP refresh reduction extension */
augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
    "rsvp:interface/rsvp:refresh-reduction" {
    description
        "RSVP refresh-reduction interface configuration extensions";
    uses refresh-reduction-extended_config;
}

/* RSVP hellos extension */
augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
    "rsvp:interface/rsvp:hellos" {
    description
        "RSVP hello interface configuration extensions";
    uses hellos-extended_config;
}

/* RSVP authentication extension */
augment "/rt:routing/rt:control-plane-protocols/"
    + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
    "rsvp:interface/rsvp:authentication" {
    description
        "RSVP authentication interface configuration extensions";
    uses authentication-extended_config;
}
}
<CODE ENDS>
```

Figure 5: RSVP extended YANG module

4. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp-extended XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-rsvp namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp
prefix: ietf-rsvp reference: RFC3209

name: ietf-rsvp-extended namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp-extended
prefix: ietf-rsvp-extended reference: RFC3209

5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF

users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which 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.

6. Acknowledgement

The authors would like to thank Lou Berger, for reviewing and providing valuable feedback on this document.

7. Contributors

Xia Chen
Huawei Technologies

Email: jescia.chenxia@huawei.com

Raqib Jones
Brocade

Email: raqib@Brocade.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com

8. References

8.1. Normative References

- [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>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, DOI 10.17487/RFC2747, January 2000, <<https://www.rfc-editor.org/info/rfc2747>>.
- [RFC2961] Berger, L., Gan, D., Swallow, G., Pan, P., Tommasi, F., and S. Molendini, "RSVP Refresh Overhead Reduction Extensions", RFC 2961, DOI 10.17487/RFC2961, April 2001, <<https://www.rfc-editor.org/info/rfc2961>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.

- [RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <<https://www.rfc-editor.org/info/rfc3473>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC5063] Satyanarayana, A., Ed. and R. Rahman, Ed., "Extensions to GMPLS Resource Reservation Protocol (RSVP) Graceful Restart", RFC 5063, DOI 10.17487/RFC5063, October 2007, <<https://www.rfc-editor.org/info/rfc5063>>.
- [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>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <<https://www.rfc-editor.org/info/rfc8022>>.

8.2. Informative References

[I-D.dsdt-nmda-guidelines]

Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K.,
and R. Wilton, "Guidelines for YANG Module Authors
(NMDA)", draft-dsdt-nmda-guidelines-01 (work in progress),
May 2017.

Authors' Addresses

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Tarek Saad (editor)
Cisco Systems, Inc.

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.

Email: rgandhi@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

Himanshu Shah
Ciena

Email: hshah@ciena.com

TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2018

V. Beeram
Juniper Networks
T. Saad, Ed.
R. Gandhi
Cisco Systems, Inc.
X. Liu
Jabil
I. Bryskin
Huawei Technologies
H. Shah
Ciena
February 19, 2018

A YANG Data Model for RSVP-TE
draft-ietf-teas-yang-rsvp-te-03

Abstract

This document defines a YANG data model for the configuration and management of RSVP (Resource Reservation Protocol) to establish Traffic-Engineered (TE) Label-Switched Paths (LSPs) for MPLS (Multi-Protocol Label Switching) and other technologies.

The model defines a generic RSVP-TE module for signaling LSPs that is technology agnostic. The generic RSVP-TE module is to be augmented by technology specific RSVP-TE modules that define technology specific data. This document defines the augmentation for RSVP-TE MPLS LSPs model.

This model covers data for the configuration, operational state, remote procedural calls, and event notifications.

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 August 23, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
1.2. Tree Diagram	3
1.3. Prefixes in Data Node Names	4
2. Design Considerations	5
2.1. Module Hierarchy	5
2.2. RSVP-TE Generic Model	6
2.2.1. Tree Diagram	6
2.2.2. YANG Module	14
2.3. RSVP-TE MPLS Model	26
2.3.1. Tree Diagram	26
2.3.2. YANG Module	28
3. IANA Considerations	41
4. Security Considerations	41
5. Acknowledgement	42
6. Contributors	42
7. Normative References	42
Authors' Addresses	43

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a generic YANG data model for configuring and managing RSVP-TE LSP(s) [RFC3209]. The RSVP-TE generic model augments the RSVP base and extended models defined in [I-D.ietf-teas-yang-rsvp], and adds TE extensions to the RSVP protocol [RFC2205] model configuration and state data. The technology specific RSVP-TE models augment the generic RSVP-TE model with additional technology specific parameters. For example, this document also defines the MPLS RSVP-TE model for configuring and managing MPLS RSVP TE LSP(s).

In addition to augmenting the RSVP YANG module, the modules defined in this document augment the TE Interfaces, Tunnels and LSP(s) YANG module defined in [I-D.ietf-teas-yang-te] to define additional parameters to enable signaling for RSVP-TE.

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

1.2. Tree Diagram

A simplified graphical representation of the data model is presented in each section of the model. The following notations are used for the YANG model data tree representation.

<status> <flags> <name> <opts> <type>

<status> is one of:

- + for current
- x for deprecated
- o for obsolete

<flags> is one of:

- rw for read-write configuration data
- ro for read-only non-configuration data
- x for execution rpcs
- n for notifications

<name> is the name of the node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>

<opts> is one of:

- ? for an optional leaf or node
- ! for a presence container
- * for a leaf-list or list

Brackets [<keys>] for a list's keys

Curly braces {<condition>} for optional feature that make node conditional

Colon : for marking case nodes

Ellipses ("...") subtree contents not shown

Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

<type> is the name of the type for leafs and leaf-lists.

1.3. 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
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
te	ietf-te	this document
te-types	ietf-te-types	this document
te-mpls-types	ietf-te-mpls-types	this document
te-dev	ietf-te-device	this document
te-mpls	ietf-te-mpls	this document
te-sr-mpls	ietf-te-sr-mpls	this document

Table 1: Prefixes and corresponding YANG modules

2. Design Considerations

2.1. Module Hierarchy

The data pertaining to RSVP-TE in this document is divided into two modules: a technology agnostic RSVP-TE module that holds generic parameters for RSVP-TE applicable to all technologies, and a technology specific RSVP-TE module (e.g. for MPLS RSVP-TE) that holds parameters specific to the technology.

This document defines YANG data models for RSVP-TE, and RSVP-TE MPLS configuration, state, notification and RPCs. The relationship between the different modules is depicted in Figure 1.

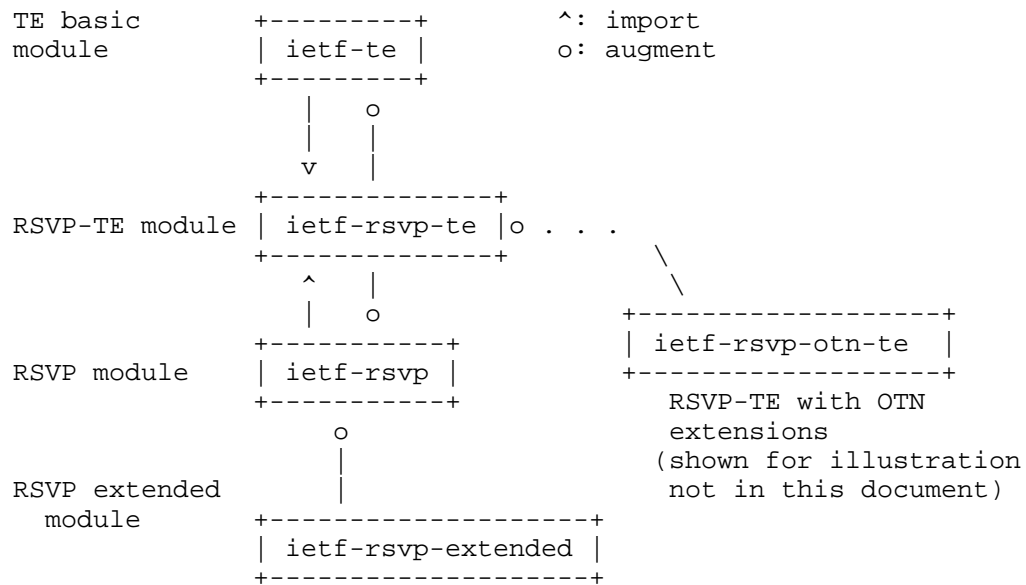


Figure 1: Relationship of RSVP and RSVP-TE modules with other protocol modules

2.2. RSVP-TE Generic Model

The RSVP-TE generic module augments the RSVP base and extended YANG modules defined in [I-D.ietf-teas-yang-rsvp] as well as the TE tunnels and interfaces module [I-D.ietf-teas-yang-te] to cover parameters specific to the configuration and management of RSVP-TE interfaces, tunnels and LSP(s).

2.2.1. Tree Diagram

There are three types of configuration and state data nodes in this module:

- o those augmenting or extending the base RSVP module
- o those augmenting or extending the base TE module
- o those that are specific to the RSVP-TE module

Below is a YANG tree representation for data items defined in the RSVP-TE generic module:

```

module: ietf-rsvp-te
augment

```

```

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:globals:
  +--rw global-soft-preemption!
    +--rw soft-preemption-timeout?   uint16
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:interfaces:
  +--rw rsvp-te-interface-attributes
    +--ro state
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:interfaces/rsvp:interface:
  +--rw rsvp-te-interface-attributes
    +--ro state
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:globals/rsvp:sessions/rsvp:session/rsvp:state/
rsvp:psbs/rsvp:psb:
  +--ro tspec-average-rate?   rt-types:bandwidth-ieee-float32
  +--ro tspec-size?           rt-types:bandwidth-ieee-float32
  +--ro tspec-peak-rate?      rt-types:bandwidth-ieee-float32
  +--ro min-policed-unit?     uint32
  +--ro max-packet-size?      uint32
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:globals/rsvp:sessions/rsvp:session/rsvp:state/
rsvp:rsbs/rsvp:rsb:
  +--ro fspec-average-rate?   rt-types:bandwidth-ieee-float32
  +--ro fspec-size?           rt-types:bandwidth-ieee-float32
  +--ro fspec-peak-rate?      rt-types:bandwidth-ieee-float32
  +--ro min-policed-unit?     uint32
  +--ro max-packet-size?      uint32
augment
/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol/
rsvp:rsvp/rsvp:neighbors:
augment /te:te/te:tunnels/te:tunnel:
  +--rw lsp-signaled-name?     string
  +--rw local-recording-desired? boolean
  +--rw se-style-desired?      boolean
  +--rw path-reevaluation-request? boolean
  +--rw soft-preemption-desired? boolean
  +--rw lsp-rerouting?         enumeration
  +--rw lsp-integrity-required? boolean
  +--rw lsp-contiguous?        boolean
  +--rw lsp-stitching-desired? boolean
  +--rw lsp-preplanned?        boolean
  +--rw lsp-oob-mapping?       boolean
  +--rw retry-timer?           uint16

```

```

augment /te:te/te:tunnels/te:tunnel/te:state:
  +--ro lsp-signaled-name?      string
  +--ro local-recording-desired? boolean
  +--ro se-style-desired?       boolean
  +--ro path-reevaluation-request? boolean
  +--ro soft-preemption-desired? boolean
  +--ro lsp-rerouting?          enumeration
  +--ro lsp-integrity-required? boolean
  +--ro lsp-contiguous?         boolean
  +--ro lsp-stitching-desired?  boolean
  +--ro lsp-preplanned?         boolean
  +--ro lsp-oob-mapping?        boolean
  +--ro retry-timer?            uint16
augment /te:te/te:lsps-state/te:lsp:
  +--ro associated-rsvp-session? ->
    /rt:routing/control-plane-protocols/control-plane-protocol/
    rsvp:rsvp/globals/sessions/session/local-index
  +--ro lsp-signaled-name?      string
  +--ro local-recording-desired? boolean
  +--ro se-style-desired?       boolean
  +--ro path-reevaluation-request? boolean
  +--ro soft-preemption-desired? boolean
  +--ro lsp-rerouting?          enumeration
  +--ro lsp-integrity-required? boolean
  +--ro lsp-contiguous?         boolean
  +--ro lsp-stitching-desired?  boolean
  +--ro lsp-preplanned?         boolean
  +--ro lsp-oob-mapping?        boolean
  +--ro explicit-route-objects
    | +--ro incoming-explicit-route-hop* [index]
    | | +--ro index -> ../state/index
    | | +--ro state
    | | | +--ro index?          uint32
    | | | +--ro (type)?
    | | | | +--:(numbered)
    | | | | | +--ro numbered-hop
    | | | | | | +--ro address?    te-types:te-tp-id
    | | | | | | +--ro hop-type?   te-hop-type
    | | | | | | +--ro direction?  te-link-direction
    | | | | +--:(as-number)
    | | | | | +--ro as-number-hop
    | | | | | | +--ro as-number?  binary
    | | | | | | +--ro hop-type?   te-hop-type
    | | | | +--:(unnumbered)
    | | | | | +--ro unnumbered-hop
    | | | | | | +--ro node-id?    te-types:te-node-id
    | | | | | | +--ro link-tp-id? te-types:te-tp-id
    | | | | | | +--ro hop-type?   te-hop-type

```

```

|         |--ro direction?      te-link-direction
|         +--:(label)
|         |         |--ro label-hop
|         |         |         |--ro te-label
|         |         |         |         |--ro (technology)?
|         |         |         |         |         +--:(generic)
|         |         |         |         |         |--ro generic?
rt-types:generalized-label
|         |         |--ro direction?      te-label-direction
+--ro outgoing-explicit-route-hop* [index]
+--ro index      -> ../state/index
+--ro state
+--ro index?          uint32
+--ro (type)?
+--:(numbered)
|         |--ro numbered-hop
|         |         |--ro address?      te-types:te-tp-id
|         |         |--ro hop-type?     te-hop-type
|         |         |--ro direction?    te-link-direction
+--:(as-number)
|         |--ro as-number-hop
|         |         |--ro as-number?    binary
|         |         |--ro hop-type?     te-hop-type
+--:(unnumbered)
|         |--ro unnumbered-hop
|         |         |--ro node-id?      te-types:te-node-id
|         |         |--ro link-tp-id?   te-types:te-tp-id
|         |         |--ro hop-type?     te-hop-type
|         |         |--ro direction?    te-link-direction
+--:(label)
+--ro label-hop
+--ro te-label
+--ro (technology)?
|         +--:(generic)
|         |--ro generic?
rt-types:generalized-label
|         |--ro direction?      te-label-direction
+--ro incoming-record-route-subobjects
+--ro incoming-record-route-subobject* [index]
+--ro index      -> ../state/index
+--ro state
+--ro index?          uint32
+--ro (type)?
+--:(numbered)
|         |--ro address?      te-types:te-tp-id
|         |--ro ip-flags?     binary
+--:(unnumbered)
|         |--ro node-id?      te-types:te-node-id

```

```

|         |  +--ro link-tp-id?      te-types:te-tp-id
|         |  +---:(label)
|         |    +--ro value?          rt-types:generalized-label
|         |    +--ro label-flags?   binary
+--ro outgoing-record-route-subobjects
  +--ro outgoing-record-route-subobject* [index]
    +--ro index      -> ../state/index
    +--ro state
      +--ro index?      uint32
      +--ro (type)?
        +---:(numbered)
          |  +--ro address?          te-types:te-tp-id
          |  +--ro ip-flags?        binary
        +---:(unnumbered)
          |  +--ro node-id?          te-types:te-node-id
          |  +--ro link-tp-id?      te-types:te-tp-id
          +---:(label)
            +--ro value?          rt-types:generalized-label
            +--ro label-flags?   binary

augment
/te:te/te:tunnels/te:tunnel/te:p2p-primary-paths/te:p2p-primary-path/
te:state/te:lsp/te:lsp:
  +--ro associated-rsvp-session?      ->
    /rt:routing/control-plane-protocols/control-plane-protocol/
    rsvp:rsvp/globals/sessions/session/local-index
  +--ro lsp-signaled-name?            string
  +--ro local-recording-desired?      boolean
  +--ro se-style-desired?             boolean
  +--ro path-reevaluation-request?    boolean
  +--ro soft-preemption-desired?      boolean
  +--ro lsp-rerouting?                enumeration
  +--ro lsp-integrity-required?       boolean
  +--ro lsp-contiguous?              boolean
  +--ro lsp-stitching-desired?       boolean
  +--ro lsp-preplanned?              boolean
  +--ro lsp-oob-mapping?              boolean
  +--ro explicit-route-objects
    |  +--ro incoming-explicit-route-hop* [index]
    |  |  +--ro index      -> ../state/index
    |  |  +--ro state
    |  |    +--ro index?      uint32
    |  |    +--ro (type)?
    |  |      +---:(numbered)
    |  |      |  +--ro numbered-hop
    |  |      |  |  +--ro address?      te-types:te-tp-id
    |  |      |  |  +--ro hop-type?    te-hop-type
    |  |      |  |  +--ro direction?  te-link-direction
    |  |      +---:(as-number)

```

```

|
|
|   +--ro as-number-hop
|   |   +--ro as-number?   binary
|   |   +--ro hop-type?    te-hop-type
|   +--:(unnumbered)
|   |   +--ro unnumbered-hop
|   |   |   +--ro node-id?      te-types:te-node-id
|   |   |   +--ro link-tp-id?   te-types:te-tp-id
|   |   |   +--ro hop-type?     te-hop-type
|   |   |   +--ro direction?    te-link-direction
|   +--:(label)
|   |   +--ro label-hop
|   |   |   +--ro te-label
|   |   |   |   +--ro (technology)?
|   |   |   |   |   +--:(generic)
|   |   |   |   |   +--ro generic?
|   +--ro generalized-label
|   |   +--ro direction?    te-label-direction
|   +--ro outgoing-explicit-route-hop* [index]
|   |   +--ro index        -> ../state/index
|   |   +--ro state
|   |   |   +--ro index?      uint32
|   |   |   +--ro (type)?
|   |   |   +--:(numbered)
|   |   |   |   +--ro numbered-hop
|   |   |   |   |   +--ro address?      te-types:te-tp-id
|   |   |   |   |   +--ro hop-type?     te-hop-type
|   |   |   |   |   +--ro direction?    te-link-direction
|   |   |   +--:(as-number)
|   |   |   |   +--ro as-number-hop
|   |   |   |   |   +--ro as-number?    binary
|   |   |   |   |   +--ro hop-type?     te-hop-type
|   |   |   +--:(unnumbered)
|   |   |   |   +--ro unnumbered-hop
|   |   |   |   |   +--ro node-id?      te-types:te-node-id
|   |   |   |   |   +--ro link-tp-id?   te-types:te-tp-id
|   |   |   |   |   +--ro hop-type?     te-hop-type
|   |   |   |   |   +--ro direction?    te-link-direction
|   |   |   +--:(label)
|   |   |   |   +--ro label-hop
|   |   |   |   |   +--ro te-label
|   |   |   |   |   |   +--ro (technology)?
|   |   |   |   |   |   |   +--:(generic)
|   |   |   |   |   |   |   +--ro generic?
|   +--ro generalized-label
|   |   +--ro direction?    te-label-direction
|   +--ro incoming-record-route-subobjects
|   |   +--ro incoming-record-route-subobject* [index]
|   |   |   +--ro index        -> ../state/index

```

```

    +--ro state
      +--ro index?          uint32
      +--ro (type)?
        +--:(numbered)
          | +--ro address?      te-types:te-tp-id
          | +--ro ip-flags?     binary
        +--:(unnumbered)
          | +--ro node-id?      te-types:te-node-id
          | +--ro link-tp-id?   te-types:te-tp-id
        +--:(label)
          +--ro value?         rt-types:generalized-label
          +--ro label-flags?   binary
+--ro outgoing-record-route-subobjects
  +--ro outgoing-record-route-subobject* [index]
    +--ro index      -> ../state/index
    +--ro state
      +--ro index?          uint32
      +--ro (type)?
        +--:(numbered)
          | +--ro address?      te-types:te-tp-id
          | +--ro ip-flags?     binary
        +--:(unnumbered)
          | +--ro node-id?      te-types:te-node-id
          | +--ro link-tp-id?   te-types:te-tp-id
        +--:(label)
          +--ro value?         rt-types:generalized-label
          +--ro label-flags?   binary
augment
/te:te/te:tunnels/te:tunnel/te:p2p-secondary-paths/
te:p2p-secondary-path/te:state/te:lsps/te:lsp:
  +--ro associated-rsvp-session?      ->
  /rt:routing/control-plane-protocols/control-plane-protocol/
  rsvp:rsvp/globals/sessions/session/local-index
  +--ro lsp-signaled-name?            string
  +--ro local-recording-desired?      boolean
  +--ro se-style-desired?             boolean
  +--ro path-reevaluation-request?    boolean
  +--ro soft-preemption-desired?      boolean
  +--ro lsp-rerouting?                enumeration
  +--ro lsp-integrity-required?       boolean
  +--ro lsp-contiguous?               boolean
  +--ro lsp-stitching-desired?        boolean
  +--ro lsp-preplanned?               boolean
  +--ro lsp-oob-mapping?               boolean
  +--ro explicit-route-objects
    | +--ro incoming-explicit-route-hop* [index]
    | | +--ro index      -> ../state/index
    | | +--ro state

```

```

+--ro index?                uint32
+--ro (type)?
  +--:(numbered)
    +--ro numbered-hop
      +--ro address?        te-types:te-tp-id
      +--ro hop-type?       te-hop-type
      +--ro direction?     te-link-direction
    +--:(as-number)
      +--ro as-number-hop
        +--ro as-number?    binary
        +--ro hop-type?     te-hop-type
    +--:(unnumbered)
      +--ro unnumbered-hop
        +--ro node-id?      te-types:te-node-id
        +--ro link-tp-id?   te-types:te-tp-id
        +--ro hop-type?     te-hop-type
        +--ro direction?    te-link-direction
    +--:(label)
      +--ro label-hop
      +--ro te-label
        +--ro (technology)?
          +--:(generic)
            +--ro generic?
rt-types:generalized-label
  +--ro outgoing-explicit-route-hop* [index]
  +--ro index    -> ../state/index
  +--ro state
    +--ro index?                uint32
    +--ro (type)?
      +--:(numbered)
        +--ro numbered-hop
          +--ro address?        te-types:te-tp-id
          +--ro hop-type?       te-hop-type
          +--ro direction?     te-link-direction
        +--:(as-number)
          +--ro as-number-hop
            +--ro as-number?    binary
            +--ro hop-type?     te-hop-type
        +--:(unnumbered)
          +--ro unnumbered-hop
            +--ro node-id?      te-types:te-node-id
            +--ro link-tp-id?   te-types:te-tp-id
            +--ro hop-type?     te-hop-type
            +--ro direction?    te-link-direction
        +--:(label)
          +--ro label-hop
          +--ro te-label

```



```

|                                     +--ro (technology)?
|                                     |   +--:(generic)
|                                     |   +--ro generic?
rt-types:generalized-label
|                                     +--ro direction?   te-label-direction
+--ro incoming-record-route-subobjects
|   +--ro incoming-record-route-subobject* [index]
|   |   +--ro index      -> ../state/index
|   |   +--ro state
|   |   |   +--ro index?          uint32
|   |   |   +--ro (type)?
|   |   |   |   +--:(numbered)
|   |   |   |   |   +--ro address?      te-types:te-tp-id
|   |   |   |   |   +--ro ip-flags?     binary
|   |   |   |   +--:(unnumbered)
|   |   |   |   |   +--ro node-id?      te-types:te-node-id
|   |   |   |   |   +--ro link-tp-id?   te-types:te-tp-id
|   |   |   |   +--:(label)
|   |   |   |   |   +--ro value?        rt-types:generalized-label
|   |   |   |   |   +--ro label-flags?  binary
+--ro outgoing-record-route-subobjects
|   +--ro outgoing-record-route-subobject* [index]
|   |   +--ro index      -> ../state/index
|   |   +--ro state
|   |   |   +--ro index?          uint32
|   |   |   +--ro (type)?
|   |   |   |   +--:(numbered)
|   |   |   |   |   +--ro address?      te-types:te-tp-id
|   |   |   |   |   +--ro ip-flags?     binary
|   |   |   |   +--:(unnumbered)
|   |   |   |   |   +--ro node-id?      te-types:te-node-id
|   |   |   |   |   +--ro link-tp-id?   te-types:te-tp-id
|   |   |   |   +--:(label)
|   |   |   |   |   +--ro value?        rt-types:generalized-label
|   |   |   |   |   +--ro label-flags?  binary
augment /te:te/te-dev:interfaces/te-dev:interface:

```

Figure 2: RSVP-TE model Tree diagram

2.2.2. YANG Module

```

<CODE BEGINS> file "ietf-rsvp-te@2018-02-19.yang"
module ietf-rsvp-te {

  namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp-te";

  prefix "rsvp-te";

```

```
import ietf-rsvp {
  prefix rsvp;
}

import ietf-routing {
  prefix "rt";
}

import ietf-routing-types {
  prefix rt-types;
}

import ietf-te {
  prefix te;
}

import ietf-te-device {
  prefix te-dev;
}

/* Import TE generic types */
import ietf-te-types {
  prefix te-types;
}

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>

  WG Chair:   Lou Berger
              <mailto:lberger@labn.net>

  WG Chair:   Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Tarek Saad
              <mailto:tsaad@cisco.com>

  Editor:     Rakesh Gandhi
              <mailto:rgandhi@cisco.com>
```

Editor: Himanshu Shah
<mailto:hshah@ciena.com>

Editor: Xufeng Liu
<mailto:xliu@kuatrotech.com>

Editor: Xia Chen
<mailto:jescia.chenxia@huawei.com>

Editor: Raqib Jones
<mailto:raqib@Brocade.com>

Editor: Bin Wen
<mailto:Bin_Wen@cable.comcast.com>;

description

"This module contains the RSVP-TE YANG generic data model.";

```
revision "2018-02-19" {  
  description "Latest revision to RSVP-TE generic YANG module";  
  reference "RFC2205, RFC3209, etc.";  
}
```

```
/**  
 * RSVP-TE LSPs groupings.  
 */
```

```
grouping lsp-record-route-information_state {  
  description "recorded route information grouping";  
  container incoming-record-route-subobjects {  
    description "RSVP recorded route object incoming information";  
    list incoming-record-route-subobject {  
      when "../te:origin-type != 'ingress'" {  
        description "Applicable on non-ingress LSPs only";  
      }  
      key "index";  
      description  
        "List of RSVP Path record-route objects";  
      leaf index {  
        type leafref {  
          path "../state/index";  
        }  
        description "RRO subobject index";  
      }  
    }  
    container state {  
      config false;  
      description
```

```

        "State parameters for the record route hop";
        uses te-types:record-route-subobject_state;
    }
}
}
container outgoing-record-route-subobjects {
    description "RSVP recorded route object outgoing information";
    list outgoing-record-route-subobject {
        when "../te:origin-type != 'egress'" {
            description "Applicable on non-egress LSPs only";
        }
        key "index";
        description
            "List of RSVP Resv record-route objects";
        leaf index {
            type leafref {
                path "../state/index";
            }
            description "RRO subobject index";
        }
        container state {
            config false;
            description
                "State parameters for the record route hop";
            uses te-types:record-route-subobject_state;
        }
    }
}
}

grouping lsp-explicit-route-information_state {
    description "RSVP-TE LSP explicit-route information";
    container explicit-route-objects {
        description "Explicit route object information";
        list incoming-explicit-route-hop {
            when "../te:origin-type != 'ingress'" {
                description "Applicable on non-ingress LSPs only";
            }
            key "index";
            description
                "List of incoming RSVP Path explicit-route objects";
            leaf index {
                type leafref {
                    path "../state/index";
                }
                description "ERO subobject index";
            }
            container state {

```

```
        config false;
        description
            "State parameters for the explicit route hop";
        uses te-types:explicit-route-hop;
    }
}
list outgoing-explicit-route-hop {
    when "../te:origin-type != 'egress'" {
        description "Applicable on non-egress LSPs only";
    }
    key "index";
    description
        "List of outgoing RSVP Path explicit-route objects";
    leaf index {
        type leafref {
            path "../state/index";
        }
        description "ERO subobject index";
    }
    container state {
        config false;
        description
            "State parameters for the explicit route hop";
        uses te-types:explicit-route-hop;
    }
}
}

grouping lsp-attributes-flags_config {
    description
        "Configuration parameters relating to RSVP-TE LSP
        attribute flags";
    leaf lsp-rerouting {
        type enumeration {
            enum end-to-end-routing {
                description
                    "End-to-end routing desired";
                reference "RFC4920, RFC5420";
            }
            enum boundary-rerouting {
                description
                    "Boundary rerouting desired";
                reference "RFC4920, RFC5420";
            }
            enum segment-based-rerouting {
                description
                    "Segment-based rerouting desired";
            }
        }
    }
}
```

```
        reference "RFC4920, RFC5420";
    }
    }
    description "LSP rerouting types";
}
leaf lsp-integrity-required {
    type boolean;
    description "LSP integrity desired";
    reference "RFC4875";
}
leaf lsp-contiguous {
    type boolean;
    description "Contiguous LSP";
    reference "RFC5151";
}
leaf lsp-stitching-desired {
    type boolean;
    description "Stitched LSP";
    reference "RFC5150";
}
leaf lsp-preplanned {
    type boolean;
    description "Preplanned LSP";
    reference "RFC6001";
}
leaf lsp-oob-mapping {
    type boolean;
    description
        "Mapping is done out-of-band";
    reference "RFC6511";
}
}

grouping lsp-session-attributes-obj-flags_config {
    description
        "Configuration parameters relating to RSVP-TE LSP
        session attribute flags";
    reference
        "RFC4859: Registry for RSVP-TE Session Flags";
    leaf local-recording-desired {
        type boolean;
        description "Path recording is desired.";
        reference "RFC3209";
    }
    leaf se-style-desired {
        type boolean;
        description "SE Style desired";
        reference "RFC3209";
    }
}
```

```
    }
    leaf path-reevaluation-request {
      type boolean;
      description "Path re-evaluation request";
      reference "RFC4736";
    }
    leaf soft-preemption-desired {
      type boolean;
      description "Soft-preemption is desired";
      reference "RFC5712";
    }
  }
}

grouping lsp-properties_config {
  description
    "Configuration parameters relating to RSVP-TE LSP
    session attribute flags";
  leaf lsp-signaled-name {
    type string;
    description
      "Sets the session name to use in the session
      attribute object.";
  }
  uses lsp-session-attributes-obj-flags_config;
  uses lsp-attributes-flags_config;
}

grouping tunnel-properties_config {
  description "RSVP-TE Tunnel properties grouping";
  leaf retry-timer {
    type uint16 {
      range 1..600;
    }
    units seconds;
    description
      "sets the time between attempts to establish the
      LSP";
  }
}
}

/**** End of RSVP-TE LSP groupings ****/

/**
 * RSVP-TE generic global properties.
 */

grouping global-soft-preemption_config {
  description
```

```
    "Configuration for global RSVP-TE soft preemption";
  leaf soft-preemption-timeout {
    type uint16 {
      range 0..300;
    }
    default 0;
    description
      "Timeout value for soft preemption to revert
       to hard preemption";
  }
}

grouping global-soft-preemption {
  description
    "Top level group for RSVP-TE soft-preemption";
  container global-soft-preemption {
    presence "Enables soft preemption on a node.";
    description
      "Top level container for RSVP-TE soft-preemption";
    uses global-soft-preemption_config;
  }
}
/*** End of RSVP-TE generic global properties. ***/

/**
 * RSVP-TE interface generic groupings.
 */

grouping rsvp-te-interface-attributes {
  description
    "Top level grouping for RSVP-TE interface properties.";
  container rsvp-te-interface-attributes {
    description
      "Top level container for RSVP-TE interface
       properties";
    container state {
      config false;
      description
        "State information associated with RSVP-TE
         bandwidth";
    }
  }
}
/*** End of RSVP-TE generic groupings ***/

/* RSVP-TE global properties */
augment "/rt:routing/rt:control-plane-protocols/"
```



```
+ "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals" {
  description
    "RSVP-TE augmentation to RSVP globals";
  uses global-soft-preemption;
}

/* Linkage to the base RSVP all links */
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces" {
  description
    "RSVP-TE generic data augmentation pertaining to interfaces";
  uses rsvp-te-interface-attributes;
}

/* Linkage to per RSVP interface */
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
  "rsvp:interface" {
  description
    "RSVP-TE generic data augmentation pertaining to specific
    interface";
  uses rsvp-te-interface-attributes;
}

/* add augmentation for sessions and neighbors */
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
+ "rsvp:sessions/rsvp:session/rsvp:state/rsvp:psbs/rsvp:psb" {
  description
    "RSVP-TE generic data augmentation pertaining to session";
  /* To be added */
  leaf tspec-average-rate {
    type rt-types:bandwidth-ieee-float32;
    units "Bytes per second";
    description "Tspec Token Bucket Average Rate";
    reference "RFC2210: RSVP with INTSERV";
  }
  leaf tspec-size {
    type rt-types:bandwidth-ieee-float32;
    units "Bytes per second";
    description "Tspec Token Bucket Burst Rate";
    reference "RFC2210";
  }
  leaf tspec-peak-rate {
    type rt-types:bandwidth-ieee-float32;
    units "Bytes per second";
    description "Tspec Token Bucket Peak Data Rate";
    reference "RFC2210";
  }
}
```

```
    }
    leaf min-policed-unit {
        type uint32;
        description "Tspec Minimum Policed Unit";
        reference "RFC2210";
    }
    leaf max-packet-size {
        type uint32;
        description "Tspec Maximum Packet Size";
        reference "RFC2210";
    }
}
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
+ "rsvp:sessions/rsvp:session/rsvp:state/rsvp:rsbs/rsvp:rsb" {
    description
        "RSVP-TE generic data augmentation pertaining to session";
    leaf fspec-average-rate {
        type rt-types:bandwidth-ieee-float32;
        units "Bytes per second";
        description "Fspec Token Bucket Average Rate";
        reference "RFC2210";
    }
    leaf fspec-size {
        type rt-types:bandwidth-ieee-float32;
        units "Bytes per second";
        description "Fspec Token Bucket Burst Rate";
        reference "RFC2210";
    }
    leaf fspec-peak-rate {
        type rt-types:bandwidth-ieee-float32;
        units "Bytes per second";
        description "Fspec Token Bucket Peak Data Rate";
        reference "RFC2210";
    }
    leaf min-policed-unit {
        type uint32;
        description "Fspec Minimum Policed Unit";
        reference "RFC2210";
    }
    leaf max-packet-size {
        type uint32;
        description "Fspec Maximum Packet Size";
        reference "RFC2210";
    }
}

augment "/rt:routing/rt:control-plane-protocols/"
```

```
+ "rt:control-plane-protocol/rsvp:rsvp/rsvp:neighbors" {
  description
    "RSVP-TE generic data augmentation pertaining to neighbors";
  /* To be added */
}

/**
 * RSVP-TE generic augmentations of generic TE model.
 */

/* TE tunnel augmentation */
augment "/te:te/te:tunnels/te:tunnel" {
  when "/te:te/te:tunnels/te:tunnel" +
    "/te:p2p-primary-paths/te:p2p-primary-path" +
    "/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
    description
      "When the path signaling protocol is RSVP-TE ";
  }
  description
    "RSVP-TE generic data augmentation pertaining to TE tunnels";
  uses lsp-properties_config;
  uses tunnel-properties_config;
}

augment "/te:te/te:tunnels/te:tunnel/te:state" {
  when "/te:te/te:tunnels/te:tunnel" +
    "/te:p2p-primary-paths/te:p2p-primary-path" +
    "/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
    description
      "When the path signaling protocol is RSVP-TE ";
  }
  description
    "RSVP-TE generic data augmentation pertaining to TE tunnels";
  uses lsp-properties_config;
  uses tunnel-properties_config;
}

/* TE LSP augmentation */
grouping rsvp-te-lsp-properties {
  description "RSVP-TE LSP properties grouping";
  leaf associated-rsvp-session {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
        + "rsvp:sessions/rsvp:session/rsvp:local-index";
    }
  }
  description
    "If the signalling protocol specified for this path is
```

```
        RSVP-TE, this leaf provides a reference to the associated
        session within the RSVP-TE protocol sessions list, such
        that details of the signaling can be retrieved.";
    }

    uses lsp-properties_config;
    uses lsp-explicit-route-information_state;
    uses lsp-record-route-information_state;
}

augment "/te:te/te:lsps-state/te:lsp" {
    when "/te:te/te:lsps-state/te:lsp" +
        "/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
        description
            "When the signaling protocol is RSVP-TE ";
    }
    description
        "RSVP-TE generic data augmentation pertaining to specific TE
        LSP";
    uses rsvp-te-lsp-properties;
}

augment "/te:te/te:tunnels/te:tunnel/te:p2p-primary-paths" +
    "/te:p2p-primary-path/te:state/te:lsps/te:lsp" {
    when "/te:te/te:tunnels/te:tunnel/te:p2p-primary-paths" +
        "/te:p2p-primary-path/te:state/te:lsps/te:lsp" +
        "/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
        description
            "When the signaling protocol is RSVP-TE ";
    }
    description
        "RSVP-TE generic data augmentation pertaining to specific TE
        LSP";
    uses rsvp-te-lsp-properties;
}

augment "/te:te/te:tunnels/te:tunnel/te:p2p-secondary-paths" +
    "/te:p2p-secondary-path/te:state/te:lsps/te:lsp" {
    when "/te:te/te:tunnels/te:tunnel/te:p2p-primary-paths" +
        "/te:p2p-primary-path/te:state/te:lsps/te:lsp" +
        "/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
        description
            "When the signaling protocol is RSVP-TE ";
    }
    description
        "RSVP-TE generic data augmentation pertaining to specific TE
        LSP";
    uses rsvp-te-lsp-properties;
}
```

```

    }

    /* TE interface augmentation */
    augment "/te:te/te-dev:interfaces/te-dev:interface" {
        description
            "RSVP-TE generic data augmentation pertaining to specific TE
            interface";
    }
}
<CODE ENDS>

```

Figure 3: RSVP TE generic YANG module

2.3. RSVP-TE MPLS Model

The MPLS RSVP-TE YANG module augments the RSVP-TE generic module with parameters to configure and manage signaling of MPLS RSVP-TE LSPs. RSVP-TE YANG modules for other dataplane technologies (e.g. OTN or WDM) are outside the scope of this document and are defined in other documents.

2.3.1. Tree Diagram

The following are possible types of configuration and state data nodes in this module:

- o those augmenting or extending the generic RSVP-TE module
- o those augmenting or extending the TE module
- o those that are specific to the RSVP-TE MPLS module

Below is a YANG tree representation for data items defined in the RSVP-TE MPLS module:

```

module: ietf-rsvp-te-mpls
  augment /rt:routing/rt:control-plane-protocols/
    rt:control-plane-protocol/rsvp:rsvp:
      +--rw fast-reroute-local-revertive
      +--rw rsvp-frr-local-revert-delay?  uint32
  augment /rt:routing/rt:control-plane-protocols/
    rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces:
  augment /rt:routing/rt:control-plane-protocols/
    rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/rsvp:interface:
  augment /rt:routing/rt:control-plane-protocols/
    rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/
  rsvp:sessions/rsvp:session/rsvp:state:
  augment /rt:routing/rt:control-plane-protocols/

```

```

rt:control-plane-protocol/rsvp:rsvp/rsvp:neighbors:
augment /te:te/te:tunnels/te:tunnel:
  +--rw local-protection-desired?          empty
  +--rw bandwidth-protection-desired?      empty
  +--rw node-protection-desired?           empty
  +--rw non-php-desired?                   empty
  +--rw entropy-label-cap?                 empty
  +--rw oam-mep-entities-desired?          empty
  +--rw oam-mip-entities-desired?          empty
augment /te:te/te:lsps-state/te:lsp:
  +--ro state
  |   +--ro local-protection-desired?      empty
  |   +--ro bandwidth-protection-desired?  empty
  |   +--ro node-protection-desired?       empty
  |   +--ro non-php-desired?               empty
  |   +--ro entropy-label-cap?             empty
  |   +--ro oam-mep-entities-desired?      empty
  |   +--ro oam-mip-entities-desired?      empty
  +--ro backup-info
  |   +--ro state
  |   |   +--ro backup-tunnel-name?        string
  |   |   +--ro backup-frr-on?             uint8
  |   |   +--ro backup-protected-lsp-num?  uint32
  augment /te:te/te:tunnels/te:tunnel/te:p2p-primary-paths/
  te:p2p-primary-path/te:state/te:lsps/te:lsp:
    +--ro state
    |   +--ro local-protection-desired?    empty
    |   +--ro bandwidth-protection-desired? empty
    |   +--ro node-protection-desired?     empty
    |   +--ro non-php-desired?              empty
    |   +--ro entropy-label-cap?            empty
    |   +--ro oam-mep-entities-desired?    empty
    |   +--ro oam-mip-entities-desired?    empty
    +--ro backup-info
    |   +--ro state
    |   |   +--ro backup-tunnel-name?      string
    |   |   +--ro backup-frr-on?           uint8
    |   |   +--ro backup-protected-lsp-num? uint32
    augment /te:te/te:tunnels/te:tunnel/te:p2p-secondary-paths/
    te:p2p-secondary-path/te:state/te:lsps/te:lsp:
      +--ro state
      |   +--ro local-protection-desired?    empty
      |   +--ro bandwidth-protection-desired? empty
      |   +--ro node-protection-desired?     empty
      |   +--ro non-php-desired?              empty
      |   +--ro entropy-label-cap?            empty
      |   +--ro oam-mep-entities-desired?    empty
      |   +--ro oam-mip-entities-desired?    empty

```

```

    +--ro backup-info
      +--ro state
        +--ro backup-tunnel-name?      string
        +--ro backup-frr-on?           uint8
        +--ro backup-protected-lsp-num? uint32
augment /te:te/te-dev:interfaces/te-dev:interface:
  +--rw bandwidth-mpls-reservable
    +--rw (bandwidth-value)?
      | +--:(absolute)
      | | +--rw absolute-value?   uint32
      | +--:(percentage)
      | | +--rw percent-value?    uint32
    +--rw (bc-model-type)?
      +--:(bc-model-rdm)
      | +--rw bc-model-rdm
      | | +--rw bandwidth-mpls-constraints
      | | | +--rw maximum-reservable? uint32
      | | | +--rw bc-value*           uint32
      +--:(bc-model-mam)
      | +--rw bc-model-mam
      | | +--rw bandwidth-mpls-constraints
      | | | +--rw maximum-reservable? uint32
      | | | +--rw bc-value*           uint32
      +--:(bc-model-mar)
      | +--rw bc-model-mar
      | | +--rw bandwidth-mpls-constraints
      | | | +--rw maximum-reservable? uint32
      | | | +--rw bc-value*           uint32
augment /te:te/te-dev:interfaces/te-dev:interface:
  +--rw rsvp-te-frr-backups
    +--rw (type)?
      +--:(static-tunnel)
      | +--rw static-backups
      | | +--rw static-backup* [backup-tunnel-name]
      | | +--rw backup-tunnel-name ->
      | /te:te/tunnels/tunnel/name
      +--:(auto-tunnel)
      | +--rw auto-tunnel-backups
      | | +--rw auto-backup-protection? identityref
      | | +--rw auto-backup-path-computation? identityref

```

Figure 4: RSVP-TE MPLS Tree diagram

2.3.2. YANG Module

```

<CODE BEGINS> file "ietf-rsvp-te-mpls@2018-02-19.yang"
module ietf-rsvp-te-mpls {

```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-rsvp-te-mpls";

prefix "rsvp-te-mpls";

import ietf-rsvp {
  prefix "rsvp";
}

import ietf-routing {
  prefix "rt";
}

import ietf-te-mpls-types {
  prefix "te-mpls-types";
}

import ietf-te-types {
  prefix "te-types";
}

import ietf-te {
  prefix "te";
}

import ietf-te-device {
  prefix "te-dev";
}

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>

  WG Chair:   Lou Berger
              <mailto:lberger@labn.net>

  WG Chair:   Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:      Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:      Tarek Saad
              <mailto:tsaad@cisco.com>
```


Editor: Rakesh Gandhi
<mailto:rgandhi@cisco.com>

Editor: Himanshu Shah
<mailto:hshah@ciena.com>

Editor: Xufeng Liu
<mailto:xliu@kuatrotech.com>

Editor: Xia Chen
<mailto:jescia.chenxia@huawei.com>

Editor: Raqib Jones
<mailto:raqib@Brocade.com>

Editor: Bin Wen
<mailto:Bin_Wen@cable.comcast.com>;

description

"Latest update to MPLS RSVP-TE YANG data model.";

```
revision "2018-02-19" {  
  description "Update to MPLS RSVP-TE YANG initial revision.";  
  reference "RFC3209, RFC6511, RFC6790, RFC7260, RFC4859, RFC4090";  
}
```

```
/* RSVP-TE MPLS LSPs groupings */  
grouping lsp-attributes-flags-mpls_config {  
  description  
    "Configuration parameters relating to RSVP-TE MPLS LSP  
    attribute flags";  
  leaf non-php-desired {  
    type empty;  
    description  
      "Non-PHP is desired";  
    reference "RFC6511";  
  }  
  leaf entropy-label-cap {  
    type empty;  
    description "Entropy label capability";  
    reference "RFC6790";  
  }  
  leaf oam-mep-entities-desired {  
    type empty;  
    description "OAM MEP entities desired";  
    reference "RFC7260";  
  }  
  leaf oam-mip-entities-desired {
```

```
        type empty;
        description "OAM MIP entities desired";
        reference "RFC7260";
    }
}

grouping lsp-session-attributes-obj-flags-mpls_config {
    description
        "Configuration parameters relating to RSVP-TE MPLS LSP
        session attribute flags";
    reference
        "RFC4859: Registry for RSVP-TE Session Flags";
    leaf local-protection-desired {
        type empty;
        description "Fastreroute local protection is desired.";
        reference
            "RFC4859: Registry for RSVP-TE Session Flags";
    }
    leaf bandwidth-protection-desired {
        type empty;
        description
            "Request FRR bandwidth protection on LSRs if
            present.";
        reference "RFC4090";
    }
    leaf node-protection-desired {
        type empty;
        description
            "Request FRR node protection on LSRs if
            present.";
        reference "RFC4090";
    }
}

grouping tunnel-properties-mpls_config {
    description
        "Top level grouping for LSP properties.";
    uses lsp-session-attributes-obj-flags-mpls_config;
    uses lsp-attributes-flags-mpls_config;
}

grouping lsp-properties-mpls {
    description
        "Top level grouping for LSP properties.";
    container state {
        config false;
        description
            "Configuration applied parameters and state";
    }
}
```

```
        uses lsp-session-attributes-obj-flags-mpls_config;
        uses lsp-attributes-flags-mpls_config;
    }
}
/* End of RSVP-TE MPLS LSPs groupings */

/* MPLS RSVP-TE interface groupings */
grouping rsvp-te-interface_state {
    description
        "The RSVP-TE interface state grouping";
    leaf over-subscribed-bandwidth {
        type uint32;
        description
            "The amount of over-subscribed bandwidth on
             the interface";
    }
}

grouping rsvp-te-interface-softpreemption_state {
    description
        "The RSVP-TE interface preeemptions state grouping";
    container interface-softpreemption-state {
        description
            "The RSVP-TE interface preeemptions state grouping";
        leaf soft-preempted-bandwidth {
            type uint32;
            description
                "The amount of soft-preempted bandwidth on
                 this interface";
        }
    }
    list lsps {
        key
            "source destination tunnel-id lsp-id "+
            "extended-tunnel-id";
        description
            "List of LSPs that are soft-preempted";
        leaf source {
            type leafref {
                path "/te:te/te:lsps-state/te:lsp/"+
                "te:source";
            }
            description
                "Tunnel sender address extracted from
                 SENDER_TEMPLATE object";
            reference "RFC3209";
        }
        leaf destination {
            type leafref {
```

```
        path "/te:te/te:lsps-state/te:lsp/" +
            "te:destination";
    }
    description
        "Tunnel endpoint address extracted from
        SESSION object";
    reference "RFC3209";
}
leaf tunnel-id {
    type leafref {
        path "/te:te/te:lsps-state/te:lsp/" +
            "te:tunnel-id";
    }
    description
        "Tunnel identifier used in the SESSION
        that remains constant over the life
        of the tunnel.";
    reference "RFC3209";
}
leaf lsp-id {
    type leafref {
        path "/te:te/te:lsps-state/te:lsp/" +
            "te:lsp-id";
    }
    description
        "Identifier used in the SENDER_TEMPLATE
        and the FILTER_SPEC that can be changed
        to allow a sender to share resources with
        itself.";
    reference "RFC3209";
}
leaf extended-tunnel-id {
    type leafref {
        path "/te:te/te:lsps-state/te:lsp/" +
            "te:extended-tunnel-id";
    }
    description
        "Extended Tunnel ID of the LSP.";
    reference "RFC3209";
}
leaf type {
    type leafref {
        path "/te:te/te:lsps-state/te:lsp/" +
            "te:type";
    }
    description "LSP type P2P or P2MP";
}
}
```

```
    }  
  }  
  
  grouping bandwidth-mpls-constraints {  
    description "Bandwidth constraints.";   
    container bandwidth-mpls-constraints {  
      description  
        "Holds the bandwidth constraints properties";  
      leaf maximum-reservable {  
        type uint32 {  
          range "0..4294967295";  
        }  
        description  
          "The maximum reservable bandwidth on the  
          interface";  
      }  
      leaf-list bc-value {  
        type uint32 {  
          range "0..4294967295";  
        }  
        max-elements 8;  
        description  
          "The bandwidth constraint type";  
      }  
    }  
  }  
}  
  
grouping bandwidth-constraint-values {  
  description  
    "Packet bandwidth constraints values";  
  choice value-type {  
    description  
      "Value representation";  
    case percentages {  
      container perc-values {  
        uses bandwidth-mpls-constraints;  
        description  
          "Percentage values";  
      }  
    }  
    case absolutes {  
      container abs-values {  
        uses bandwidth-mpls-constraints;  
        description  
          "Absolute values";  
      }  
    }  
  }  
}
```

```
}

grouping bandwidth-mpls-reservable_config {
  description
    "Interface bandwidth reservable configuration grouping";
  choice bandwidth-value {
    description "Reservable bandwidth configuration choice";
    case absolute {
      leaf absolute-value {
        type uint32;
        description "Absolute value of the bandwidth";
      }
    }
    case percentage {
      leaf percent-value {
        type uint32 {
          range "0..4294967295";
        }
        description "Percentage reservable bandwidth";
      }
    }
    description
      "The maximum reservable bandwidth on the
      interface";
  }
}

choice bc-model-type {
  description
    "Reservable bandwidth percentage capacity
    values.";
  case bc-model-rdm {
    container bc-model-rdm {
      description
        "Russian Doll Model Bandwidth Constraints.";
      uses bandwidth-mpls-constraints;
    }
  }
  case bc-model-mam {
    container bc-model-mam {
      uses bandwidth-mpls-constraints;
      description
        "Maximum Allocation Model Bandwidth
        Constraints.";
    }
  }
  case bc-model-mar {
    container bc-model-mar {
      uses bandwidth-mpls-constraints;
      description

```

```
        "Maximum Allocation with Reservation Model
        Bandwidth Constraints.";
    }
}
}

grouping bandwidth-mpls-reservable {
  description
    "Packet reservable bandwidth";
  container bandwidth-mpls-reservable {
    description
      "Interface bandwidth reservable container";
    uses bandwidth-mpls-reservable_config;
  }
}
/* End of RSVP-TE interface groupings */

/* RSVP-TE FRR groupings */
grouping rsvp-te-frr-auto-tunnel-backup_config {
  description
    "Auto-tunnel backup configuration grouping";
  leaf auto-backup-protection {
    type identityref {
      base te-mpls-types:backup-protection-type;
    }
    default
      te-mpls-types:backup-protection-node-link;
    description
      "Describes whether the backup should offer
      protection against link, node, or either";
  }
  leaf auto-backup-path-computation {
    type identityref {
      base
        te-types:path-computation-srlg-type;
    }
    description
      "FRR backup computation type";
  }
}

grouping rsvp-te-frr-backups_config {
  description
    "Top level container for RSVP-TE FRR backup parameters";
  choice type {
    description
      "FRR backup tunnel type";
  }
}
```

```
    case static-tunnel {
      container static-backups {
        description "List of static backups";
        list static-backup {
          key "backup-tunnel-name";
          description
            "List of static backup tunnels that
            protect the RSVP-TE interface.";
          leaf backup-tunnel-name {
            type leafref {
              path "/te:te/te:tunnels/te:tunnel/te:name";
            }
            description "FRR Backup tunnel name";
          }
        }
      }
    }
  }
  case auto-tunnel {
    container auto-tunnel-backups {
      description "Auto-tunnel choice";
      uses rsvp-te-frr-auto-tunnel-backup_config;
    }
  }
}

grouping rsvp-te-frr-backups {
  description
    "RSVP-TE facility backup grouping";
  container rsvp-te-frr-backups {
    description
      "RSVP-TE facility backup properties";
    uses rsvp-te-frr-backups_config;
  }
}

grouping lsp-backup-info_state {
  description "LSP backup information grouping";
  leaf backup-tunnel-name {
    type string;
    description
      "If an LSP has an FRR backup LSP that can protect it,
      this field identifies the tunnel name of the backup LSP.
      Otherwise, this field is empty.";
  }
  leaf backup-frr-on {
    type uint8;
    description

```



```
        "Whether currently this backup is carrying traffic";
    }
    leaf backup-protected-lsp-num {
        type uint32;
        description
            "Number of LSPs protected by this backup";
    }
}

grouping lsp-backup-info {
    description "Backup/bypass LSP related information";
    container backup-info {
        description
            "backup information";
        container state {
            config false;
            description
                "Configuration applied parameters and state";
            uses lsp-backup-info_state;
        }
    }
}

grouping fast-reroute-local-revertive_config {
    description "RSVP-TE FRR local revertive grouping";
    leaf rsvp-frr-local-revert-delay {
        type uint32;
        description
            "Time to wait after primary link is restored
            before node attempts local revertive
            procedures.";
    }
}

/** End of RSVP-TE FRR backup information */

grouping fast-reroute-local-revertive {
    description
        "Top level grouping for globals properties";
    container fast-reroute-local-revertive {
        description "RSVP-TE FRR local revertive container";
        uses fast-reroute-local-revertive_config;
    }
}

/* RSVP-TE global properties */
augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol/rsvp:rsvp" {
```

```
    description
      "RSVP-TE augmentation to RSVP globals";
    uses fast-reroute-local-revertive;
  }

  /* Linkage to the base RSVP all interfaces */
  augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces" {
    description
      "Augmentations for RSVP-TE MPLS all interfaces properties";
    /* To be added */
  }

  /* Linkage to per RSVP interface */
  augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:interfaces/" +
  "rsvp:interface" {
    description
      "Augmentations for RSVP-TE MPLS per interface properties";
    /* To be added */
  }

  /* add augmentation for sessions neighbors */
  augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:globals/"
  + "rsvp:sessions/rsvp:session/rsvp:state" {
    description
      "Augmentations for RSVP-TE MPLS sessions";
    /* To be added */
  }

  augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol/rsvp:rsvp/rsvp:neighbors" {
    description
      "Augmentations for RSVP-TE MPLS neighbors properties";
    /* To be added */
  }

  /**
   * Augmentation to TE generic module
   */
  augment "/te:te/te:tunnels/te:tunnel" {
    description
      "Augmentations for RSVP-TE MPLS TE tunnel properties";
    uses tunnel-properties-mpls_config;
  }

  augment "/te:te/te:lsps-state/te:lsp" {
```

```
when "/te:te/te:lsps-state/te:lsp" +
"/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
  description
    "When the signaling protocol is RSVP-TE ";
}
description
  "RSVP-TE MPLS LSP state properties";
uses lsp-properties-mpls;
uses lsp-backup-info;
}

augment "/te:te/te:tunnels/te:tunnel/te:p2p-primary-paths" +
"/te:p2p-primary-path/te:state/te:lsps/te:lsp" {
  when "/te:te/te:tunnels/te:tunnel" +
"/te:p2p-secondary-paths/te:p2p-secondary-path/" +
"/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
    description
      "When the signaling protocol is RSVP-TE ";
  }
  description
    "RSVP-TE MPLS LSP state properties";
  uses lsp-properties-mpls;
  uses lsp-backup-info;
}

augment "/te:te/te:tunnels/te:tunnel/te:p2p-secondary-paths" +
"/te:p2p-secondary-path/te:state/te:lsps/te:lsp" {
  when "/te:te/te:tunnels/te:tunnel" +
"/te:p2p-secondary-paths/te:p2p-secondary-path/" +
"/te:path-setup-protocol = 'te-types:path-setup-rsvp'" {
    description
      "When the signaling protocol is RSVP-TE ";
  }
  description
    "RSVP-TE MPLS LSP state properties";
  uses lsp-properties-mpls;
  uses lsp-backup-info;
}

augment "/te:te/te-dev:interfaces/te-dev:interface" {
  description
    "RSVP reservable bandwidth configuration properties";
  uses bandwidth-mpls-reservable;
}

augment "/te:te/te-dev:interfaces/te-dev:interface" {
  description
    "RSVP reservable bandwidth configuration properties";
```

```
    uses rsvp-te-frr-backups;
  }
}
<CODE ENDS>
```

Figure 5: RSVP TE MPLS YANG module

Figure 5 shows the YANG tree representation of the RSVP TE MPLS module that augments RSVP-TE module as well as RSVP and TE YANG modules.

3. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp-te XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-rsvp-te-mpls XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-rsvp namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp-te
prefix: ietf-rsvp reference: RFC3209

name: ietf-rsvp-te namespace: urn:ietf:params:xml:ns:yang:ietf-rsvp-te-mpls
prefix: ietf-rsvp-te reference: RFC3209

4. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF

users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which 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.

5. Acknowledgement

The authors would like to thank Lou Berger for reviewing and providing valuable feedback on this document.

6. Contributors

Xia Chen
Huawei Technologies

Email: jescia.chenxia@huawei.com

Raqib Jones
Brocade

Email: raqib@Brocade.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com

7. Normative References

[I-D.ietf-teas-yang-rsvp]

Beeram, V., Saad, T., Gandhi, R., Liu, X., Bryskin, I., and H. Shah, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-08 (work in progress), October 2017.

[I-D.ietf-teas-yang-te]

Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-11 (work in progress), February 2018.

[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>>.

- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [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>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.

Authors' Addresses

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Tarek Saad (editor)
Cisco Systems, Inc.

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems, Inc.

Email: rgandhi@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

Himanshu Shah
Ciena

Email: hshah@ciena.com

TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

T. Saad, Ed.
R. Gandhi
Cisco Systems Inc
X. Liu
Jabil
V. Beeram
Juniper Networks
H. Shah
Ciena
I. Bryskin
Huawei Technologies
March 05, 2018

A YANG Data Model for Traffic Engineering Tunnels and Interfaces
draft-ietf-teas-yang-te-14

Abstract

This document defines a YANG data model for the configuration and management of Traffic Engineering (TE) interfaces, tunnels and Label Switched Paths (LSPs). The model is divided into YANG modules that classify data into generic, device-specific, technology agnostic, and technology-specific elements. The model also includes module(s) that contain reusable TE data types and data groupings.

This model covers data for configuration, operational state, remote procedural calls, and event notifications.

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 September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
1.2. Tree Diagram	3
1.3. Prefixes in Data Node Names	4
1.4. Open Issues and Next Steps	5
1.4.1. TE Technology Models	5
1.4.2. State Data Organization	6
2. Model Overview	6
2.1. Module(s) Relationship	6
2.2. Design Considerations	8
2.3. Optional Features	9
2.4. Configuration Inheritance	9
3. TE Generic Model Organization	9
3.1. Global Configuration and State Data	10
3.2. Interfaces Configuration and State Data	11
3.3. Tunnels Configuration and State Data	12
3.3.1. Tunnel Compute-Only Mode	13
3.3.2. Tunnel Hierarchical Link Endpoint	14
3.4. TE LSPs State Data	14
3.5. Global RPC Data	14
3.6. Interface RPC Data	14
3.7. Tunnel RPC Data	14
3.8. Global Notifications Data	15
3.9. Interfaces Notifications Data	15
3.10. Tunnel Notification Data	15
4. TE Generic and Helper YANG Modules	52
5. IANA Considerations	164
6. Security Considerations	165
7. Acknowledgement	166
8. Contributors	166
9. Normative References	167

Authors' Addresses	168
--------------------	-----

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. RESTCONF [RFC8040]) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document describes the YANG data models for TE Tunnels, Label Switched Paths (LSPs) and TE interfaces that cover data applicable to generic or device-independent, device-specific, Multiprotocol Label Switching (MPLS) technology specific, and Segment Routing (SR) TE technology. It also describes helper modules that define TE grouping(s) and data types that can be imported by other modules.

The document defines the high-level relationship between the modules defined in this document, as well as other external protocol modules. It is expected other data plane technology model(s) will augment the TE generic model. Also, the TE generic model does not include any data specific to a signaling protocol. It is expected YANG models for TE signaling protocols, such as RSVP-TE ([RFC3209], [RFC3473]), or Segment-Routing TE (SR-TE) will augment the TE generic module.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

1.2. Tree Diagram

A simplified graphical representation of the data model is presented in each section of the model. The following notations are used for the YANG model data tree representation.

<status> <flags> <name> <opts> <type>

<status> is one of:

- + for current
- x for deprecated
- o for obsolete

<flags> is one of:

- rw for read-write configuration data
- ro for read-only non-configuration data
- x for execution rpcs
- n for notifications

<name> is the name of the node

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>

<opts> is one of:

- ? for an optional leaf or node
- ! for a presence container
- * for a leaf-list or list
- Brackets [<keys>] for a list's keys
- Curly braces {<condition>} for optional feature that make node conditional
- Colon : for marking case nodes
- Ellipses ("...") subtree contents not shown
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

<type> is the name of the type for leafs and leaf-lists.

1.3. 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
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
te	ietf-te	this document
te-types	ietf-te-types	this document
te-mpls-types	ietf-te-mpls-types	this document
te-dev	ietf-te-device	this document
te-mpls	ietf-te-mpls	this document
te-sr-mpls	ietf-te-sr-mpls	this document

Table 1: Prefixes and corresponding YANG modules

1.4. Open Issues and Next Steps

This section describes the number of open issues that are under consideration. As issues are resolved, this section will be updated to reflect this and be left there for reference. It is expected that all the issues in this section will be addressed before the document will be ready for final publication.

1.4.1. TE Technology Models

This document describes the generic TE YANG data model that is independent of any dataplane technology. One of the design objectives is to allow specific data plane technologies models to reuse the generic TE data model and possibly augment it with technology specific data model(s). There are multiple options being considered to achieve this:

- o The generic TE model, including the lists of TE tunnels, LSPs, and interfaces can be defined and rooted at the top of the YANG tree. Specific leaf(s) under the TE tunnel, LSP, or interface, in this case, can identify the specific technology layer that it belongs to. This approach implies a single list for each of TE tunnel(s), LSP(s), and interface(s) in the model carries elements of different technology layers.
- o An instance of the generic TE YANG model can be mounted in the YANG tree once for each TE technology layer(s). This approach provides separation of elements belonging to different technology layers into separate lists per layer in the data model.
- o The generic TE data node(s) and TE list(s) for tunnels, LSPs, and interfaces are defined as grouping(s) in a separate module. The

specific technology layer imports the generic TE groupings and uses them their respective technology specific module.

This revision of the model leverages the LSP encoding type of a tunnel (and interfaces) to identify the specific technology associated with the a TE interfaces, tunnel(s) and the LSP(s). For example, for an MPLS TE LSP, the LSP encoding type is assumed to be "lsp-encoding-packet".

Finally, the TE generic model does not include any signaling protocol data. It is expected that TE signaling protocol module(s) will be defined in other document(s) that will cover the RSVP-TE ([RFC3209], [RFC3473]), and Segment-Routing TE (SR-TE) model and that augment the TE generic model.

1.4.2. State Data Organization

Pure state data (for example, ephemeral or protocol derived state objects) can be modeled using one of the options below:

- o Contained inside a read-write container, in a "state" sub-container, as shown in Figure 3
- o Contained inside a separate read-only container, for example a lsps-state container

The Network Management Datastore Architecture (NMDA) addresses the "OpState" that was discussed in the IETF. As per NMDA guidelines for new models and models that are not concerned with the operational state of configuration information, this revision of the draft adopts the NMDA proposal for configuration and state data of this model.

2. Model Overview

The data model defined in this document covers the core TE features that are commonly supported across different vendor implementations. The support of extended or vendor specific TE feature(s) are expected to be in augmentations to the data models defined in this document.

2.1. Module(s) Relationship

The TE generic model defined in "ietf-te.yang" covers the building blocks that are device independent and agnostic of any specific technology or control plane instances. The TE device model defined in "ietf-te-device.yang" augments the TE generic model and covers data that is specific to a device - for example, attributes of TE interfaces, or TE timers that are local to a TE node.

The TE data relevant to a specific instantiations of data plane technology exists in a separate YANG module(s) that augment the TE generic model. For example, the MPLS-TE module "ietf-te-mpls.yang" is defined in Figure 10 and augments the TE generic model as shown in Figure 1. Similarly, the module "ietf-te-sr-mpls.yang" models the Segment Routing (SR) TE specific data and augments the TE generic and MPLS-TE model(s).

The TE data relevant to a TE specific signaling protocol instantiation is outside the scope and is covered in other documents. For example, the RSVP-TE [RFC3209] YANG model augmentation of the TE model is covered in [I-D.ietf-teas-yang-rsvp], and other signaling protocol model(s) (e.g. for Segment-Routing TE) are expected to also augment the TE generic model.

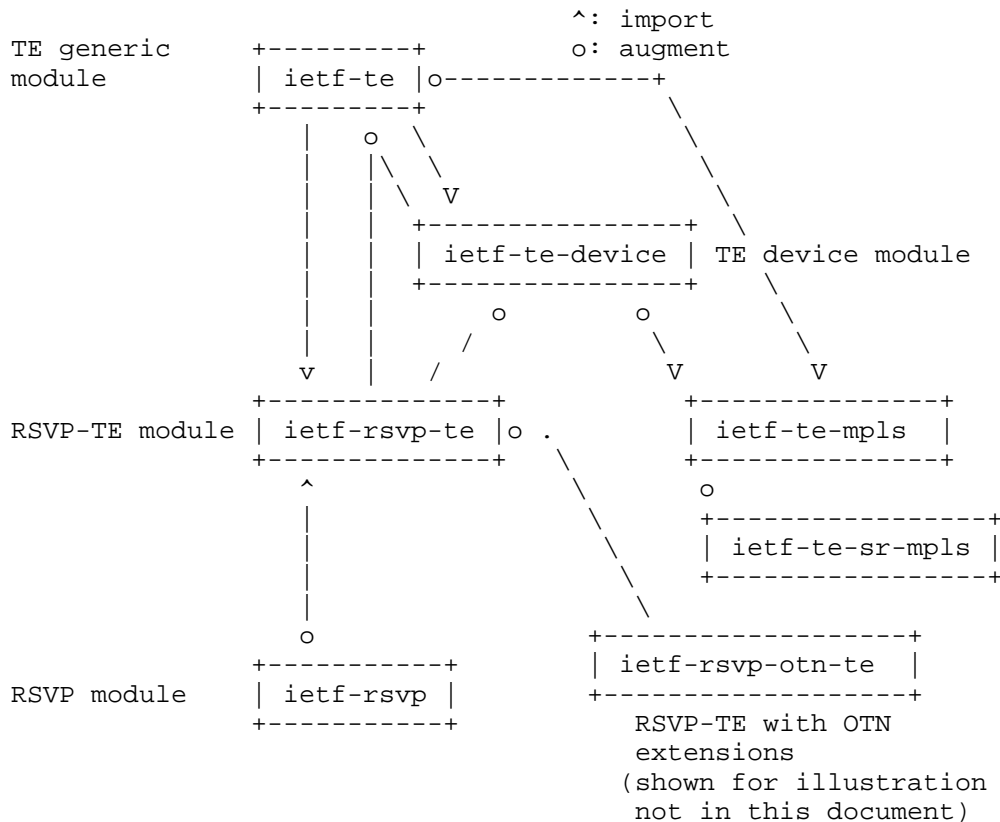


Figure 1: Relationship of TE module(s) with other signaling protocol modules

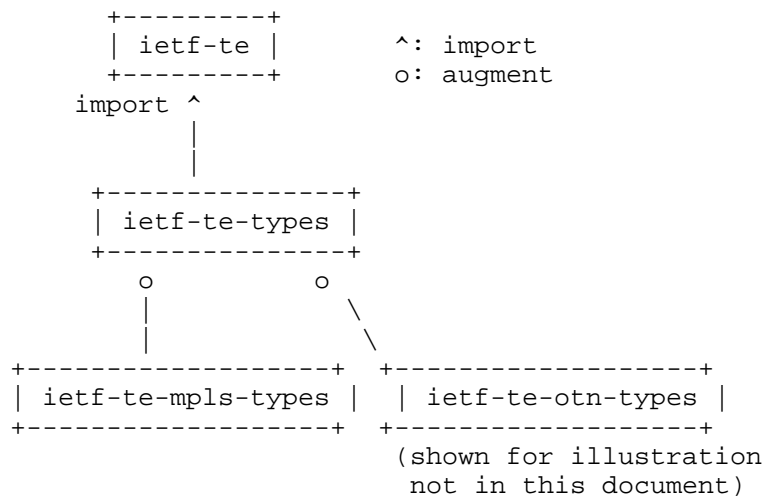


Figure 2: Relationship between generic and technology specific TE types modules

2.2. Design Considerations

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

- o reusable data elements are grouped into separate TE types module(s) that can be readily imported by other modules whenever needed
- o reusable TE data types that are data plane independent are grouped in the TE generic types module "ietf-te-types.yang"
- o reusable TE data elements that are data plane specific (e.g. packet MPLS or switching technologies as defined in [RFC3473]) are expected to be grouped in a technology- specific types module, e.g. "ietf-te-mpls-types.yang". It is expected that technology specific types will augment TE generic types as shown in Figure 2
- o The TE generic model contains device independent data and can be used to model data off a device (e.g. on a controller). The TE data that is device-specific are grouped in a separate module as shown in Figure 1.
- o In general, little information in the model is designated as "mandatory", to allow freedom to vendors to adapt the data model to their specific product implementation.

2.3. Optional Features

Optional features that are beyond the base TE model are left to the specific vendor to decide support using vendor model augmentation and/or using feature checks.

This model declares a number of TE functions as features (such as P2MP-TE, soft-preemption etc.).

2.4. Configuration Inheritance

The defined data model supports configuration inheritance for tunnels, paths, and interfaces. Data elements defined in the main container (e.g. that encompasses the list of tunnels, interfaces, or paths) are assumed to apply equally to all elements of the list, unless overridden explicitly for a certain element of a list (e.g. a tunnel, interface or path).

3. TE Generic Model Organization

The TE generic model covers configuration, state, RPCs, and notifications data pertaining to TE global parameters, interfaces, tunnels and LSPs parameters that are device independent.

The container "te" is the top level container in this data model. The presence of this container is expected to enable TE function system wide.

The model follows the guidelines in for modeling the intended, applied and derived state.


```
module: ietf-te
  +--rw te!
    +--rw globals
      .
      .
    +--rw tunnels
      .
      .
    +-- lsp-state

rpcs:
  +---x globals-rpc
  +---x tunnels-rpc
notifications:
  +---n globals-notif
  +---n tunnels-notif
```

Figure 3: TE generic highlevel model view

3.1. Global Configuration and State Data

This branch of the data model covers configurations that control TE features behavior system-wide, and its respective state. Examples of such configuration data are:

- o Table of named SRLG mappings
- o Table of named (extended) administrative groups mappings
- o Table of named explicit paths to be referenced by TE tunnels
- o Table of named path-constraints sets
- o Auto-bandwidth global parameters
- o TE diff-serve TE-class maps
- o System-wide capabilities for LSP reoptimization (included in the TE device model)
 - * Reoptimization timers (periodic interval, LSP installation and cleanup)
- o System-wide capabilities for TE state flooding (included in the TE device model)

- * Periodic flooding interval
- o Global capabilities that affect the originating, traversing and terminating LSPs. For example:
 - * Path selection parameters (e.g. metric to optimize, etc.)
 - * Path or segment protection parameters

The global state data is represented under the global "state" sub-container as shown in Figure 3.

Examples of such states are:

- o Global statistics (signaling, admission, preemption, flooding)
- o Global counters (number of tunnels/LSPs/interfaces)

3.2. Interfaces Configuration and State Data

This branch of the model covers configuration and state data items corresponding to TE interfaces that are present on a specific device. A new module is introduced that holds the TE device specific properties.

Examples of TE interface properties are:

- o Maximum reservable bandwidth, bandwidth constraints (BC)
- o Flooding parameters
 - * Flooding intervals and threshold values
- o Fast reroute backup tunnel properties (such as static, auto-tunnel)
- o interface attributes
 - * (Extended) administrative groups
 - * SRLG values
 - * TE metric value

The state corresponding to the TE interfaces applied configuration, protocol derived state, and stats and counters all fall under the interface "state" sub-container as shown in Figure 4 below:

```
module: ietf-te
  +--rw te!
    +--rw interfaces
      .
      +-- rw te-attributes
        +-- rw config
          <<intended configuration>>
        .
        +-- ro state
          <<applied configuration>>
          <<derived state associated with the TE interface>>
```

Figure 4: TE interface state

This covers state data for TE interfaces such as:

- o Bandwidth information: maximum bandwidth, available bandwidth at different priorities and for each class-type (CT)
- o List of admitted LSPs
 - * Name, bandwidth value and pool, time, priority
- o Statistics: state counters, flooding counters, admission counters (accepted/rejected), preemption counters
- o Adjacency information
 - * Neighbor address
 - * Metric value

3.3. Tunnels Configuration and State Data

This branch of the model covers intended, and corresponding applied configuration for tunnels. As well, it holds possible derived state pertaining to TE tunnels.

```
module: ietf-te
  +--rw te!
    +--rw tunnels
      .
      +-- rw config
        <<intended configuration>>
      .
      +-- ro state
        <<applied configuration>>
        <<derived state associated with the tunnel>>
```

Figure 5: TE interface state tree

Examples of tunnel configuration data for TE tunnels:

- o Name and type (e.g. P2P, P2MP) of the TE tunnel
- o Admin-state
- o Set of primary and corresponding secondary paths
- o Routing usage (auto-route announce, forwarding adjacency)
- o Policy based routing (PBR) parameters

3.3.1. Tunnel Compute-Only Mode

By default, a configured TE tunnel is provisioned so it can carry traffic as soon as a valid path is computed and an LSP instantiated in the network. In other cases, a TE tunnel may be provisioned for computed path reporting purposes without the need to instantiate an LSP or commit resources in the network. In such a case, a tunnel configuration in "compute-only" mode to distinguish it from default tunnel behavior.

A "compute-only" TE tunnel is configured as a usual TE tunnel with associated path constraint(s) and properties on a device or controller. The device or controller is expected to compute the feasible path(s) subject to configured constraints for of "compute-only" tunnel and reflect the computed path(s) in the LSP(s) Record-Route Object (RRO) list. A client may query "on-demand" the "compute-only" TE tunnel computed path(s) properties by querying the state of the tunnel. Alternatively, the client can subscribe on the "compute-only" TE tunnel to be notified of computed path(s) and whenever it changes.

3.3.2. Tunnel Hierarchical Link Endpoint

TE LSPs can be set up in MPLS or Generalized MPLS (GMPLS) networks to be used to form links to carry traffic in in other (client) networks [RFC6107]. In this case, the model introduces the TE tunnel hierarchical link endpoint parameters to identify the specific link in the client layer that the TE tunnel is associated with.

3.4. TE LSPs State Data

TE LSPs are derived state data that is usually instantiated via signaling protocols. TE LSPs exists on routers as ingress (starting point of LSP), transit (mid-point of LSP), or egress (termination point of the LSP). TE LSPs are distinguished by the 5 tuple, and LSP type (P2P or P2MP). In the model, the nodes holding LSPs data exist in the read-only lsp-state list as show in Figure 6.

3.5. Global RPC Data

This branch of the model covers system-wide RPC execution data to trigger actions and optionally expect responses. Examples of such TE commands are to:

- o Clear global TE statistics of various features

3.6. Interface RPC Data

This collection of data in the model defines TE interface RPC execution commands. Examples of these are to:

- o Clear TE statistics for all or for individual TE interfaces
- o Trigger immediate flooding for one or all TE interfaces

3.7. Tunnel RPC Data

This branch of the model covers TE tunnel RPC execution data to trigger actions and optionally expect responses. Examples of such TE commands are:

- o Clear statistics for all or for individual tunnels
- o Trigger the tear and setup of existing tunnels or LSPs.

3.8. Global Notifications Data

This branch of the model covers system-wide notifications data. The node notifies the registered events to the server using the defined notification messages.

3.9. Interfaces Notifications Data

This branch of the model covers TE interfaces related notifications data. The TE interface configuration is used for specific events registration. Notifications are sent for registered events to the server. Example events for TE interfaces are:

- o Interface creation and deletion
- o Interface state transitions
- o (Soft) preemption triggers
- o Fast reroute activation

3.10. Tunnel Notification Data

This branch of the model covers TE tunnels related notifications data. The TE tunnels configuration is used for specific events registration. Notifications are sent for registered events to the server. Example events for TE tunnels are:

- o Tunnel creation and deletion events
- o Tunnel state up/down changes
- o Tunnel state reoptimization changes

Figure Figure 6 below shows the tree diagram of the YANG model defined in modules: ietf-te.yang, ietf-te-device.yang, ietf-te-mpls.yang, and ietf-te-sr.yang.

```
module: ietf-te
+--rw te!
  +--rw globals
    |   +--rw named-admin-groups
    |   |   +--rw named-admin-group* [name]
    |   |   {te-types:extended-admin-groups,te-types:named-extended-admin-
    |   |   groups}?
    |   |   +--rw name                string
    |   |   +--rw bit-position?      uint32
    |   +--rw named-srlgs
```

```

+---rw named-srlg* [name] {te-types:named-srlg-groups}?
+---rw name string
+---rw group? te-types:srlg
+---rw cost? uint32
+---rw named-path-constraints
+---rw named-path-constraint* [name]
{te-types:named-path-constraints}?
+---rw name string
+---rw te-bandwidth
+---rw (technology)?
+---:(generic)
+---rw generic? te-bandwidth
+---rw setup-priority? uint8
+---rw hold-priority? uint8
+---rw signaling-type? identityref
+---rw disjointness?
te-types:te-path-disjointness
+---rw path-metric-bounds
+---rw path-metric-bound* [metric-type]
+---rw metric-type identityref
+---rw upper-bound? uint64
+---rw path-affinities
+---rw constraints* [usage]
+---rw usage identityref
+---rw (style)?
+---:(value)
+---rw value? te-types:admin-groups
+---:(named)
+---rw affinity-names* [name]
+---rw name string
+---rw path-srlgs
+---rw (style)?
+---:(values)
+---rw usage? identityref
+---rw values* te-types:srlg
+---:(named)
+---rw constraints
+---rw constraint* [usage]
+---rw usage identityref
+---rw constraint
+---rw srlg-names* [name]
+---rw name string
+---rw explicit-route-objects
+---rw route-object-exclude-always* [index]
+---rw index uint32
+---rw (type)?
+---:(numbered)
+---rw numbered-hop

```

```

+--rw address?          te-types:te-tp-id
+--rw hop-type?         te-hop-type
+--rw direction?        te-link-direction
+--:(as-number)
+--rw as-number-hop
+--rw as-number?        binary
+--rw hop-type?         te-hop-type
+--:(unnumbered)
+--rw unnumbered-hop
+--rw node-id?          te-types:te-node-id
+--rw link-tp-id?       te-types:te-tp-id
+--rw hop-type?         te-hop-type
+--rw direction?        te-link-direction
+--:(label)
+--rw label-hop
+--rw te-label
+--rw (technology)?
|   +--:(generic)
|   +--rw generic?
rt-types:generalized-label
+--rw direction?        te-label-direction
+--rw route-object-include-exclude* [index]
+--rw explicit-route-usage? identityref
+--rw index              uint32
+--rw (type)?
+--:(numbered)
+--rw numbered-hop
+--rw address?          te-types:te-tp-id
+--rw hop-type?         te-hop-type
+--rw direction?        te-link-direction
+--:(as-number)
+--rw as-number-hop
+--rw as-number?        binary
+--rw hop-type?         te-hop-type
+--:(unnumbered)
+--rw unnumbered-hop
+--rw node-id?          te-types:te-node-id
+--rw link-tp-id?       te-types:te-tp-id
+--rw hop-type?         te-hop-type
+--rw direction?        te-link-direction
+--:(label)
+--rw label-hop
+--rw te-label
+--rw (technology)?
|   +--:(generic)
|   +--rw generic?
rt-types:generalized-label
+--rw direction?        te-label-direction

```



```
|      |      |      |--rw shared-resources-tunnels  
|      |      |      |--rw lsp-shared-resources-tunnel*    te:tunnel-ref  
|--rw path-in-segment!  
|      |      |--rw forward  
|          |--rw label-set* [inclusive-exclusive label-start]  
|              |--rw inclusive-exclusive        enumeration  
|                  |--rw label-start  
rt-types:generalized-label  
|          |--rw label-end?  
rt-types:generalized-label  
|      |      |--rw range-bitmap?                binary  
|          |--rw reverse  
|              |--rw label-set* [inclusive-exclusive label-start]  
|                  |--rw inclusive-exclusive        enumeration  
|                      |--rw label-start  
rt-types:generalized-label  
|          |--rw label-end?  
rt-types:generalized-label  
|      |      |--rw range-bitmap?                binary  
|--rw path-out-segment!  
|      |      |--rw forward  
|          |--rw label-set* [inclusive-exclusive label-start]  
|              |--rw inclusive-exclusive        enumeration  
|                  |--rw label-start  
rt-types:generalized-label  
|          |--rw label-end?  
rt-types:generalized-label  
|      |      |--rw range-bitmap?                binary  
|          |--rw reverse  
|              |--rw label-set* [inclusive-exclusive label-start]  
|                  |--rw inclusive-exclusive        enumeration  
|                      |--rw label-start  
rt-types:generalized-label  
|          |--rw label-end?  
rt-types:generalized-label  
|      |      |--rw range-bitmap?                binary  
|--ro state  
|      |--ro bandwidth-generic_state?    te-types:te-bandwidth  
|      |--ro disjointness_state?  
te-types:te-path-disjointness  
|      |--rw te-mpls:bandwidth  
|          |--rw te-mpls:specification-type?  
te-mpls-types:te-bandwidth-type  
|      |--rw te-mpls:set-bandwidth?  
te-mpls-types:bandwidth-kbps  
|      |--rw te-mpls:class-type?  
te-types:te-ds-class  
|      |--ro te-mpls:state
```

```

| | | | | +--ro te-mpls:signaled-bandwidth?
te-mpls-types:bandwidth-kbps
| | | | | +--rw te-sr-mpls:sid-selection-mode?
te-sid-selection-mode
| | | | | +--rw te-sr-mpls:sid-protection? identityref
| | | | | +--rw te-dev:lsp-install-interval? uint32
| | | | | +--rw te-dev:lsp-cleanup-interval? uint32
| | | | | +--rw te-dev:lsp-invalidation-interval? uint32
+--rw tunnels
+--rw tunnel* [name]
| | | | | +--rw name string
| | | | | +--rw identifier? uint16
| | | | | +--rw description? string
| | | | | +--rw encoding? identityref
| | | | | +--rw switching-type? identityref
| | | | | +--rw provisioning-state? identityref
| | | | | +--rw preference? uint8
| | | | | +--rw reoptimize-timer? uint16
| | | | | +--rw source? inet:ip-address
| | | | | +--rw destination? inet:ip-address
| | | | | +--rw src-tp-id? binary
| | | | | +--rw dst-tp-id? binary
| | | | | +--rw protection
| | | | | | | +--rw enable? boolean
| | | | | | | +--rw protection-type? identityref
| | | | | | | +--rw protection-reversion-disable? boolean
| | | | | | | +--rw hold-off-time? uint32
| | | | | | | +--rw wait-to-revert? uint16
| | | | | | | +--rw aps-signal-id? uint8
+--rw restoration
| | | | | +--rw enable? boolean
| | | | | +--rw restoration-type? identityref
| | | | | +--rw restoration-scheme? identityref
| | | | | +--rw restoration-reversion-disable? boolean
| | | | | +--rw hold-off-time? uint32
| | | | | +--rw wait-to-restore? uint16
| | | | | +--rw wait-to-revert? uint16
+--rw te-topology-identifier
| | | | | +--rw provider-id? te-types:te-global-id
| | | | | +--rw client-id? te-types:te-global-id
| | | | | +--rw topology-id? te-types:te-topology-id
+--rw te-bandwidth
| | | | | +--rw (technology)?
| | | | | | | +--:(generic)
| | | | | | | +--rw generic? te-bandwidth
+--rw setup-priority? uint8
+--rw hold-priority? uint8
+--rw signaling-type? identityref

```

```

| | | +--rw dependency-tunnels
| | | | +--rw dependency-tunnel* [name]
| | | | | +--rw name ->
| | | | | ..../..../..../tunnels/tunnel/name
| | | | | +--rw encoding? identityref
| | | | | +--rw switching-type? identityref
| | | | +--ro state
| | | | | +--ro operational-state? identityref
| | | | | +--ro te-dev:lsp-install-interval? uint32
| | | | | +--ro te-dev:lsp-cleanup-interval? uint32
| | | | | +--ro te-dev:lsp-invalidation-interval? uint32
| | | | +--rw bidirectional
| | | | | +--rw association
| | | | | | +--rw id? uint16
| | | | | | +--rw source? inet:ip-address
| | | | | | +--rw global-source? inet:ip-address
| | | | | | +--rw type? identityref
| | | | | | +--rw provisioning? identityref
| | | | +--rw p2p-primary-paths
| | | | | +--rw p2p-primary-path* [name]
| | | | | | +--rw hierarchical-link
| | | | | | | +--rw local-te-node-id? te-types:te-node-id
| | | | | | | +--rw local-te-link-tp-id? te-types:te-tp-id
| | | | | | | +--rw remote-te-node-id? te-types:te-node-id
| | | | | | | +--rw te-topology-identifier
| | | | | | | | +--rw provider-id? te-types:te-global-id
| | | | | | | | +--rw client-id? te-types:te-global-id
| | | | | | | | +--rw topology-id? te-types:te-topology-id
| | | | | | +--rw name string
| | | | | | +--rw path-setup-protocol? identityref
| | | | | | +--rw path-computation-method? identityref
| | | | | | +--rw path-computation-server? inet:ip-address
| | | | | | +--rw compute-only? empty
| | | | | | +--rw use-path-computation? boolean
| | | | | | +--rw lockdown? empty
| | | | | | +--rw path-scope? identityref
| | | | | +--rw te-bandwidth
| | | | | | +--rw (technology)?
| | | | | | | +--:(generic)
| | | | | | | | +--rw generic? te-bandwidth
| | | | | +--rw setup-priority? uint8
| | | | | +--rw hold-priority? uint8
| | | | | +--rw signaling-type? identityref
| | | | | +--rw disjointness?
| | | | te-types:te-path-disjointness
| | | | +--rw path-metric-bounds
| | | | | +--rw path-metric-bound* [metric-type]
| | | | | | +--rw metric-type identityref

```

```

    +---rw upper-bound?      uint64
+---rw path-affinities
    +---rw constraints* [usage]
        +---rw usage          identityref
        +---rw (style)?
            +---:(value)
                +---rw value?

te-types:admin-groups
    +---:(named)
        +---rw affinity-names* [name]
            +---rw name        string

+---rw path-srlgs
    +---rw (style)?
        +---:(values)
            +---rw usage?      identityref
            +---rw values*     te-types:srlg
        +---:(named)
            +---rw constraints
                +---rw constraint* [usage]
                    +---rw usage          identityref
                    +---rw constraint
                        +---rw srlg-names* [name]
                            +---rw name        string

+---rw explicit-route-objects
    +---rw route-object-exclude-always* [index]
        +---rw index            uint32
        +---rw (type)?
            +---:(numbered)
                +---rw numbered-hop
                    +---rw address?      te-types:te-tp-id
                    +---rw hop-type?     te-hop-type
                    +---rw direction?    te-link-direction
            +---:(as-number)
                +---rw as-number-hop
                    +---rw as-number?    binary
                    +---rw hop-type?     te-hop-type
            +---:(unnumbered)
                +---rw unnumbered-hop
                    +---rw node-id?      te-types:te-node-id
                    +---rw link-tp-id?   te-types:te-tp-id
                    +---rw hop-type?     te-hop-type
                    +---rw direction?    te-link-direction
            +---:(label)
                +---rw label-hop
                    +---rw te-label
                        +---rw (technology)?
                            +---:(generic)
                                +---rw generic?

```

```

rt-types:generalized-label
|
|         +---rw direction?      te-label-direction
|     +---rw route-object-include-exclude* [index]
|         +---rw explicit-route-usage?  identityref
|         +---rw index                uint32
|         +---rw (type)?
|             +---:(numbered)
|                 +---rw numbered-hop
|                     +---rw address?      te-types:te-tp-id
|                     +---rw hop-type?     te-hop-type
|                     +---rw direction?    te-link-direction
|             +---:(as-number)
|                 +---rw as-number-hop
|                     +---rw as-number?    binary
|                     +---rw hop-type?     te-hop-type
|             +---:(unnumbered)
|                 +---rw unnumbered-hop
|                     +---rw node-id?      te-types:te-node-id
|                     +---rw link-tp-id?   te-types:te-tp-id
|                     +---rw hop-type?     te-hop-type
|                     +---rw direction?    te-link-direction
|             +---:(label)
|                 +---rw label-hop
|                     +---rw te-label
|                         +---rw (technology)?
|                             +---:(generic)
|                                 +---rw generic?

rt-types:generalized-label
|
|         +---rw direction?      te-label-direction
|     +---rw shared-resources-tunnels
|         +---rw lsp-shared-resources-tunnel*  te:tunnel-ref
|     +---rw path-in-segment!
|         +---rw forward
|             +---rw label-set* [inclusive-exclusive

label-start]
|
|         +---rw inclusive-exclusive  enumeration
|         +---rw label-start

rt-types:generalized-label
|
|         +---rw label-end?

rt-types:generalized-label
|
|         +---rw range-bitmap?      binary
|     +---rw reverse
|         +---rw label-set* [inclusive-exclusive

label-start]
|
|         +---rw inclusive-exclusive  enumeration
|         +---rw label-start

rt-types:generalized-label
|
|         +---rw label-end?

```

```
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw path-out-segment!
| | | | | +---rw forward
| | | | | +---rw label-set* [inclusive-exclusive
label-start]
| | | | | +---rw inclusive-exclusive enumeration
| | | | | +---rw label-start
rt-types:generalized-label
| | | | | +---rw label-end?
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw reverse
| | | | | +---rw label-set* [inclusive-exclusive
label-start]
| | | | | +---rw inclusive-exclusive enumeration
| | | | | +---rw label-start
rt-types:generalized-label
| | | | | +---rw label-end?
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw optimizations
| | | | | +---rw (algorithm)?
| | | | | +---:(metric) {path-optimization-metric}?
| | | | | +---rw optimization-metric* [metric-type]
| | | | | +---rw metric-type
identityref
| | | | | +---rw weight?
uint8
| | | | | +---rw explicit-route-exclude-objects
| | | | | +---rw route-object-exclude-object*
[index]
| | | | | +---rw index uint32
| | | | | +---rw (type)?
| | | | | +---:(numbered)
| | | | | | +---rw numbered-hop
| | | | | | +---rw address?
te-types:te-tp-id
| | | | | +---rw hop-type?
te-hop-type
| | | | | +---rw direction?
te-link-direction
| | | | | +---:(as-number)
| | | | | +---rw as-number-hop
| | | | | +---rw as-number? binary
| | | | | +---rw hop-type?
te-hop-type
| | | | | +---:(unnumbered)
```

```

| | | | | | +---rw unnumbered-hop
| | | | | | +---rw node-id?
te-types:te-node-id
| | | | | | +---rw link-tp-id?
te-types:te-tp-id
| | | | | | +---rw hop-type?
te-hop-type
| | | | | | +---rw direction?
te-link-direction
| | | | | | +---:(label)
| | | | | | +---rw label-hop
| | | | | | +---rw te-label
| | | | | | +---rw (technology)?
| | | | | | | +---:(generic)
| | | | | | | +---rw generic?
rt-types:generalized-label
| | | | | | +---rw direction?
te-label-direction
| | | | | | +---rw explicit-route-include-objects
| | | | | | +---rw route-object-include-object*
[index]
| | | | | | +---rw index uint32
| | | | | | +---rw (type)?
| | | | | | +---:(numbered)
| | | | | | | +---rw numbered-hop
| | | | | | | +---rw address?
te-types:te-tp-id
| | | | | | +---rw hop-type?
te-hop-type
| | | | | | +---rw direction?
te-link-direction
| | | | | | +---:(as-number)
| | | | | | | +---rw as-number-hop
| | | | | | | +---rw as-number? binary
| | | | | | | +---rw hop-type?
te-hop-type
| | | | | | +---:(unnumbered)
| | | | | | | +---rw unnumbered-hop
| | | | | | | +---rw node-id?
te-types:te-node-id
| | | | | | +---rw link-tp-id?
te-types:te-tp-id
| | | | | | +---rw hop-type?
te-hop-type
| | | | | | +---rw direction?
te-link-direction
| | | | | | +---:(label)
| | | | | | +---rw label-hop

```

```

+---rw te-label
+---rw (technology)?
|   +---:(generic)
|       +---rw generic?
rt-types:generalized-label
|
|       +---rw direction?
te-label-direction
|
|       +---rw tiebreakers
|           +---rw tiebreaker* [tiebreaker-type]
|               +---rw tiebreaker-type    identityref
|               +---:(objective-function)
{path-optimization-objective-function}?
|       +---rw objective-function
|       +---rw objective-function-type?
identityref
|
|       +---rw preference?                uint8
|       +---rw named-path-constraint?    ->
../../../../../../../../globals/named-path-constraints/named-path-constraint
/name
{te-types:named-path-constraints}?
|
|       +---ro state
|           +---ro path-properties
|               +---ro path-metric* [metric-type]
|                   +---ro metric-type    -> ../state/metric-type
|                   +---ro state
|                       +---ro metric-type?    identityref
|                       +---ro accumulative-value?    uint64
|               +---ro path-affinities
|                   +---ro constraints* [usage]
|                       +---ro usage                identityref
|                       +---ro (style)?
|                           +---:(value)
|                               | +---ro value?
te-types:admin-groups
|
|       +---:(named)
|           +---ro affinity-names* [name]
|               +---ro name    string
+---ro path-srlgs
+---ro (style)?
+---:(values)
|   +---ro usage?                identityref
|   +---ro values*                te-types:srlg
+---:(named)
+---ro constraints
+---ro constraint* [usage]
+---ro usage                identityref
+---ro constraint
+---ro srlg-names* [name]

```


[illegible]

```
uint16                                +--ro lsp-id
|                                     |
|                                     +--ro extended-tunnel-id
inet:ip-address                       +--ro operational-state?
identityref                          +--ro path-setup-protocol?
identityref                          +--ro origin-type?
enumeration                         +--ro lsp-resource-status?
enumeration                         +--ro lockout-of-normal?
boolean                             +--ro freeze?
boolean                             +--ro lsp-protection-role?
enumeration                         +--ro lsp-protection-state?
identityref                        +--ro protection-group-ingress-node-id?
te-types:te-node-id                +--ro protection-group-egress-node-id?
te-types:te-node-id                +--ro lsp-shared-resources-tunnel?
te:tunnel-ref                      +--ro lsp-record-route-subobjects
|                                   +--ro record-route-subobject* [index]
|                                   +--ro index          uint32
|                                   +--ro (type)?
|                                   +---:(numbered)
|                                   |   +--ro address?
te-types:te-tp-id                  |   +--ro ip-flags?      binary
|                                   +---:(unnumbered)
|                                   |   +--ro node-id?
te-types:te-node-id               |   +--ro link-tp-id?
te-types:te-tp-id                 +---:(label)
|                                   +--ro value?
rt-types:generalized-label         +--ro label-flags?    binary
|                                   +--ro path-properties
|                                   +--ro path-metric* [metric-type]
|                                   +--ro metric-type     ->
../state/metric-type              +--ro state
```

identityref	<pre> +--ro metric-type? +--ro accumulative-value? uint64 +--ro path-affinities +--ro constraints* [usage] +--ro usage identityref +--ro (style)? +--:(value) +--ro value? </pre>
te-types:admin-groups	<pre> +--:(named) +--ro affinity-names* [name] +--ro name string +--ro path-srlgs +--ro (style)? +--:(values) +--ro usage? identityref +--ro values* te-types:srlg +--:(named) +--ro constraints +--ro constraint* [usage] +--ro usage </pre>
identityref	<pre> +--ro constraint +--ro srlg-names* [name] +--ro name string +--ro path-route-objects +--ro path-computed-route-object* </pre>
[index]	<pre> +--ro index -> ../state/index +--ro state +--ro index? uint32 +--ro (type)? +--:(numbered) +--ro numbered-hop +--ro address? </pre>
te-types:te-tp-id	<pre> +--ro hop-type? </pre>
te-hop-type	<pre> +--ro direction? </pre>
te-link-direction	<pre> +--:(as-number) +--ro as-number-hop +--ro as-number? binary +--ro hop-type? </pre>
te-hop-type	<pre> +--:(unnumbered) +--ro unnumbered-hop </pre>

```

| | | | | | | +---ro node-id?
te-types:te-node-id      | | | | | 
| | | | | | | +---ro link-tp-id?
te-types:te-tp-id       | | | | | 
| | | | | | | +---ro hop-type?
te-hop-type             | | | | | 
| | | | | | | +---ro direction?
te-link-direction        | | | | | 
| | | | | | | +--:(label)
| | | | | | |   |--ro label-hop
| | | | | | |     |--ro te-label
| | | | | | |       |--ro (technology)?
| | | | | | |         |--:(generic)
| | | | | | |           |--ro generic?
rt-types:generalized-label | | | | | 
| | | | | | | +---ro direction?
te-label-direction        | | | | | 
| | | | | | |   +---ro shared-resources-tunnels
| | | | | | |     +---ro lsp-shared-resources-tunnel*
te:tunnel-ref            | | | | | 
| | | | | | |   +---ro te-dev:lsp-timers
| | | | | | |     +---ro te-dev:life-time?          uint32
| | | | | | |       +---ro te-dev:time-to-install?    uint32
| | | | | | |         +---ro te-dev:time-to-destroy?  uint32
| | | | | | |       +---ro te-dev:downstream-info
| | | | | | |         +---ro te-dev:nhop?
inet:ip-address          | | | | | 
| | | | | | |   +---ro te-dev:outgoing-interface?
if:interface-ref         | | | | | 
| | | | | | |   +---ro te-dev:neighbor?
inet:ip-address          | | | | | 
| | | | | | |     +---ro te-dev:label?
rt-types:generalized-label | | | | | 
| | | | | | |       +---ro te-dev:upstream-info
| | | | | | |         +---ro te-dev:phop?          inet:ip-address
| | | | | | |           +---ro te-dev:neighbor?    inet:ip-address
| | | | | | |             +---ro te-dev:label?
rt-types:generalized-label | | | | | 
| | | | | | |       +---ro te-mpls:static-lsp-name?
mpls-static:static-lsp-ref | | | | | 
| | | | | | |       +---rw p2p-reverse-primary-path
| | | | | | |         +---rw name?                  string
| | | | | | |           +---rw path-setup-protocol? identityref
| | | | | | |             +---rw path-computation-method? identityref
| | | | | | |               +---rw path-computation-server? inet:ip-address
| | | | | | |                 +---rw compute-only?      empty
| | | | | | |                   +---rw use-path-computation? boolean
| | | | | | |                     +---rw lockdown?       empty

```

```

+--rw path-scope?                                identityref
+--rw te-bandwidth
|   +--rw (technology)?
|       +--:(generic)
|           +--rw generic?    te-bandwidth
+--rw setup-priority?                            uint8
+--rw hold-priority?                             uint8
+--rw signaling-type?                            identityref
+--rw disjointness?
te-types:te-path-disjointness
+--rw path-metric-bounds
|   +--rw path-metric-bound* [metric-type]
|       +--rw metric-type    identityref
|       +--rw upper-bound?   uint64
+--rw path-affinities
|   +--rw constraints* [usage]
|       +--rw usage          identityref
|       +--rw (style)?
|           +--:(value)
|               | +--rw value?
te-types:admin-groups
|   +--:(named)
|       +--rw affinity-names* [name]
|           +--rw name        string
+--rw path-srlgs
|   +--rw (style)?
|       +--:(values)
|           | +--rw usage?      identityref
|           | +--rw values*     te-types:srlg
|       +--:(named)
|           +--rw constraints
|               +--rw constraint* [usage]
|                   +--rw usage    identityref
|                   +--rw constraint
|                       +--rw srlg-names* [name]
|                           +--rw name    string
+--rw explicit-route-objects
|   +--rw route-object-exclude-always* [index]
|       +--rw index                    uint32
|       +--rw (type)?
|           +--:(numbered)
|               | +--rw numbered-hop
|               |     +--rw address?    te-types:te-tp-id
|               |     +--rw hop-type?   te-hop-type
|               |     +--rw direction?  te-link-direction
|               +--:(as-number)
|                   +--rw as-number-hop
|                       +--rw as-number? binary

```

```

|         +---rw hop-type?      te-hop-type
|         +---:(unnumbered)
|         |         +---rw unnumbered-hop
|         |         +---rw node-id?
te-types:te-node-id
|
|         +---rw link-tp-id?    te-types:te-tp-id
|         +---rw hop-type?      te-hop-type
|         +---rw direction?     te-link-direction
+---:(label)
|         +---rw label-hop
|         +---rw te-label
|         |         +---rw (technology)?
|         |         |         +---:(generic)
|         |         |         +---rw generic?
rt-types:generalized-label
|         +---rw direction?
te-label-direction
|         +---rw route-object-include-exclude* [index]
|         +---rw explicit-route-usage?  identityref
|         +---rw index                    uint32
|         +---rw (type)?
|         +---:(numbered)
|         |         +---rw numbered-hop
|         |         |         +---rw address?      te-types:te-tp-id
|         |         |         +---rw hop-type?      te-hop-type
|         |         |         +---rw direction?     te-link-direction
|         +---:(as-number)
|         |         +---rw as-number-hop
|         |         |         +---rw as-number?    binary
|         |         |         +---rw hop-type?      te-hop-type
|         +---:(unnumbered)
|         |         +---rw unnumbered-hop
|         |         +---rw node-id?
te-types:te-node-id
|
|         +---rw link-tp-id?    te-types:te-tp-id
|         +---rw hop-type?      te-hop-type
|         +---rw direction?     te-link-direction
+---:(label)
|         +---rw label-hop
|         +---rw te-label
|         |         +---rw (technology)?
|         |         |         +---:(generic)
|         |         |         +---rw generic?
rt-types:generalized-label
|         +---rw direction?
te-label-direction
|         +---rw shared-resources-tunnels
|         +---rw lsp-shared-resources-tunnel*

```

```
te:tunnel-ref
| | | | | +---rw path-in-segment!
| | | | | +---rw forward
| | | | | +---rw label-set* [inclusive-exclusive
label-start]
| | | | | +---rw inclusive-exclusive enumeration
| | | | | +---rw label-start
rt-types:generalized-label
| | | | | +---rw label-end?
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw reverse
| | | | | +---rw label-set* [inclusive-exclusive
label-start]
| | | | | +---rw inclusive-exclusive enumeration
| | | | | +---rw label-start
rt-types:generalized-label
| | | | | +---rw label-end?
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw path-out-segment!
| | | | | +---rw forward
| | | | | +---rw label-set* [inclusive-exclusive
label-start]
| | | | | +---rw inclusive-exclusive enumeration
| | | | | +---rw label-start
rt-types:generalized-label
| | | | | +---rw label-end?
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw reverse
| | | | | +---rw label-set* [inclusive-exclusive
label-start]
| | | | | +---rw inclusive-exclusive enumeration
| | | | | +---rw label-start
rt-types:generalized-label
| | | | | +---rw label-end?
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw optimizations
| | | | | +---rw (algorithm)?
| | | | | +---:(metric) {path-optimization-metric}?
| | | | | +---rw optimization-metric* [metric-type]
| | | | | +---rw metric-type
identityref
| | | | | +---rw weight?
uint8
| | | | | +---rw explicit-route-exclude-objects
```

[illegible]

[illegible]

```

+--ro (style)?
+--:(value)
|   +--ro value?
te-types:admin-groups
+--:(named)
+--ro affinity-names* [name]
+--ro name          string
+--ro path-srlgs
+--ro (style)?
+--:(values)
|   +--ro usage?          identityref
|   +--ro values*         te-types:srlg
+--:(named)
+--ro constraints
+--ro constraint* [usage]
+--ro usage          identityref
+--ro constraint
+--ro srlg-names* [name]
+--ro name          string
+--ro path-route-objects
+--ro path-computed-route-object* [index]
+--ro index          -> ../state/index
+--ro state
+--ro index?          uint32
+--ro (type)?
+--:(numbered)
|   +--ro numbered-hop
|   +--ro address?
|   +--ro hop-type?
|   +--ro direction?
+--:(as-number)
|   +--ro as-number-hop
|   +--ro as-number?    binary
|   +--ro hop-type?
+--:(unnumbered)
|   +--ro unnumbered-hop
|   +--ro node-id?
|   +--ro link-tp-id?
|   +--ro hop-type?
|   +--ro direction?

```

[illegible]

						++-ro record-route-subobject* [index]
						++-ro index uint32
						++-ro (type)?
						++:(numbered)
						++-ro address?
						++-ro ip-flags? binary
						++:(unnumbered)
						++-ro node-id?
						++-ro link-tp-id?
						++:(label)
						++-ro value?
						++-ro label-flags? binary
						++-ro path-properties
						++-ro path-metric* [metric-type]
						++-ro metric-type ->
						++-ro state
						++-ro metric-type?
						++-ro accumulative-value? uint64
						++-ro path-affinities
						++-ro constraints* [usage]
						++-ro usage
						++-ro (style)?
						++:(value)
						++-ro value?
						++:(named)
						++-ro affinity-names* [name]
						++-ro name string
						++-ro path-srlgs
						++-ro (style)?
						++:(values)
						++-ro usage?
						++-ro values*
						++:(named)
						++-ro constraints
						++-ro constraint* [usage]
						++-ro usage
						++-ro constraint

					+++ro srlg-names*
					+++ro name
					+++ro path-route-objects
					+++ro path-computed-route-object*
					+++ro index -> ../state/index
					+++ro state
					+++ro index? uint32
					+++ro (type)?
					+++:(numbered)
					+++ro numbered-hop
					+++ro address?
					+++ro hop-type?
					+++ro direction?
					+++:(as-number)
					+++ro as-number-hop
					+++ro as-number?
					+++ro hop-type?
					+++:(unnumbered)
					+++ro unnumbered-hop
					+++ro node-id?
					+++ro link-tp-id?
					+++ro hop-type?
					+++ro direction?
					+++:(label)
					+++ro label-hop
					+++ro te-label
					+++ro (technology)?
					+++:(generic)
					+++ro
					+++ro direction?
					+++ro shared-resources-tunnels
					+++ro lsp-shared-resources-tunnel*
					+++rw p2p-reverse-secondary-path

```

| | | | |      +--rw secondary-path?      ->
| | | | |      |      ..../..../..../p2p-secondary-paths/p2p-secondary-path/name
| | | | |      |      +--rw path-setup-protocol?  identityref
| | | | |      |      +--rw candidate-p2p-secondary-paths
| | | | |      |      |      +--rw candidate-p2p-secondary-path* [secondary-path]
| | | | |      |      |      +--rw secondary-path      ->
| | | | |      |      |      ..../..../..../p2p-secondary-paths/p2p-secondary-path/name
| | | | |      |      |      +--rw path-setup-protocol?  identityref
| | | | |      |      |      +--ro state
| | | | |      |      |      |      +--ro active?  boolean
| | | | |      |      |      +--rw te-mpls:static-lsp-name?
| | | | |      |      +--rw te-mpls:static-lsp-ref
| | | | |      |      |      +--rw p2p-secondary-paths
| | | | |      |      |      |      +--rw p2p-secondary-path* [name]
| | | | |      |      |      |      |      +--rw name                      string
| | | | |      |      |      |      |      +--rw path-setup-protocol?      identityref
| | | | |      |      |      |      |      +--rw path-computation-method?   identityref
| | | | |      |      |      |      |      +--rw path-computation-server?   inet:ip-address
| | | | |      |      |      |      |      +--rw compute-only?              empty
| | | | |      |      |      |      |      +--rw use-path-computation?       boolean
| | | | |      |      |      |      |      +--rw lockdown?                  empty
| | | | |      |      |      |      |      +--rw path-scope?                 identityref
| | | | |      |      |      |      |      +--rw te-bandwidth
| | | | |      |      |      |      |      |      +--rw (technology)?
| | | | |      |      |      |      |      |      |      +--:(generic)
| | | | |      |      |      |      |      |      |      |      +--rw generic?  te-bandwidth
| | | | |      |      |      |      |      +--rw setup-priority?             uint8
| | | | |      |      |      |      |      +--rw hold-priority?              uint8
| | | | |      |      |      |      |      +--rw signaling-type?             identityref
| | | | |      |      |      |      |      +--rw disjointness?
| | | | |      |      +--rw te-types:te-path-disjointness
| | | | |      |      |      +--rw path-metric-bounds
| | | | |      |      |      |      +--rw path-metric-bound* [metric-type]
| | | | |      |      |      |      |      +--rw metric-type      identityref
| | | | |      |      |      |      |      +--rw upper-bound?     uint64
| | | | |      |      |      +--rw path-affinities
| | | | |      |      |      |      +--rw constraints* [usage]
| | | | |      |      |      |      |      +--rw usage              identityref
| | | | |      |      |      |      |      +--rw (style)?
| | | | |      |      |      |      |      |      +--:(value)
| | | | |      |      |      |      |      |      |      +--rw value?
| | | | |      |      +--rw te-types:admin-groups
| | | | |      |      |      +--:(named)
| | | | |      |      |      |      +--rw affinity-names* [name]
| | | | |      |      |      |      |      +--rw name      string
| | | | |      |      |      +--rw path-srlgs
| | | | |      |      |      |      +--rw (style)?
| | | | |      |      |      |      |      +--:(values)

```

[illegible]

[illegible]


```
label-start]
| | | | | +---rw inclusive-exclusive enumeration
| | | | | +---rw label-start
rt-types:generalized-label
| | | | | +---rw label-end?
rt-types:generalized-label
| | | | | +---rw range-bitmap? binary
| | | | | +---rw optimizations
| | | | | +---rw (algorithm)?
| | | | | +---:(metric) {path-optimization-metric}?
| | | | | +---rw optimization-metric* [metric-type]
| | | | | +---rw metric-type
identityref
| | | | | +---rw weight?
uint8
| | | | | +---rw explicit-route-exclude-objects
| | | | | +---rw route-object-exclude-object*
[index]
| | | | | +---rw index uint32
| | | | | +---rw (type)?
| | | | | +---:(numbered)
| | | | | | +---rw numbered-hop
| | | | | | +---rw address?
te-types:te-tp-id
| | | | | | +---rw hop-type?
te-hop-type
| | | | | | +---rw direction?
te-link-direction
| | | | | +---:(as-number)
| | | | | | +---rw as-number-hop
| | | | | | +---rw as-number? binary
| | | | | | +---rw hop-type?
te-hop-type
| | | | | +---:(unnumbered)
| | | | | | +---rw unnumbered-hop
| | | | | | +---rw node-id?
te-types:te-node-id
| | | | | | +---rw link-tp-id?
te-types:te-tp-id
| | | | | | +---rw hop-type?
te-hop-type
| | | | | | +---rw direction?
te-link-direction
| | | | | +---:(label)
| | | | | | +---rw label-hop
| | | | | | +---rw te-label
| | | | | | +---rw (technology)?
| | | | | | +---:(generic)
```

[illegible]

```

identityref
| | | | | +--rw preference?                uint8
| | | | | +--rw named-path-constraint?    ->
| | | | | ../../../../../../globals/named-path-constraints/
named-path-constraint/name
{te-types:named-path-constraints}?
| | | | | +--rw protection
| | | | | | +--rw enable?                    boolean
| | | | | | +--rw protection-type?          identityref
| | | | | | +--rw protection-reversion-disable? boolean
| | | | | | +--rw hold-off-time?            uint32
| | | | | | +--rw wait-to-revert?           uint16
| | | | | | +--rw aps-signal-id?            uint8
| | | | | +--rw restoration
| | | | | | +--rw enable?                    boolean
| | | | | | +--rw restoration-type?          identityref
| | | | | | +--rw restoration-scheme?        identityref
| | | | | | +--rw restoration-reversion-disable? boolean
| | | | | | +--rw hold-off-time?            uint32
| | | | | | +--rw wait-to-restore?          uint16
| | | | | | +--rw wait-to-revert?          uint16
| | | | | +--ro state
| | | | | | +--ro path-properties
| | | | | | | +--ro path-metric* [metric-type]
| | | | | | | | +--ro metric-type    -> ../state/metric-type
| | | | | | | | +--ro state
| | | | | | | | | +--ro metric-type?    identityref
| | | | | | | | | +--ro accumulative-value? uint64
| | | | | | | +--ro path-affinities
| | | | | | | | +--ro constraints* [usage]
| | | | | | | | | +--ro usage            identityref
| | | | | | | | | +--ro (style)?
| | | | | | | | | | +--:(value)
| | | | | | | | | | | +--ro value?
te-types:admin-groups
| | | | | | +--:(named)
| | | | | | | +--ro affinity-names* [name]
| | | | | | | | +--ro name            string
| | | | | | +--ro path-srlgs
| | | | | | | +--ro (style)?
| | | | | | | | +--:(values)
| | | | | | | | | +--ro usage?          identityref
| | | | | | | | | +--ro values*        te-types:srlg
| | | | | | | | +--:(named)
| | | | | | | | +--ro constraints
| | | | | | | | | +--ro constraint* [usage]
| | | | | | | | | | +--ro usage            identityref
| | | | | | | | | +--ro constraint

```

	+--ro srlg-names* [name]
	+--ro name string
	+--ro path-route-objects
	+--ro path-computed-route-object* [index]
	+--ro index -> ../state/index
	+--ro state
	+--ro index? uint32
	+--ro (type)?
	+--:(numbered)
	+--ro numbered-hop
	+--ro address?
te-types:te-tp-id	
	+--ro hop-type? te-hop-type
	+--ro direction?
te-link-direction	
	+--:(as-number)
	+--ro as-number-hop
	+--ro as-number? binary
	+--ro hop-type? te-hop-type
	+--:(unnumbered)
	+--ro unnumbered-hop
	+--ro node-id?
te-types:te-node-id	
	+--ro link-tp-id?
te-types:te-tp-id	
	+--ro hop-type? te-hop-type
	+--ro direction?
te-link-direction	
	+--:(label)
	+--ro label-hop
	+--ro te-label
	+--ro (technology)?
	+--:(generic)
	+--ro generic?
rt-types:generalized-label	
	+--ro direction?
te-label-direction	
	+--ro shared-resources-tunnels
	+--ro lsp-shared-resources-tunnel*
te:tunnel-ref	
	+--ro lsps
	+--ro lsp* [source destination tunnel-id lsp-id
extended-tunnel-id]	
	+--ro source
inet:ip-address	
	+--ro destination
inet:ip-address	
	+--ro tunnel-id

```
uint16 | | | | | +--ro lsp-id
uint16 | | | | | +--ro extended-tunnel-id
inet:ip-address | | | | | +--ro operational-state?
identityref | | | | | +--ro path-setup-protocol?
identityref | | | | | +--ro origin-type?
enumeration | | | | | +--ro lsp-resource-status?
enumeration | | | | | +--ro lockout-of-normal?
boolean | | | | | +--ro freeze?
boolean | | | | | +--ro lsp-protection-role?
enumeration | | | | | +--ro lsp-protection-state?
identityref | | | | | +--ro protection-group-ingress-node-id?
te-types:te-node-id | | | | | +--ro protection-group-egress-node-id?
te-types:te-node-id | | | | | +--ro lsp-shared-resources-tunnel?
te:tunnel-ref | | | | | +--ro lsp-record-route-subobjects
| | | | | | +--ro record-route-subobject* [index]
| | | | | | | +--ro index uint32
| | | | | | | +--ro (type)?
| | | | | | | | +--:(numbered)
| | | | | | | | | +--ro address?
te-types:te-tp-id | | | | | | | +--ro ip-flags? binary
| | | | | | | +--:(unnumbered)
| | | | | | | | +--ro node-id?
te-types:te-node-id | | | | | | | +--ro link-tp-id?
te-types:te-tp-id | | | | | | | +--:(label)
| | | | | | | | +--ro value?
rt-types:generalized-label | | | | | | | +--ro label-flags? binary
+--ro path-properties
| +--ro path-metric* [metric-type]
| | +--ro metric-type ->
../state/metric-type
```

	+--ro state
	+--ro metric-type?
identityref	
	+--ro accumulative-value? uint64
	+--ro path-affinities
	+--ro constraints* [usage]
	+--ro usage identityref
	+--ro (style)?
	+--:(value)
	+--ro value?
te-types:admin-groups	
	+--:(named)
	+--ro affinity-names* [name]
	+--ro name string
	+--ro path-srlgs
	+--ro (style)?
	+--:(values)
	+--ro usage? identityref
	+--ro values* te-types:srlg
	+--:(named)
	+--ro constraints
	+--ro constraint* [usage]
	+--ro usage
identityref	
	+--ro constraint
	+--ro srlg-names* [name]
	+--ro name string
	+--ro path-route-objects
	+--ro path-computed-route-object*
[index]	
	+--ro index -> ../state/index
	+--ro state
	+--ro index? uint32
	+--ro (type)?
	+--:(numbered)
	+--ro numbered-hop
	+--ro address?
te-types:te-tp-id	
	+--ro hop-type?
te-hop-type	
	+--ro direction?
te-link-direction	
	+--:(as-number)
	+--ro as-number-hop
	+--ro as-number? binary
	+--ro hop-type?
te-hop-type	
	+--:(unnumbered)

```

| | | | | | | | | +---ro unnumbered-hop
| | | | | | | | |   +---ro node-id?
te-types:te-node-id
| | | | | | | | |   +---ro link-tp-id?
te-types:te-tp-id
| | | | | | | | |   +---ro hop-type?
te-hop-type
| | | | | | | | |   +---ro direction?
te-link-direction
| | | | | | | | |     +---:(label)
| | | | | | | | |       +--ro label-hop
| | | | | | | | |         +--ro te-label
| | | | | | | | |           +--ro (technology)?
| | | | | | | | |             |--:(generic)
| | | | | | | | |               +--ro generic?
rt-types:generalized-label
| | | | | | | | |     +--ro direction?
te-label-direction
| | | | | | | | |       +---ro shared-resources-tunnels
| | | | | | | | |       +---ro lsp-shared-resources-tunnel*
te:tunnel-ref
| | | | | | | | |     +---ro te-dev:lsp-timers
| | | | | | | | |       +---ro te-dev:life-time?          uint32
| | | | | | | | |       +---ro te-dev:time-to-install?      uint32
| | | | | | | | |       +---ro te-dev:time-to-destroy?      uint32
+---ro te-dev:downstream-info
| | | | | | | | |       +---ro te-dev:nhop?
inet:ip-address
| | | | | | | | |     +---ro te-dev:outgoing-interface?
if:interface-ref
| | | | | | | | |     +---ro te-dev:neighbor?
inet:ip-address
| | | | | | | | |     +---ro te-dev:label?
rt-types:generalized-label
| | | | | | | | |       +---ro te-dev:upstream-info
| | | | | | | | |         +---ro te-dev:phop?              inet:ip-address
| | | | | | | | |         +---ro te-dev:neighbor?          inet:ip-address
| | | | | | | | |         +---ro te-dev:label?
rt-types:generalized-label
| | | | | | | | |       +---ro te-mpls:static-lsp-name?
mpls-static:static-lsp-ref
| | | | | | | | |       +---rw te-mpls:static-lsp-name?
mpls-static:static-lsp-ref
| | | | | | | | |     +---x tunnel-action
| | | | | | | | |       +---w input
| | | | | | | | |         | +---w action-type?    identityref
| | | | | | | | |       +---ro output
| | | | | | | | |         +---ro action-result?    identityref
```

```

| | | +---x protection-external-commands
| | | | +---w input
| | | | | +---w protection-external-command? identityref
| | | | | +---w protection-group-ingress-node-id?
te-types:te-node-id
| | | | +---w protection-group-egress-node-id?
te-types:te-node-id
| | | | | +---w path-ref? path-ref
| | | | | +---w traffic-type? enumeration
| | | | | +---w extra-traffic-tunnel-ref? te:tunnel-ref
| | | +--rw te-dev:lsp-install-interval? uint32
| | | +--rw te-dev:lsp-cleanup-interval? uint32
| | | +--rw te-dev:lsp-invalidation-interval? uint32
| | | +--rw te-mpls:tunnel-igp-shortcut
| | | | +--rw te-mpls:shortcut-eligible? boolean
| | | | +--rw te-mpls:metric-type? identityref
| | | | +--rw te-mpls:metric? int32
| | | | +--rw te-mpls:routing-afs* inet:ip-version
| | | +--rw te-mpls:forwarding
| | | | +--rw te-mpls:binding-label? rt-types:mpls-label
| | | | +--rw te-mpls:load-share? uint32
| | | | +--rw te-mpls:policy-class? uint8
| | | +--rw te-mpls:bandwidth-mpls
| | | | +--rw te-mpls:specification-type?
te-mpls-types:te-bandwidth-type
| | | | +--rw te-mpls:set-bandwidth?
te-mpls-types:bandwidth-kbps
| | | | +--rw te-mpls:class-type? te-types:te-ds-class
| | | | +--ro te-mpls:state
| | | | | +--ro te-mpls:signaled-bandwidth?
te-mpls-types:bandwidth-kbps
| | | | +--rw te-mpls:auto-bandwidth
| | | | +--rw te-mpls:enabled? boolean
| | | | +--rw te-mpls:min-bw?
te-mpls-types:bandwidth-kbps
| | | | +--rw te-mpls:max-bw?
te-mpls-types:bandwidth-kbps
| | | | +--rw te-mpls:adjust-interval? uint32
| | | | +--rw te-mpls:adjust-threshold? te-types:percentage
| | | | +--rw te-mpls:overflow
| | | | | +--rw te-mpls:enabled? boolean
| | | | | +--rw te-mpls:overflow-threshold?
te-types:percentage
| | | | +--rw te-mpls:trigger-event-count? uint16
| | | | +--rw te-mpls:underflow
| | | | +--rw te-mpls:enabled? boolean
| | | | +--rw te-mpls:underflow-threshold?
te-types:percentage

```



```

|           +--rw te-mpls:trigger-event-count?  uint16
| +--rw tunnel-p2mp* [name]
|   +--rw name          string
|   +--rw identifier?    uint16
|   +--rw description?   string
|   +--ro state
|       +--ro operational-state?  identityref
+--ro lsp-state
|   +--ro lsp* [source destination tunnel-id lsp-id
extended-tunnel-id]
|       +--ro source          inet:ip-address
|       +--ro destination     inet:ip-address
|       +--ro tunnel-id       uint16
|       +--ro lsp-id          uint16
|       +--ro extended-tunnel-id  inet:ip-address
|       +--ro operational-state? identityref
|       +--ro path-setup-protocol? identityref
|       +--ro origin-type?     enumeration
|       +--ro lsp-resource-status? enumeration
|       +--ro lockout-of-normal? boolean
|       +--ro freeze?         boolean
|       +--ro lsp-protection-role? enumeration
|       +--ro lsp-protection-state? identityref
|       +--ro protection-group-ingress-node-id? te-types:te-node-id
|       +--ro protection-group-egress-node-id? te-types:te-node-id
|       +--ro lsp-record-route-subobjects
|       |   +--ro record-route-subobject* [index]
|       |   |   +--ro index          uint32
|       |   |   +--ro (type)?
|       |   |       +--:(numbered)
|       |   |       |   +--ro address?      te-types:te-tp-id
|       |   |       |   +--ro ip-flags?     binary
|       |   |       +--:(unnumbered)
|       |   |       |   +--ro node-id?      te-types:te-node-id
|       |   |       |   +--ro link-tp-id?   te-types:te-tp-id
|       |   |       +--:(label)
|       |   |       |   +--ro value?        rt-types:generalized-label
|       |   |       |   +--ro label-flags?  binary
|       |   +--ro te-dev:lsp-timers
|       |   |   +--ro te-dev:life-time?      uint32
|       |   |   +--ro te-dev:time-to-install? uint32
|       |   |   +--ro te-dev:time-to-destroy? uint32
|       |   +--ro te-dev:downstream-info
|       |   |   +--ro te-dev:nhop?          inet:ip-address
|       |   |   +--ro te-dev:outgoing-interface? if:interface-ref
|       |   |   +--ro te-dev:neighbor?      inet:ip-address
|       |   +--ro te-dev:label?
rt-types:generalized-label

```

```

|      +---ro te-dev:upstream-info
|      |      +---ro te-dev:phop?          inet:ip-address
|      |      +---ro te-dev:neighbor?      inet:ip-address
|      |      +---ro te-dev:label?         rt-types:generalized-label
+---rw te-dev:interfaces
|   +---rw te-dev:threshold-type?          enumeration
|   +---rw te-dev:delta-percentage?         te-types:percentage
|   +---rw te-dev:threshold-specification?  enumeration
|   +---rw te-dev:up-thresholds*           te-types:percentage
|   +---rw te-dev:down-thresholds*         te-types:percentage
|   +---rw te-dev:up-down-thresholds*      te-types:percentage
+---rw te-dev:interface* [interface]
|   +---rw te-dev:interface                if:interface-ref
|   +---rw te-dev:te-metric?               te-types:te-metric
+---rw (te-dev:admin-group-type)?
|   +---:(te-dev:value-admin-groups)
|   |   +---rw (te-dev:value-admin-group-type)?
|   |   |   +---:(te-dev:admin-groups)
|   |   |   |   +---rw te-dev:admin-group?
|   |   |   |   te-types:admin-group
|   |   |   |   +---:(te-dev:extended-admin-groups)
|   |   |   |   {te-types:extended-admin-groups}?
|   |   |   |   +---rw te-dev:extended-admin-group?
|   |   |   |   te-types:extended-admin-group
|   |   |   |   +---:(te-dev:named-admin-groups)
|   |   |   |   +---rw te-dev:named-admin-groups* [named-admin-group]
|   |   |   |   {te-types:extended-admin-groups,te-types:
|   |   |   |   named-extended-admin-groups}?
|   |   |   |   +---rw te-dev:named-admin-group    ->
|   |   |   |   ../../../../te:globals/named-admin-groups/named-admin-group/
|   |   |   |   name
|   |   |   |   +---rw (te-dev:srlg-type)?
|   |   |   |   |   +---:(te-dev:value-srlgs)
|   |   |   |   |   |   +---rw te-dev:values* [value]
|   |   |   |   |   |   +---rw te-dev:value      uint32
|   |   |   |   |   +---:(te-dev:named-srlgs)
|   |   |   |   |   +---rw te-dev:named-srlgs* [named-srlg]
|   |   |   |   |   {te-types:named-srlg-groups}?
|   |   |   |   |   +---rw te-dev:named-srlg      ->
|   |   |   |   |   ../../../../te:globals/named-srlgs/named-srlg/name
|   |   |   |   +---rw te-dev:threshold-type?      enumeration
|   |   |   |   +---rw te-dev:delta-percentage?     te-types:percentage
|   |   |   |   +---rw te-dev:threshold-specification? enumeration
|   |   |   |   +---rw te-dev:up-thresholds*        te-types:percentage
|   |   |   |   +---rw te-dev:down-thresholds*      te-types:percentage
|   |   |   |   +---rw te-dev:up-down-thresholds*   te-types:percentage
|   |   |   |   +---rw te-dev:switching-capabilities* [switching-capability]
|   |   |   |   |   +---rw te-dev:switching-capability  identityref

```

```

    |   +--rw te-dev:encoding?                identityref
  +--ro te-dev:state
    +--ro te-dev:te-advertisements_state
      +--ro te-dev:flood-interval?            uint32
      +--ro te-dev:last-flooded-time?         uint32
      +--ro te-dev:next-flooded-time?         uint32
      +--ro te-dev:last-flooded-trigger?      enumeration
      +--ro te-dev:advertized-level-areas* [level-area]
        +--ro te-dev:level-area              uint32

rpcs:
  +---x globals-rpc
  +---x interfaces-rpc
  +---x tunnels-rpc
    +---w input
      +---w tunnel-info
        +---w (type)?
          +---:(tunnel-p2p)
            |   +---w p2p-id?      te:tunnel-ref
          +---:(tunnel-p2mp)
            |   +---w p2mp-id?    te:tunnel-p2mp-ref
      +--ro output
        +--ro result
          +--ro result?    enumeration

notifications:
  +---n globals-notif
  +---n tunnels-notif
module: ietf-te-device

rpcs:
  +---x interfaces-rpc

notifications:
  +---n interfaces-notif

```

Figure 6: TE generic model configuration and state tree

4. TE Generic and Helper YANG Modules

```

<CODE BEGINS> file "ietf-te-types@2018-03-05.yang"
module ietf-te-types {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te-types";

  /* Replace with IANA when assigned */
  prefix "te-types";

```

```
import ietf-inet-types {
  prefix inet;
}

import ietf-yang-types {
  prefix "yang";
}

import ietf-routing-types {
  prefix "rt-types";
}

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>

  WG Chair:   Lou Berger
              <mailto:lberger@labn.net>

  WG Chair:   Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Tarek Saad
              <mailto:tsaad@cisco.com>

  Editor:     Rakesh Gandhi
              <mailto:rgandhi@cisco.com>

  Editor:     Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Himanshu Shah
              <mailto:hshah@ciena.com>

  Editor:     Xufeng Liu
              <mailto:xufeng.liu@ericsson.com>

  Editor:     Igor Bryskin
              <mailto:Igor.Bryskin@huawei.com>";

description
  "This module contains a collection of generally
  useful TE specific YANG data type definitions.";
```

```
revision "2018-03-05" {
  description "Latest revision of TE types";
  reference "RFC3209";
}

/*
 * Identities
 */
identity objective-function-type {
  description "Base objective function type";
  reference "RFC4657";
}
identity of-minimize-cost-path {
  base objective-function-type;
  description
    "Minimize cost of path objective function";
}
identity of-minimize-load-path {
  base objective-function-type;
  description
    "Minimize the load on path(s) objective
    function";
}
identity of-maximize-residual-bandwidth {
  base objective-function-type;
  description
    "Maximize the residual bandwidth objective
    function";
}
identity of-minimize-aggregate-bandwidth-consumption {
  base objective-function-type;
  description
    "minimize the aggregate bandwidth consumption
    objective function";
}
identity of-minimize-load-most-loaded-link {
  base objective-function-type;
  description
    "Minimize the load on the most loaded link
    objective function";
}
identity of-minimize-cost-path-set {
  base objective-function-type;
  description
    "Minimize the cost on a path set objective
    function";
}
```

```

identity path-computation-method {
  description
    "base identity for supported path computation
    mechanisms";
}

identity path-locally-computed {
  base path-computation-method;
  description
    "indicates a constrained-path LSP in which the
    path is computed by the local LER";
}

identity path-externally-queried {
  base path-computation-method;
  description
    "Constrained-path LSP in which the path is
    obtained by querying an external source, such as a PCE server.
    In the case that an LSP is defined to be externally queried, it
    may also have associated explicit definitions (provided
    to the external source to aid computation); and the path that is
    returned by the external source is not required to provide a
    wholly resolved path back to the originating system - that is to
    say, some local computation may also be required";
}

identity path-explicitly-defined {
  base path-computation-method;
  description
    "constrained-path LSP in which the path is
    explicitly specified as a collection of strict or/and loose
    hops";
}

/**
 * Typedefs
 */

typedef te-bandwidth {
  type string {
    pattern
      '0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
      + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?))?[pP](\+)?(12[0-7]|'
      + '1[01]\d|0?\d?\d)?|0[xX][\da-fA-F]{1,8}|\d+'
      + '(\,0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
      + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?))?[pP](\+)?(12[0-7]|'
      + '1[01]\d|0?\d?\d)?|0[xX][\da-fA-F]{1,8}|\d+))*';
  }
}

```

```
description
  "This is the generic bandwidth type that is a string containing
  a list of numbers separated by commas, with each of these
  number can be non-negative decimal, hex integer, or hex float:
  (dec | hex | float)[*(','(dec | hex | float))]"
  For packet switching type, a float number is used, such as
  0xlp10.
  For OTN switching type, a list of integers can be used, such
  as '0,2,3,1', indicating 2 odu0's and 1 odu3.
  For DWDM, a list of pairs of slot number and width can be
  used, such as '0, 2, 3, 3', indicating a frequency slot 0 with
  slot width 2 and a frequency slot 3 with slot width 3.";
} // te-bandwidth

typedef te-ds-class {
  type uint8 {
    range "0..7";
  }
  description
    "The Differentiated Class-Type of traffic.";
  reference "RFC4124: section-4.3.1";
}

typedef te-link-direction {
  type enumeration {
    enum INCOMING {
      description
        "explicit route represents an incoming link on a node";
    }
    enum OUTGOING {
      description
        "explicit route represents an outgoing link on a node";
    }
  }
  description
    "enumerated type for specifying direction of link on a node";
}

typedef te-label-direction {
  type enumeration {
    enum FORWARD {
      description
        "Label allocated for the forward LSP direction";
    }
    enum REVERSE {
      description
        "Label allocated for the reverse LSP direction";
    }
  }
}
```

```
    }
    description
      "enumerated type for specifying the forward or reverse
      label";
  }

  typedef te-hop-type {
    type enumeration {
      enum LOOSE {
        description
          "loose hop in an explicit path";
      }
      enum STRICT {
        description
          "strict hop in an explicit path";
      }
    }
    description
      "enumerated type for specifying loose or strict
      paths";
    reference "RFC3209: section-4.3.2";
  }

  identity LSP_METRIC_TYPE {
    description
      "Base identity for types of LSP metric specification";
  }

  identity LSP_METRIC_RELATIVE {
    base LSP_METRIC_TYPE;
    description
      "The metric specified for the LSPs to which this identity refers
      is specified as a relative value to the IGP metric cost to the
      LSP's tail-end.";
  }

  identity LSP_METRIC_ABSOLUTE {
    base LSP_METRIC_TYPE;
    description
      "The metric specified for the LSPs to which this identity refers
      is specified as an absolute value";
  }

  identity LSP_METRIC_INHERITED {
    base LSP_METRIC_TYPE;
    description
      "The metric for for the LSPs to which this identity refers is
      not specified explicitly - but rather inherited from the IGP
```



```
        cost directly";
    }

    identity tunnel-type {
        description
            "Base identity from which specific tunnel types are
            derived.";
    }

    identity tunnel-p2p {
        base tunnel-type;
        description
            "TE point-to-point tunnel type.";
    }

    identity tunnel-p2mp {
        base tunnel-type;
        description
            "TE point-to-multipoint tunnel type.";
        reference "RFC4875";
    }

    identity tunnel-action-type {
        description
            "Base identity from which specific tunnel action types
            are derived.";
    }

    identity tunnel-action-resetup {
        base tunnel-action-type;
        description
            "TE tunnel action resetup. Tears the
            tunnel's current LSP (if any) and
            attempts to re-establish a new LSP";
    }

    identity tunnel-action-reoptimize {
        base tunnel-action-type;
        description
            "TE tunnel action reoptimize.
            Reoptimizes placement of the tunnel LSP(s)";
    }

    identity tunnel-action-switchpath {
        base tunnel-action-type;
        description
            "TE tunnel action reoptimize
            Switches the tunnel's LSP to use the specified path";
    }
```

```
}

identity te-action-result {
  description
    "Base identity from which specific TE action results
    are derived.";
}

identity te-action-success {
  base te-action-result;
  description "TE action successful.";
}

identity te-action-fail {
  base te-action-result;
  description "TE action failed.";
}

identity tunnel-action-inprogress {
  base te-action-result;
  description "TE action inprogress.";
}

identity tunnel-admin-state-type {
  description
    "Base identity for TE tunnel admin states";
}

identity tunnel-admin-state-up {
  base tunnel-admin-state-type;
  description "Tunnel administratively state up";
}

identity tunnel-admin-state-down {
  base tunnel-admin-state-type;
  description "Tunnel administratively state down";
}

identity tunnel-state-type {
  description
    "Base identity for TE tunnel states";
}

identity tunnel-state-up {
  base tunnel-state-type;
  description "Tunnel state up";
}
```

```
identity tunnel-state-down {
  base tunnel-state-type;
  description "Tunnel state down";
}

identity lsp-state-type {
  description
    "Base identity for TE LSP states";
}

identity lsp-path-computing {
  base lsp-state-type;
  description
    "State path compute in progress";
}

identity lsp-path-computation-ok {
  base lsp-state-type;
  description
    "State path compute successful";
}

identity lsp-path-computatione-failed {
  base lsp-state-type;
  description
    "State path compute failed";
}

identity lsp-state-setting-up {
  base lsp-state-type;
  description
    "State setting up";
}

identity lsp-state-setup-ok {
  base lsp-state-type;
  description
    "State setup successful";
}

identity lsp-state-setup-failed {
  base lsp-state-type;
  description
    "State setup failed";
}

identity lsp-state-up {
  base lsp-state-type;
```

```
    description "State up";
  }

  identity lsp-state-tearing-down {
    base lsp-state-type;
    description
      "State tearing down";
  }

  identity lsp-state-down {
    base lsp-state-type;
    description "State down";
  }

  identity path-invalidation-action-type {
    description
      "Base identity for TE path invalidation action types";
  }

  identity path-invalidation-action-drop-type {
    base path-invalidation-action-type;
    description
      "TE path invalidation action drop";
  }

  identity path-invalidation-action-drop-tear {
    base path-invalidation-action-type;
    description
      "TE path invalidation action tear";
  }

  identity lsp-restoration-type {
    description
      "Base identity from which LSP restoration types are
      derived.";
  }

  identity lsp-restoration-restore-any {
    base lsp-restoration-type;
    description
      "Restores when any of the LSPs is affected by a failure";
  }

  identity lsp-restoration-restore-all {
    base lsp-restoration-type;
    description
      "Restores when all the tunnel LSPs are affected by failure";
  }
```

```
identity restoration-scheme-type {
  description
    "Base identity for LSP restoration schemes";
  reference "RFC4872";
}

identity restoration-scheme-preconfigured {
  base restoration-scheme-type;
  description
    "Restoration LSP is preconfigured prior to the failure";
}

identity restoration-scheme-precomputed {
  base restoration-scheme-type;
  description
    "Restoration LSP is precomputed prior to the failure";
}

identity restoration-scheme-presignaled {
  base restoration-scheme-type;
  description
    "Restoration LSP is presignaled prior to the failure";
}

identity lsp-protection-type {
  description
    "Base identity from which LSP protection types are
    derived.";
}

identity lsp-protection-unprotected {
  base lsp-protection-type;
  description
    "LSP protection 'Unprotected'";
  reference "RFC4872";
}

identity lsp-protection-reroute-extra {
  base lsp-protection-type;
  description
    "LSP protection '(Full) Rerouting'";
  reference "RFC4872";
}

identity lsp-protection-reroute {
  base lsp-protection-type;
  description
    "LSP protection 'Rerouting without Extra-Traffic'";
```

```
    reference "RFC4872";
  }

  identity lsp-protection-1-for-n {
    base lsp-protection-type;
    description
      "LSP protection '1:N Protection with Extra-Traffic'";
    reference "RFC4872";
  }

  identity lsp-protection-unidir-1-to-1 {
    base lsp-protection-type;
    description
      "LSP protection '1+1 Unidirectional Protection'";
    reference "RFC4872";
  }

  identity lsp-protection-bidir-1-to-1 {
    base lsp-protection-type;
    description
      "LSP protection '1+1 Bidirectional Protection'";
    reference "RFC4872";
  }

  identity lsp-protection-extra-traffic {
    base lsp-protection-type;
    description
      "LSP protection 'Extra-Traffic'";
    reference
      "ITU-T G.808, RFC 4427.";
  }

  identity lsp-protection-state {
    description
      "Base identity of protection states for reporting
      purposes.";
  }

  identity normal {
    base lsp-protection-state;
    description "Normal state.";
  }

  identity signal-fail-of-protection {
    base lsp-protection-state;
    description
      "There is a SF condition on the protection transport
      entity which has higher priority than the FS command.";
```

```
        reference
            "ITU-T G.873.1, G.8031, G.8131";
    }

    identity lockout-of-protection {
        base lsp-protection-state;
        description
            "A Loss of Protection (LoP) command is active.";
        reference
            "ITU-T G.808, RFC 4427";
    }

    identity forced-switch {
        base lsp-protection-state;
        description
            "A forced switch (FS) command is active.";
        reference
            "ITU-T G.808, RFC 4427";
    }

    identity signal-fail {
        base lsp-protection-state;
        description
            "There is a SF condition on either the working
            or the protection path.";
        reference
            "ITU-T G.808, RFC 4427";
    }

    identity signal-degrade {
        base lsp-protection-state;
        description
            "There is an SD condition on either the working or the
            protection path.";
        reference
            "ITU-T G.808, RFC 4427";
    }

    identity manual-switch {
        base lsp-protection-state;
        description
            "A manual switch (MS) command is active.";
        reference
            "ITU-T G.808, RFC 4427";
    }

    identity wait-to-restore {
        base lsp-protection-state;
```

```
    description
      "A wait time to restore (WTR) is running.";
    reference
      "ITU-T G.808, RFC 4427";
  }

  identity do-not-revert {
    base lsp-protection-state;
    description
      "A DNR condition is active because of a non-revertive
      behavior.";
    reference
      "ITU-T G.808, RFC 4427";
  }

  identity failure-of-protocol {
    base lsp-protection-state;
    description
      "The protection is not working because of a failure of
      protocol condition.";
    reference
      "ITU-T G.873.1, G.8031, G.8131";
  }

  identity protection-external-commands {
    description
      "Protection external commands for trouble shooting
      purposes.";
  }

  identity action-freeze {
    base protection-external-commands;
    description
      "A temporary configuration action initiated by an operator
      command to prevent any switch action to be taken and as such
      freezes the current state.";
    reference
      "ITU-T G.808, RFC 4427";
  }

  identity clear-freeze {
    base protection-external-commands;
    description
      "An action that clears the active freeze state.";
    reference
      "ITU-T G.808, RFC 4427";
  }
```



```
identity action-lockout-of-normal {
  base protection-external-commands;
  description
    "A temporary configuration action initiated by an operator
    command to ensure that the normal traffic is not allowed
    to use the protection transport entity.";
  reference
    "ITU-T G.808, RFC 4427";
}

identity clear-lockout-of-normal {
  base protection-external-commands;
  description
    "An action that clears the active lockout of normal state.";
  reference
    "ITU-T G.808, RFC 4427";
}

identity action-lockout-of-protection {
  base protection-external-commands;
  description
    "A temporary configuration action initiated by an operator
    command to ensure that the protection transport entity is
    temporarily not available to transport a traffic signal
    (either normal or extra traffic).";
  reference
    "ITU-T G.808, RFC 4427";
}

identity action-forced-switch {
  base protection-external-commands;
  description
    "A switch action initiated by an operator command to switch
    the extra traffic signal, the normal traffic signal, or the
    null signal to the protection transport entity, unless an
    equal or higher priority switch command is in effect.";
  reference
    "ITU-T G.808, RFC 4427";
}

identity action-manual-switch {
  base protection-external-commands;
  description
    "A switch action initiated by an operator command to switch
    the extra traffic signal, the normal traffic signal #i, or
    the null signal to the protection transport entity, unless
    a fault condition exists on other transport entities or an
    equal or higher priority switch command is in effect.";
```

```
    reference
      "ITU-T G.808, RFC 4427";
  }

  identity action-exercise {
    base protection-external-commands;
    description
      "An action to start testing if the APS communication is
       operating correctly. It is lower priority than any other
       state or command.";
    reference
      "ITU-T G.808, RFC 4427";
  }

  identity clear {
    base protection-external-commands;
    description
      "An action that clears the active near-end lockout of
       protection, forced switch, manual switch, WTR state,
       or exercise command.";
    reference
      "ITU-T G.808, RFC 4427";
  }

  identity switching-capabilities {
    description
      "Base identity for interface switching capabilities";
    reference "RFC3471";
  }

  identity switching-pscl {
    base switching-capabilities;
    description
      "Packet-Switch Capable-1 (PSC-1)";
    reference "RFC3471";
  }

  identity switching-evpl {
    base switching-capabilities;
    description
      "Ethernet Virtual Private Line (EVPL)";
  }

  identity switching-l2sc {
    base switching-capabilities;
    description
```

```
    "Layer-2 Switch Capable (L2SC)";
    reference "RFC3471";
}

identity switching-tdm {
    base switching-capabilities;
    description
        "Time-Division-Multiplex Capable (TDM)";
    reference "RFC3471";
}

identity switching-otn {
    base switching-capabilities;
    description
        "OTN-TDM capable";
}

identity switching-dcsc {
    base switching-capabilities;
    description
        "Data Channel Switching Capable (DCSC)";
}

identity switching-lsc {
    base switching-capabilities;
    description
        "Lambda-Switch Capable (LSC)";
    reference "RFC3471";
}

identity switching-fsc {
    base switching-capabilities;
    description
        "Fiber-Switch Capable (FSC)";
    reference "RFC3471";
}

identity lsp-encoding-types {
    description
        "Base identity for encoding types";
    reference "RFC3471";
}

identity lsp-encoding-packet {
    base lsp-encoding-types;
    description
        "Packet LSP encoding";
    reference "RFC3471";
}
```

```
}

identity lsp-encoding-ethernet {
  base lsp-encoding-types;
  description
    "Ethernet LSP encoding";
  reference "RFC3471";
}

identity lsp-encoding-pdh {
  base lsp-encoding-types;
  description
    "ANSI/ETSI LSP encoding";
  reference "RFC3471";
}

identity lsp-encoding-sdh {
  base lsp-encoding-types;
  description
    "SDH ITU-T G.707 / SONET ANSI T1.105 LSP encoding";
  reference "RFC3471";
}

identity lsp-encoding-digital-wrapper {
  base lsp-encoding-types;
  description
    "Digital Wrapper LSP encoding";
  reference "RFC3471";
}

identity lsp-encoding-lambda {
  base lsp-encoding-types;
  description
    "Lambda (photonic) LSP encoding";
  reference "RFC3471";
}

identity lsp-encoding-fiber {
  base lsp-encoding-types;
  description
    "Fiber LSP encoding";
  reference "RFC3471";
}

identity lsp-encoding-fiber-channel {
  base lsp-encoding-types;
  description
    "FiberChannel LSP encoding";
```

```
    reference "RFC3471";
  }

  identity lsp-encoding-oduk {
    base lsp-encoding-types;
    description
      "G.709 ODUk (Digital Path)LSP encoding";
  }

  identity lsp-encoding-optical-channel {
    base lsp-encoding-types;
    description
      "Line (e.g., 8B/10B) LSP encoding";
  }

  identity lsp-encoding-line {
    base lsp-encoding-types;
    description
      "Line (e.g., 8B/10B) LSP encoding";
  }

  identity path-signaling-type {
    description
      "base identity from which specific LSPs path
       setup types are derived";
  }

  identity path-setup-static {
    base path-signaling-type;
    description
      "Static LSP provisioning path setup";
  }

  identity path-setup-rsvp {
    base path-signaling-type;
    description
      "RSVP-TE signaling path setup";
    reference "RFC3209";
  }

  identity path-setup-sr {
    base path-signaling-type;
    description
      "Segment-routing path setup";
  }

  identity path-scope-type {
    description
```

```
        "base identity from which specific path
        scope types are derived";
    }

    identity path-scope-segment {
        base path-scope-type;
        description
            "Path scope segment";
    }

    identity path-scope-end-to-end {
        base path-scope-type;
        description
            "Path scope end to end";
    }

    /* TE basic features */
    feature p2mp-te {
        description
            "Indicates support for P2MP-TE";
        reference "RFC4875";
    }

    feature frr-te {
        description
            "Indicates support for TE FastReroute (FRR)";
        reference "RFC4090";
    }

    feature extended-admin-groups {
        description
            "Indicates support for TE link extended admin
            groups.";
        reference "RFC7308";
    }

    feature named-path-affinities {
        description
            "Indicates support for named path affinities";
    }

    feature named-extended-admin-groups {
        description
            "Indicates support for named extended admin groups";
    }

    feature named-srlg-groups {
        description
```

```
    "Indicates support for named SRLG groups";
  }

  feature named-path-constraints {
    description
      "Indicates support for named path constraints";
  }

  feature path-optimization-metric {
    description
      "Indicates support for path optimization metric";
  }

  feature path-optimization-objective-function {
    description
      "Indicates support for path optimization objective function";
  }

  identity route-usage-type {
    description
      "Base identity for route usage";
  }

  identity route-include-ero {
    base route-usage-type;
    description
      "Include ERO from route";
  }

  identity route-exclude-ero {
    base route-usage-type;
    description
      "Exclude ERO from route";
  }

  identity route-exclude-srlg {
    base route-usage-type;
    description
      "Exclude SRLG from route";
  }

  identity path-metric-type {
    description
      "Base identity for path metric type";
  }

  identity path-metric-te {
    base path-metric-type;
```

```
    description
      "TE path metric";
    reference "RFC3785";
  }

  identity path-metric-igp {
    base path-metric-type;
    description
      "IGP path metric";
    reference "RFC3785";
  }

  identity path-metric-hop {
    base path-metric-type;
    description
      "Hop path metric";
  }

  identity path-metric-delay-average {
    base path-metric-type;
    description
      "Unidirectional average link delay";
    reference "RFC7471";
  }

  identity path-metric-residual-bandwidth {
    base path-metric-type;
    description
      "Unidirectional Residual Bandwidth, which is defined to be
       Maximum Bandwidth [RFC3630] minus the bandwidth currently
       allocated to LSPs.";
    reference "RFC7471";
  }

  identity path-metric-optimize-includes {
    base path-metric-type;
    description
      "A metric that optimizes the number of included resources
       specified in a set";
  }

  identity path-metric-optimize-excludes {
    base path-metric-type;
    description
      "A metric that optimizes the number of excluded resources
       specified in a set";
  }
```



```
identity path-tiebreaker-type {
  description
    "Base identity for path tie-breaker type";
}

identity path-tiebreaker-minfill {
  base path-tiebreaker-type;
  description
    "Min-Fill LSP path placement";
}

identity path-tiebreaker-maxfill {
  base path-tiebreaker-type;
  description
    "Max-Fill LSP path placement";
}

identity path-tiebreaker-randoom {
  base path-tiebreaker-type;
  description
    "Random LSP path placement";
}

identity bidir-provisioning-mode {
  description
    "Base identity for bidirectional provisioning
    mode.";
  reference "RFC7551";
}

identity bidir-provisioning-single-sided {
  base bidir-provisioning-mode;
  description
    "Single-sided bidirectional provioning mode";
  reference "RFC7551";
}

identity bidir-provisioning-double-sided {
  base bidir-provisioning-mode;
  description
    "Double-sided bidirectional provioning mode";
  reference "RFC7551";
}

identity bidir-association-type {
  description
    "Base identity for bidirectional association type";
  reference "RFC7551";
}
```

```
}

identity bidir-assoc-corouted {
  base bidir-association-type;
  description
    "Co-routed bidirectional association type";
  reference "RFC7551";
}

identity bidir-assoc-non-corouted {
  base bidir-association-type;
  description
    "Non co-routed bidirectional association type";
  reference "RFC7551";
}

identity resource-affinities-type {
  description
    "Base identity for resource affinities";
  reference "RFC2702";
}

identity resource-aff-include-all {
  base resource-affinities-type;
  description
    "The set of attribute filters associated with a
    tunnel all of which must be present for a link
    to be acceptable";
  reference "RFC2702 and RFC3209";
}

identity resource-aff-include-any {
  base resource-affinities-type;
  description
    "The set of attribute filters associated with a
    tunnel any of which must be present for a link
    to be acceptable";
  reference "RFC2702 and RFC3209";
}

identity resource-aff-exclude-any {
  base resource-affinities-type;
  description
    "The set of attribute filters associated with a
    tunnel any of which renders a link unacceptable";
  reference "RFC2702 and RFC3209";
}
```

```
typedef optimization-goal {
  type enumeration {
    enum minimize {
      description "Pick lowest path metric goal";
    }
    enum maximize {
      description "Pick highest path metric goal";
    }
    enum randomize {
      description
        "Pick a path at random from list of
         equally favorable ones";
    }
  }
  description "TE optimization goal";
}

identity te-optimization-criterion {
  description
    "Base identity for TE optimization criterion.";
  reference
    "RFC3272: Overview and Principles of Internet Traffic
     Engineering.";
}

identity not-optimized {
  base te-optimization-criterion;
  description "Optimization is not applied.";
}

identity cost {
  base te-optimization-criterion;
  description "Optimized on cost.";
}

identity delay {
  base te-optimization-criterion;
  description "Optimized on delay.";
}

/*
 * Typedefs
 */

typedef percentage {
  type uint8 {
    range "0..100";
  }
}
```

```
    description
      "Integer indicating a percentage value";
  }

typedef performance-metric-normality {
  type enumeration {
    enum "unknown" {
      value 0;
      description
        "Unknown.";
    }
    enum "normal" {
      value 1;
      description
        "Normal.";
    }
    enum "abnormal" {
      value 2;
      description
        "Abnormal. The anomalous bit is set.";
    }
  }
  description
    "Indicates whether a performance metric is normal, abnormal, or
    unknown.";
  reference
    "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
    RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
    RFC7823: Performance-Based Path Selection for Explicitly
    Routed Label Switched Paths (LSPs) Using TE Metric
    Extensions";
}

typedef te-admin-status {
  type enumeration {
    enum up {
      description
        "Enabled.";
    }
    enum down {
      description
        "Disabled.";
    }
    enum testing {
      description
        "In some test mode.";
    }
    enum preparing-maintenance {
```

```
        description
            "Resource is disabled in the control plane to prepare for
            graceful shutdown for maintenance purposes.";
        reference
            "RFC5817: Graceful Shutdown in MPLS and Generalized MPLS
            Traffic Engineering Networks";
    }
    enum maintenance {
        description
            "Resource is disabled in the data plane for maintenance
            purposes.";
    }
}
description
    "Defines a type representing the administrative status of
    a TE resource.";
}

typedef te-global-id {
    type uint32;
    description
        "An identifier to uniquely identify an operator, which can be
        either a provider or a client.
        The definition of this type is taken from RFC6370 and RFC5003.
        This attribute type is used solely to provide a globally
        unique context for TE topologies.";
}

typedef te-link-access-type {
    type enumeration {
        enum point-to-point {
            description
                "The link is point-to-point.";
        }
        enum multi-access {
            description
                "The link is multi-access, including broadcast and NBMA.";
        }
    }
}
description
    "Defines a type representing the access type of a TE link.";
reference
    "RFC3630: Traffic Engineering (TE) Extensions to OSPF
    Version 2.";
}

typedef te-node-id {
    type yang:dotted-quad;
```

```
    description
      "An identifier for a node in a topology.
      The identifier is represented as 32-bit unsigned integer in
      the dotted-quad notation.
      This attribute is mapped to Router ID in
      RFC3630, RFC5329, RFC5305, and RFC6119.";
  }

typedef te-oper-status {
  type enumeration {
    enum up {
      description
        "Operational up.";
    }
    enum down {
      description
        "Operational down.";
    }
    enum testing {
      description
        "In some test mode.";
    }
    enum unknown {
      description
        "Status cannot be determined for some reason.";
    }
    enum preparing-maintenance {
      description
        "Resource is disabled in the control plane to prepare for
        graceful shutdown for maintenance purposes.";
      reference
        "RFC5817: Graceful Shutdown in MPLS and Generalized MPLS
        Traffic Engineering Networks";
    }
    enum maintenance {
      description
        "Resource is disabled in the data plane for maintenance
        purposes.";
    }
  }
  description
    "Defines a type representing the operational status of
    a TE resource.";
}

typedef te-path-disjointness {
  type bits {
    bit node {
```

```
        position 0;
        description "Node disjoint.";
    }
    bit link {
        position 1;
        description "Link disjoint.";
    }
    bit srlg {
        position 2;
        description "SRLG (Shared Risk Link Group) disjoint.";
    }
}
description
    "Type of the resource disjointness for a TE tunnel path.";
reference
    "RFC4872: RSVP-TE Extensions in Support of End-to-End
    Generalized Multi-Protocol Label Switching (GMPLS)
    Recovery";
} // te-path-disjointness

typedef te-recovery-status {
    type enumeration {
        enum normal {
            description
                "Both the recovery and working spans are fully
                allocated and active, data traffic is being
                transported over (or selected from) the working
                span, and no trigger events are reported.";
        }
        enum recovery-started {
            description
                "The recovery action has been started, but not completed.";
        }
        enum recovery-succeeded {
            description
                "The recovery action has succeeded. The working span has
                reported a failure/degrade condition and the user traffic
                is being transported (or selected) on the recovery span.";
        }
        enum recovery-failed {
            description
                "The recovery action has failed.";
        }
        enum reversion-started {
            description
                "The reversion has started.";
        }
        enum reversion-failed {
```

```
        description
            "The reversion has failed.";
    }
    enum recovery-unavailable {
        description
            "The recovery is unavailable -- either as a result of an
            operator Lockout command or a failure condition detected
            on the recovery span.";
    }
    enum recovery-admin {
        description
            "The operator has issued a command switching the user
            traffic to the recovery span.";
    }
    enum wait-to-restore {
        description
            "The recovery domain is recovering from a failuer/degrade
            condition on the working span that is being controlled by
            the Wait-to-Restore (WTR) timer.";
    }
}
description
    "Defines the status of a recovery action.";
reference
    "RFC4427: Recovery (Protection and Restoration) Terminology
    for Generalized Multi-Protocol Label Switching (GMPLS).
    RFC6378: MPLS Transport Profile (MPLS-TP) Linear Protection";
}

typedef te-template-name {
    type string {
        pattern '/?([a-zA-Z0-9\-\_\.]+)(/[a-zA-Z0-9\-\_\.]+)*';
    }
    description
        "A type for the name of a TE node template or TE link
        template.";
}

typedef te-topology-event-type {
    type enumeration {
        enum "add" {
            value 0;
            description
                "A TE node or te-link has been added.";
        }
        enum "remove" {
            value 1;
            description
```



```
        "A TE node or te-link has been removed.";
    }
    enum "update" {
        value 2;
        description
            "A TE node or te-link has been updated.";
    }
}
description "TE Event type for notifications";
} // te-topology-event-type

typedef te-topology-id {
    type string {
        pattern
            '([a-zA-Z0-9\-\_\.]+:)*'
            + '/?([a-zA-Z0-9\-\_\.]+)(/[a-zA-Z0-9\-\_\.]+)*';
    }
    description
        "An identifier for a topology.
        It is optional to have one or more prefixes at the begining,
        separated by colons. The prefixes can be the network-types,
        defined in ietf-network.yang, to help user to understand the
        topology better before further inquiry.";
}

typedef te-tp-id {
    type union {
        type uint32;           // Unnumbered
        type inet:ip-address; // IPv4 or IPv6 address
    }
    description
        "An identifier for a TE link endpoint on a node.
        This attribute is mapped to local or remote link identifier in
        RFC3630 and RFC5305.";
}

typedef admin-group {
    type binary {
        length 4;
    }
    description
        "Administrative group/Resource class/Color.";
    reference "RFC3630 and RFC5305";
}

typedef extended-admin-group {
    type binary;
    description
```

```
    "Extended administrative group/Resource class/Color.";
    reference "RFC7308";
}

typedef admin-groups {
    type union {
        type admin-group;
        type extended-admin-group;
    }
    description "TE administrative group derived type";
}

typedef srlg {
    type uint32;
    description "SRLG type";
    reference "RFC4203 and RFC5307";
}

identity path-computation-srlg-type {
    description
        "Base identity for SRLG path computation";
}

identity srlg-ignore {
    base path-computation-srlg-type;
    description
        "Ignores SRLGs in path computation";
}

identity srlg-strict {
    base path-computation-srlg-type;
    description
        "Include strict SRLG check in path computation";
}

identity srlg-preferred {
    base path-computation-srlg-type;
    description
        "Include preferred SRLG check in path computation";
}

identity srlg-weighted {
    base path-computation-srlg-type;
    description
        "Include weighted SRLG check in path computation";
}

typedef te-metric {
```

```
    type uint32;
    description
      "TE link metric";
    reference "RFC3785";
  }

  /**
   * TE bandwidth groupings
   **/
  identity otn-rate-type {
    description
      "Base type to identify OTN bit rates of various information
       structures.";
    reference "RFC7139";
  }
  identity odu0 {
    base otn-rate-type;
    description
      "ODU0 bit rate.";
  }
  identity odu1 {
    base otn-rate-type;
    description
      "ODU1 bit rate.";
  }
  identity odu2 {
    base otn-rate-type;
    description
      "ODU2 bit rate.";
  }
  identity odu3 {
    base otn-rate-type;
    description
      "ODU3 bit rate.";
  }
  identity odu4 {
    base otn-rate-type;
    description
      "ODU4 bit rate.";
  }
  identity odu2e {
    base otn-rate-type;
    description
      "ODU2e bit rate.";
  }
  identity oduc {
    base otn-rate-type;
    description
```

```
        "ODUCn bit rate.";
    }
    identity oduflex {
        base otn-rate-type;
        description
            "ODUflex bit rate.";
    }

    identity wdm-spectrum-type {
        description
            "Base type to identify WDM spectrum type.";
    }
    identity cwdm {
        base wdm-spectrum-type;
        description "CWDM.";
        reference "RFC6205";
    }
    identity dwdm {
        base wdm-spectrum-type;
        description "DWDM.";
        reference "RFC6205";
    }
    identity flexible-grid {
        base wdm-spectrum-type;
        description "Flexible grid.";
        reference "RFC6205";
    }
}

grouping te-bandwidth {
    description
        "This grouping defines the generic TE bandwidth.
        For some known data plane technologies, specific modeling
        structures are specified. The string encoded te-bandwidth
        type is used for un-specified technologies.
        The modeling structure can be augmented later for other
        technologies.";
    container te-bandwidth {
        description
            "Container that specifies TE bandwidth.";
        choice technology {
            default generic;
            description
                "Data plane technology type.";
            case generic {
                leaf generic {
                    type te-bandwidth;
                    description
                        "Bandwidth specified in a generic format.";
                }
            }
        }
    }
}
```

```

    }
  }
}

/**
 * TE label groupings
 */
grouping te-label {
  description
    "This grouping defines the generic TE label.
    The modeling structure can be augmented for each technology.
    For un-specified technologies, rt-types:generalized-label
    is used.";
  container te-label {
    description
      "Container that specifies TE label.";
    choice technology {
      default generic;
      description
        "Data plane technology type.";
      case generic {
        leaf generic {
          type rt-types:generalized-label;
          description
            "TE label specified in a generic format.";
        }
      }
    }
    leaf direction {
      type te-label-direction;
      description "Label direction";
    }
  }
}

/**
 * TE performance metric groupings
 */
grouping performance-metric-container {
  description
    "A container containing performance metric attributes.";
  container performance-metric {
    description
      "Link performance information in real time.";
    reference
      "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions."
  }
}

```

```

    RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
    RFC7823: Performance-Based Path Selection for Explicitly
    Routed Label Switched Paths (LSPs) Using TE Metric
    Extensions";
  container measurement {
    description
      "Measured performance metric values. Static configuration
      and manual overrides of these measurements are also
      allowed.";
    uses performance-metric-attributes;
  }
  container normality
  {
    description
      "Performance metric normality values.";
    uses performance-metric-normality-attributes;
  }
  uses performance-metric-throttle-container;
}
} // performance-metric-container

grouping te-topology-identifier {
  description
    "Augmentation for TE topology.";
  container te-topology-identifier {
    description "TE topology identifier container";
    leaf provider-id {
      type te-types:te-global-id;
      description
        "An identifier to uniquely identify a provider.";
    }
    leaf client-id {
      type te-types:te-global-id;
      description
        "An identifier to uniquely identify a client.";
    }
    leaf topology-id {
      type te-types:te-topology-id;
      description
        "It is presumed that a datastore will contain many
        topologies. To distinguish between topologies it is
        vital to have UNIQUE topology identifiers.";
    }
  }
}
}
```

```
grouping performance-metric-attributes {
  description
    "Link performance information in real time.";
  reference
    "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
     RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
     RFC7823: Performance-Based Path Selection for Explicitly
     Routed Label Switched Paths (LSPs) Using TE Metric
     Extensions";
  leaf unidirectional-delay {
    type uint32 {
      range 0..16777215;
    }
    description "Delay or latency in micro seconds.";
  }
  leaf unidirectional-min-delay {
    type uint32 {
      range 0..16777215;
    }
    description "Minimum delay or latency in micro seconds.";
  }
  leaf unidirectional-max-delay {
    type uint32 {
      range 0..16777215;
    }
    description "Maximum delay or latency in micro seconds.";
  }
  leaf unidirectional-delay-variation {
    type uint32 {
      range 0..16777215;
    }
    description "Delay variation in micro seconds.";
  }
  leaf unidirectional-packet-loss {
    type decimal64 {
      fraction-digits 6;
      range "0 .. 50.331642";
    }
    description
      "Packet loss as a percentage of the total traffic sent
       over a configurable interval. The finest precision is
       0.000003%.";
  }
  leaf unidirectional-residual-bandwidth {
    type rt-types:bandwidth-ieee-float32;
    description
      "Residual bandwidth that subtracts tunnel
       reservations from Maximum Bandwidth (or link capacity)
```

```
        [RFC3630] and provides an aggregated remainder across QoS
        classes.";
    }
    leaf unidirectional-available-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        description
            "Available bandwidth that is defined to be residual
            bandwidth minus the measured bandwidth used for the
            actual forwarding of non-RSVP-TE LSP packets. For a
            bundled link, available bandwidth is defined to be the
            sum of the component link available bandwidths.";
    }
    leaf unidirectional-utilized-bandwidth {
        type rt-types:bandwidth-ieee-float32;
        description
            "Bandwidth utilization that represents the actual
            utilization of the link (i.e. as measured in the router).
            For a bundled link, bandwidth utilization is defined to
            be the sum of the component link bandwidth
            utilizations.";
    }
} // performance-metric-attributes

grouping performance-metric-normality-attributes {
    description
        "Link performance metric normality attributes.";
    reference
        "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
        RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
        RFC7823: Performance-Based Path Selection for Explicitly
        Routed Label Switched Paths (LSPs) Using TE Metric
        Extensions";
    leaf unidirectional-delay {
        type te-types:performance-metric-normality;
        description "Delay normality.";
    }
    leaf unidirectional-min-delay {
        type te-types:performance-metric-normality;
        description "Minimum delay or latency normality.";
    }
    leaf unidirectional-max-delay {
        type te-types:performance-metric-normality;
        description "Maximum delay or latency normality.";
    }
    leaf unidirectional-delay-variation {
        type te-types:performance-metric-normality;
        description "Delay variation normality.";
    }
}
```



```
leaf unidirectional-packet-loss {
  type te-types:performance-metric-normality;
  description "Packet loss normality.";
}
leaf unidirectional-residual-bandwidth {
  type te-types:performance-metric-normality;
  description "Residual bandwidth normality.";
}
leaf unidirectional-available-bandwidth {
  type te-types:performance-metric-normality;
  description "Available bandwidth normality.";
}
leaf unidirectional-utilized-bandwidth {
  type te-types:performance-metric-normality;
  description "Bandwidth utilization normality.";
}
} // performance-metric-normality-attributes

grouping performance-metric-throttle-container {
  description
    "A container controlling performance metric throttle.";
  container throttle {
    must "suppression-interval >= measure-interval" {
      error-message
        "suppression-interval cannot be less than
        measure-interval.";
      description
        "Constraint on suppression-interval and
        measure-interval.";
    }
  }
  description
    "Link performance information in real time.";
  reference
    "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
    RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
    RFC7823: Performance-Based Path Selection for Explicitly
    Routed Label Switched Paths (LSPs) Using TE Metric
    Extensions";
  leaf unidirectional-delay-offset {
    type uint32 {
      range 0..16777215;
    }
    description
      "Offset value to be added to the measured delay value.";
  }
  leaf measure-interval {
    type uint32;
    default 30;
  }
}
```

```
        description
            "Interval in seconds to measure the extended metric
            values.";
    }
    leaf advertisement-interval {
        type uint32;
        description
            "Interval in seconds to advertise the extended metric
            values.";
    }
    leaf suppression-interval {
        type uint32 {
            range "1 .. max";
        }
        default 120;
        description
            "Interval in seconds to suppress advertising the extended
            metric values.";
    }
    container threshold-out {
        uses performance-metric-attributes;
        description
            "If the measured parameter falls outside an upper bound
            for all but the min delay metric (or lower bound for
            min-delay metric only) and the advertised value is not
            already outside that bound, anomalous announcement will be
            triggered.";
    }
    container threshold-in {
        uses performance-metric-attributes;
        description
            "If the measured parameter falls inside an upper bound
            for all but the min delay metric (or lower bound for
            min-delay metric only) and the advertised value is not
            already inside that bound, normal (anomalous-flag cleared)
            announcement will be triggered.";
    }
    container threshold-accelerated-advertisement {
        description
            "When the difference between the last advertised value and
            current measured value exceed this threshold, anomalous
            announcement will be triggered.";
        uses performance-metric-attributes;
    }
}
} // performance-metric-throttle-container

/**
```

```
* TE tunnel generic groupings
**/

/* Tunnel path selection parameters */
grouping explicit-route-hop {
  description
    "The explicit route subobject grouping";
  leaf index {
    type uint32;
    description "ERO subobject index";
  }
  choice type {
    description
      "The explicit route subobject type";
    case numbered {
      description
        "Numbered link explicit route subobject";
      container numbered-hop {
        description "Numbered link hop type";
        leaf address {
          type te-types:te-tp-id;
          description
            "Numbered link TE termination point address.";
        }
        leaf hop-type {
          type te-hop-type;
          description "strict or loose hop";
        }
        leaf direction {
          type te-link-direction;
          default INCOMING;
          description "Link ERO direction";
        }
      }
    }
  }
  case as-number {
    container as-number-hop {
      leaf as-number {
        type binary {
          length 16;
        }
        description "AS number";
      }
      leaf hop-type {
        type te-hop-type;
        description
          "strict or loose hop";
      }
    }
  }
}
```

```
        description
            "Autonomous System explicit route subobject";
    }
}
case unnumbered {
    container unnumbered-hop {
        leaf node-id {
            type te-types:te-node-id;
            description
                "The identifier of a node in the TE topology.";
        }
        leaf link-tp-id {
            type te-types:te-tp-id;
            description
                "TE link termination point identifier, used
                together with te-node-id to identify the
                link termination point";
        }
        leaf hop-type {
            type te-hop-type;
            description "strict or loose hop";
        }
        leaf direction {
            type te-link-direction;
            description "Unnumbered Link ERO direction";
        }
        description
            "Unnumbered link explicit route subobject";
        reference
            "RFC3477: Signalling Unnumbered Links in
            RSVP-TE";
    }
}
case label {
    container label-hop {
        description "Label hop type";
        uses te-label;
    }
    description
        "The Label ERO subobject";
}
}

grouping record-route-subobject_state {
    description
        "The record route subobject grouping";
    leaf index {
```

```
    type uint32;
    description "RRO subobject index";
  }
  choice type {
    description
      "The record route subobject type";
    case numbered {
      leaf address {
        type te-types:te-tp-id;
        description
          "Numbered link TE termination point address.";
      }
      leaf ip-flags {
        type binary {
          length 8;
        }
        description
          "RRO IP address sub-object flags";
        reference "RFC3209";
      }
    }
    case unnumbered {
      leaf node-id {
        type te-types:te-node-id;
        description
          "The identifier of a node in the TE topology.";
      }
      leaf link-tp-id {
        type te-types:te-tp-id;
        description
          "TE link termination point identifier, used
          together with te-node-id to identify the
          link termination point";
      }
      description
        "Unnumbered link record route subobject";
      reference
        "RFC3477: Signalling Unnumbered Links in
        RSVP-TE";
    }
    case label {
      leaf value {
        type rt-types:generalized-label;
        description "the label value";
      }
      leaf label-flags {
        type binary {
          length 8;
        }
      }
    }
  }
}
```

```

    }
    description
      "Label sub-object flags";
    reference "RFC3209";
  }
  description
    "The Label ERO subobject";
}
}
}

grouping label-set-item-info {
  description "Label set item info";
  leaf inclusive-exclusive {
    type enumeration {
      enum inclusive {
        description "The label or label range is inclusive.";
      }
      enum exclusive {
        description "The label or label range is exclusive.";
      }
    }
  }
  description
    "Whether the list item is inclusive or exclusive.";
}
leaf label-start {
  type rt-types:generalized-label;
  description
    "This is the starting lable if a lable range is specified.
    This is the lable value if a single lable is specified,
    in which case, attribute 'label-end' is not set.";
}
leaf label-end {
  type rt-types:generalized-label;
  description
    "The ending lable if a lable range is specified;
    This attribute is not set, If a single lable is
    specified.";
}
leaf range-bitmap {
  type binary;
  description
    "When there are gaps between label-start and label-end,
    this attribute is used to specified the possitions
    of the used labels.";
}
}

```

```
grouping label-set-info {
  description "Label set info grouping";
  list label-set {
    key "inclusive-exclusive label-start";
    description
      "The absence of label-set implies that all labels are
       acceptable; otherwise only restricted labels are
       available.";

    uses label-set-item-info;
  }
}

/** End of TE tunnel groupings */
grouping optimizations_config {
  description "Optimization metrics configuration grouping";
  leaf metric-type {
    type identityref {
      base te-types:path-metric-type;
    }
    description "TE path metric type";
  }
  leaf weight {
    type uint8;
    description "TE path metric normalization weight";
  }
  container explicit-route-exclude-objects {
    when "../metric-type = " +
      "'te-types:path-metric-optimize-excludes'";
    description
      "Container for the exclude route object list";
    uses path-route-exclude-objects;
  }
  container explicit-route-include-objects {
    when "../metric-type = " +
      "'te-types:path-metric-optimize-includes'";
    description
      "Container for the include route object list";
    uses path-route-include-objects;
  }
}

grouping common-constraints_config {
  description
    "Common constraints grouping that can be set on
     a constraint set or directly on the tunnel";

  uses te-types:te-bandwidth {
```

```
    description
      "A requested bandwidth to use for path computation";
  }

  leaf setup-priority {
    type uint8 {
      range "0..7";
    }
    description
      "TE LSP requested setup priority";
    reference "RFC3209";
  }
  leaf hold-priority {
    type uint8 {
      range "0..7";
    }
    description
      "TE LSP requested hold priority";
    reference "RFC3209";
  }
  leaf signaling-type {
    type identityref {
      base te-types:path-signaling-type;
    }
    description "TE tunnel path signaling type";
  }
}

grouping tunnel-constraints_config {
  description
    "Tunnel constraints grouping that can be set on
    a constraint set or directly on the tunnel";
  uses te-types:te-topology-identifier;
  uses te-types:common-constraints_config;
}

grouping path-metrics-bounds_config {
  description "TE path metric bounds grouping";
  leaf metric-type {
    type identityref {
      base te-types:path-metric-type;
    }
    description "TE path metric type";
  }
  leaf upper-bound {
    type uint64;
    description "Upper bound on end-to-end TE path metric";
  }
}
```



```
}

grouping path-objective-function_config {
  description "Optimization metrics configuration grouping";
  leaf objective-function-type {
    type identityref {
      base te-types:objective-function-type;
    }
    description
      "Objective function entry";
  }
}

/**
 * TE interface generic groupings
 **/
grouping path-route-objects {
  description
    "List of EROs to be included or excluded when performing
    the path computation.";
  container explicit-route-objects {
    description
      "Container for the exclude route object list";
    list route-object-exclude-always {
      key index;
      description
        "List of explicit route objects to always exclude
        from path computation";
      uses te-types:explicit-route-hop;
    }
    list route-object-include-exclude {
      key index;
      description
        "List of explicit route objects to include or
        exclude in path computation";
      leaf explicit-route-usage {
        type identityref {
          base te-types:route-usage-type;
        }
        description "Explicit-route usage.";
      }
      uses te-types:explicit-route-hop;
    }
  }
}

grouping path-route-include-objects {
  description
```

```
    "List of EROs to be included when performing
      the path computation.";
  list route-object-include-object {
    key index;
    description
      "List of explicit route objects to be included
        in path computation";
    uses te-types:explicit-route-hop;
  }
}

grouping path-route-exclude-objects {
  description
    "List of EROs to be included when performing
      the path computation.";
  list route-object-exclude-object {
    key index;
    description
      "List of explicit route objects to be excluded
        in path computation";
    uses te-types:explicit-route-hop;
  }
}

grouping generic-path-metric-bounds {
  description "TE path metric bounds grouping";
  container path-metric-bounds {
    description "TE path metric bounds container";
    list path-metric-bound {
      key metric-type;
      description "List of TE path metric bounds";
      uses path-metrics-bounds_config;
    }
  }
}

grouping generic-path-optimization {
  description "TE generic path optimization grouping";

  container optimizations {
    description
      "The objective function container that includes
        attributes to impose when computing a TE path";

    choice algorithm {
      description "Optimizations algorithm.";
      case metric {
        if-feature path-optimization-metric;
      }
    }
  }
}
```

```

    /* Optimize by metric */
    list optimization-metric {
        key "metric-type";
        description "TE path metric type";
        uses optimizations_config;
    }
    /* Tiebreakers */
    container tiebreakers {
        description
            "The list of tiebreaker criterion to apply
            on an equally favored set of paths to pick best";
        list tiebreaker {
            key "tiebreaker-type";
            description
                "The list of tiebreaker criterion to apply
                on an equally favored set of paths to pick best";
            leaf tiebreaker-type {
                type identityref {
                    base te-types:path-metric-type;
                }
                description "The objective function";
            }
        }
    }
}
case objective-function {
    if-feature path-optimization-objective-function;
    /* Objective functions */
    container objective-function {
        description
            "The objective function container that includes
            attributes to impose when computing a TE path";
        uses path-objective-function_config;
    }
}
}
}

grouping generic-path-affinities {
    description
        "Path affinities grouping";
    container path-affinities {
        description
            "Path affinities container";
        list constraint {
            key "usage";
            description

```

```
        "List of named affinity constraints";
    leaf usage {
        type identityref {
            base resource-affinities-type;
        }
        description "Affinities usage";
    }
    leaf value {
        type admin-groups;
        description "Affinity value";
    }
}
}

grouping generic-path-srlgs {
    description
        "Path SRLG grouping";
    container path-srlgs {
        description
            "Path SRLG properties container";
        leaf usage {
            type identityref {
                base te-types:route-exclude-srlg;
            }
            description "SRLG usage";
        }
        leaf-list values {
            type srlg;
            description "SRLG value";
        }
    }
}

grouping generic-path-disjointness {
    description "Path disjointness grouping";
    leaf disjointness {
        type te-types:te-path-disjointness;
        description
            "The type of resource disjointness.
            Under primary path, disjointness level applies to
            all secondary LSPs. Under secondary, disjointness
            level overrides the one under primary";
    }
}

grouping generic-path-constraints {
    description
```

```

    "Global named path constraints configuration
    grouping";
  container path-constraints {
    description "TE named path constraints container";
    uses common-constraints_config;
    uses generic-path-disjointness;
    uses generic-path-metric-bounds;
    uses generic-path-affinities;
    uses generic-path-srlgs;
  }
}

grouping generic-path-properties {
  description "TE generic path properties grouping";
  container path-properties {
    config false;
    description "The TE path properties";
    list path-metric {
      key metric-type;
      description "TE path metric type";
      leaf metric-type {
        type identityref {
          base te-types:path-metric-type;
        }
        description "TE path metric type";
      }
      leaf accumulative-value {
        type uint64;
        description "TE path metric accumulative value";
      }
    }
  }
  uses generic-path-affinities;
  uses generic-path-srlgs;
  container path-route-objects {
    description
      "Container for the list of route objects either returned by
      the computation engine or actually used by an LSP";
    list path-route-object {
      key index;
      description
        "List of route objects either returned by the computation
        engine or actually used by an LSP";
      uses explicit-route-hop;
    }
  }
}
}
}

```

<CODE ENDS>

Figure 7: TE basic types YANG module

```
<CODE BEGINS> file "ietf-te@2018-03-03.yang"
module ietf-te {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-te";

  /* Replace with IANA when assigned */
  prefix "te";

  /* Import TE generic types */
  import ietf-te-types {
    prefix te-types;
  }

  import ietf-inet-types {
    prefix inet;
  }

  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>

     WG Chair:  Lou Berger
                <mailto:lberger@labn.net>

     WG Chair:  Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

     Editor:    Tarek Saad
                <mailto:tsaad@cisco.com>

     Editor:    Rakesh Gandhi
                <mailto:rgandhi@cisco.com>

     Editor:    Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

     Editor:    Himanshu Shah
                <mailto:hshah@ciena.com>
```

Editor: Xufeng Liu
<mailto:Xufeng_Liu@jabil.com>

Editor: Igor Bryskin
<mailto:Igor.Bryskin@huawei.com>";

```
description
  "YANG data module for TE configuration,
  state, RPC and notifications.";

revision "2018-03-03" {
  description "Latest update to TE generic YANG module.";
  reference "TBA";
}

typedef tunnel-ref {
  type leafref {
    path "/te:te/te:tunnels/te:tunnel/te:name";
  }
  description
    "This type is used by data models that need to reference
    configured TE tunnel.";
}

typedef path-ref {
  type union {
    type leafref {
      path "/te:te/te:tunnels/te:tunnel/" +
        "te:p2p-primary-paths/te:p2p-primary-path/te:name";
    }
    type leafref {
      path "/te:te/te:tunnels/te:tunnel/" +
        "te:p2p-secondary-paths/te:p2p-secondary-path/te:name";
    }
  }
  description
    "This type is used by data models that need to reference
    configured primary or secondary path of a TE tunnel.";
}

typedef tunnel-p2mp-ref {
  type leafref {
    path "/te:te/te:tunnels/te:tunnel-p2mp/te:name";
  }
  description
    "This type is used by data models that need to reference
    configured P2MP TE tunnel.";
  reference "RFC4875";
```

```
}

/**
 * TE tunnel generic groupings
 */
grouping path-affinities-contents_config {
  description
    "Path affinities constraints grouping";
  reference "RFC3630 and RFC5305";
  leaf usage {
    type identityref {
      base te-types:resource-affinities-type;
    }
    description "Affinities usage";
  }
  choice style {
    description
      "Path affinities representation style";
    case value {
      leaf value {
        type te-types:admin-groups;
        description
          "Bitmap indicating what bits are of significance";
      }
    }
    case named {
      list affinity-names {
        key "name";
        leaf name {
          type string;
          description "Affinity name";
        }
        description "List of named affinities";
      }
    }
  }
}

grouping path-affinities {
  description "Path affinities grouping";
  container path-affinities {
    description "Path affinities container";
    list constraints {
      key "usage";
      description "List of named affinity constraints";
      uses path-affinities-contents_config;
    }
  }
}
```



```
}

grouping path-srlgs-values_config {
  description "Path SRLG values properties grouping";
  reference "RFC4203 and RFC5307";
  leaf usage {
    type identityref {
      base te-types:route-exclude-srlg;
    }
    description "SRLG usage";
  }
  leaf-list values {
    type te-types:srlg;
    description "SRLG value";
    reference "RFC4203 and RFC5307";
  }
}

grouping path-srlgs {
  description "Path SRLG properties grouping";
  container path-srlgs {
    description "Path SRLG properties container";
    choice style {
      description "Type of SRLG representation";
      case values {
        uses path-srlgs-values_config;
      }
      case named {
        container constraints {
          description "SRLG named constraints";
          list constraint {
            key "usage";
            leaf usage {
              type identityref {
                base te-types:route-exclude-srlg;
              }
              description "SRLG usage";
            }
          }
          container constraint {
            description "Container for named SRLG list";
            list srlg-names {
              key "name";
              leaf name {
                type string;
                description "The SRLG name";
              }
            }
            description "List named SRLGs";
          }
        }
      }
    }
  }
}
```

```
    }
    description "List of named SRLG constraints";
  }
}
}
}
}

grouping bidirectional-association_config {
  description
    "TE tunnel associated bidirectional leaves
    grouping";
  reference "RFC7551";
  leaf id {
    type uint16;
    description
      "The TE tunnel association identifier.";
  }
  leaf source {
    type inet:ip-address;
    description "The TE tunnel association source.";
  }
  leaf global-source {
    type inet:ip-address;
    description "The TE tunnel association global source.";
  }
  leaf type {
    type identityref {
      base te-types:bidir-association-type;
    }
    default te-types:bidir-assoc-non-corouted;
    description "The TE tunnel association type.";
  }
  leaf provisioning {
    type identityref {
      base te-types:bidir-provisioning-mode;
    }
    description
      "Describes the provisioning model of the
      associated bidirectional LSP";
    reference
      "draft-ietf-teas-mpls-tp-rsvp-te-ext-
      associated-lsp, section-3.2";
  }
}

grouping bidir-assoc-properties {
```

```
description
  "TE tunnel associated bidirectional properties
  grouping";
reference "RFC7551";
container bidirectional {
  description
    "TE tunnel associated bidirectional attributes.";
  container association {
    description
      "Tunnel bidirectional association properties";
    uses bidirectional-association_config;
  }
}

grouping p2p-reverse-primary-path-properties {
  description "tunnel path properties.";
  reference "RFC7551";
  container p2p-reverse-primary-path {
    description "Tunnel reverse primary path properties";
    uses p2p-path-reverse-properties_config;
    container state {
      config false;
      description
        "Configuration applied parameters and state";
      uses p2p-path-properties_state;
    }
    container p2p-reverse-secondary-path {
      description "Tunnel reverse secondary path properties";
      uses p2p-reverse-path-candidate-secondary-path_config;
    }
  }
}

grouping p2p-secondary-path-properties {
  description "tunnel path properties.";
  uses p2p-path-properties_config;
  uses protection-restoration-params_config;
  container state {
    config false;
    description
      "Configuration applied parameters and state";
    uses p2p-path-properties_state;
  }
}

grouping p2p-primary-path-properties {
  description
```

```
    "TE tunnel primary path properties grouping";
    uses hierarchical-link;
    uses p2p-path-properties_config;
    container state {
        config false;
        description
            "Configuration applied parameters and state";
        uses p2p-path-properties_state;
    }
}

grouping path-properties_state {
    description "Computed path properties grouping";
    leaf metric-type {
        type identityref {
            base te-types:path-metric-type;
        }
        description "TE path metric type";
    }
    leaf accumulative-value {
        type uint64;
        description "TE path metric accumulative value";
    }
}

grouping path-properties {
    description "TE computed path properties grouping";
    container path-properties {
        description "The TE path computed properties";
        list path-metric {
            key metric-type;
            description "TE path metric type";
            leaf metric-type {
                type leafref {
                    path "../state/metric-type";
                }
                description "TE path metric type";
            }
        }
        container state {
            config false;
            description
                "Configuration applied parameters and state";
            uses path-properties_state;
        }
    }
    uses path-affinities;
    uses path-srlgs;
    container path-route-objects {
```

```
    description
      "Container for the list of computed route objects
      as returned by the computation engine";
    list path-computed-route-object {
      key index;
      description
        "List of computed route objects returned by the
        computation engine";
      leaf index {
        type leafref {
          path "../state/index";
        }
        description "Index of computed route object";
      }
      container state {
        config false;
        description
          "Configuration applied parameters and state";
        uses te-types:explicit-route-hop;
      }
    }
  }
}
uses shared-resources-tunnels;
}

grouping p2p-path-properties_state {
  description "TE per path state parameters";
  uses path-properties {
    description "The TE path computed properties";
  }
  container lsps {
    description "TE LSPs container";
    list lsp {
      key
        "source destination tunnel-id lsp-id "+
        "extended-tunnel-id";
      description "List of LSPs associated with the tunnel.";
      uses lsp-properties_state;
      uses shared-resources-tunnels_state;
      uses lsp-record-route-information_state;
      uses path-properties {
        description "The TE path actual properties";
      }
    }
  }
}
}
```

```
grouping p2p-path-properties-common_config {
  description
    "TE tunnel common path properties configuration grouping";
  leaf name {
    type string;
    description "TE path name";
  }
  leaf path-setup-protocol {
    type identityref {
      base te-types:path-signaling-type;
    }
    description
      "Signaling protocol used to set up this tunnel";
  }
  leaf path-computation-method {
    type identityref {
      base te-types:path-computation-method;
    }
    default te-types:path-locally-computed;
    description
      "The method used for computing the path, either
      locally computed, queried from a server or not
      computed at all (explicitly configured).";
  }
  leaf path-computation-server {
    when "../path-computation-method = "+"
    "'te-types:path-externally-queried'" {
      description
        "The path-computation server when the path is
        externally queried";
    }
    type inet:ip-address;
    description
      "Address of the external path computation
      server";
  }
  leaf compute-only {
    type empty;
    description
      "When set, the path is computed and updated whenever
      the topology is updated. No resources are committed
      or reserved in the network.";
  }
  leaf use-path-computation {
    when "../path-computation-method =" +
    "'te-types:path-locally-computed'";
    type boolean;
    description "A CSPF dynamically computed path";
  }
}
```

```
    }
    leaf lockdown {
      type empty;
      description
        "Indicates no reoptimization to be attempted for
        this path.";
    }
    leaf path-scope {
      type identityref {
        base te-types:path-scope-type;
      }
      default te-types:path-scope-end-to-end;
      description "Path scope if segment or an end-to-end path";
    }
  }
}

grouping p2p-path-reverse-properties_config {
  description
    "TE tunnel reverse path properties configuration
    grouping";
  uses p2p-path-properties-common_config;
  uses path-constraints_config;
  uses te-types:generic-path-optimization;
  leaf named-path-constraint {
    if-feature te-types:named-path-constraints;
    type leafref {
      path "../../../../../globals/"
        + "named-path-constraints/named-path-constraint/"
        + "name";
    }
    description
      "Reference to a globally defined named path
      constraint set";
  }
}

grouping p2p-path-properties_config {
  description
    "TE tunnel path properties configuration grouping";
  uses p2p-path-properties-common_config;
  uses path-constraints_config;
  uses te-types:generic-path-optimization;
  leaf preference {
    type uint8 {
      range "1..255";
    }
    description
      "Specifies a preference for this path. The lower the
```

```
        number higher the preference";
    }
    leaf named-path-constraint {
        if-feature te-types:named-path-constraints;
        type leafref {
            path "../../../../../globals/"
            + "named-path-constraints/named-path-constraint/"
            + "name";
        }
        description
            "Reference to a globally defined named path
            constraint set";
    }
}

/* TE tunnel configuration data */
grouping tunnel-p2mp-params_config {
    description
        "Configuration parameters relating to TE tunnel";
    leaf name {
        type string;
        description "TE tunnel name.";
    }
    leaf identifier {
        type uint16;
        description
            "TE tunnel Identifier.";
    }
    leaf description {
        type string;
        description
            "Textual description for this TE tunnel";
    }
}

grouping hierarchical-link_config {
    description
        "Hierarchical link configuration grouping";
    reference "RFC4206";
    leaf local-te-node-id {
        type te-types:te-node-id;
        description
            "Local TE node identifier";
    }
    leaf local-te-link-tp-id {
        type te-types:te-tp-id;
        description
            "Local TE link termination point identifier";
    }
}
```



```
    }
    leaf remote-te-node-id {
        type te-types:te-node-id;
        description
            "Remote TE node identifier";
    }
    uses te-types:te-topology-identifier;
}

grouping hierarchical-link {
    description
        "Hierarchical link grouping";
    reference "RFC4206";
    container hierarchical-link {
        description
            "Identifies a hierarchical link (in client layer)
             that this tunnel is associated with.";
        uses hierarchical-link_config;
    }
}

grouping protection-restoration-params_state {
    description
        "Protection parameters grouping";
    leaf lockout-of-normal {
        type boolean;
        description
            "
            When set to 'True', it represents a lockout of normal
            traffic external command. When set to 'False', it
            represents a clear lockout of normal traffic external
            command. The lockout of normal traffic command applies
            to this Tunnel.
            ";
        reference
            "ITU-T G.808, RFC 4427";
    }
    leaf freeze {
        type boolean;
        description
            "
            When set to 'True', it represents a freeze external
            command. When set to 'False', it represents a clear
            freeze external command. The freeze command command
            applies to all the Tunnels which are sharing the
            protection resources with this Tunnel.
            ";
        reference
```

```
        "ITU-T G.808, RFC 4427";
    }
    leaf lsp-protection-role {
        type enumeration {
            enum working {
                description
                    "A working LSP must be a primary LSP whilst a protecting
                     LSP can be either a primary or a secondary LSP. Also,
                     known as protected LSPs when working LSPs are associated
                     with protecting LSPs.";
            }
            enum protecting {
                description
                    "A secondary LSP is an LSP that has been provisioned
                     in the control plane only; e.g. resource allocation
                     has not been committed at the data plane";
            }
        }
        description "LSP role type";
        reference "rfc4872, section 4.2.1";
    }

    leaf lsp-protection-state {
        type identityref {
            base te-types:lsp-protection-state;
        }
        description
            "The state of the APS state machine controlling which
             tunnels is using the resources of the protecting LSP.";
    }

    leaf protection-group-ingress-node-id {
        type te-types:te-node-id;
        description
            "Indicates the te-node-id of the protection group
             ingress node when the APS state represents an external
             command (LoP, SF, MS) applied to it or a WTR timer
             running on it. If the external command is not applied to
             the ingress node or the WTR timer is not running on it,
             this attribute is not specified. If value 0.0.0.0 is used
             when the te-node-id of the protection group ingress node is
             unknown (e.g., because the ingress node is outside the scope
             of control of the server)";
    }

    leaf protection-group-egress-node-id {
        type te-types:te-node-id;
        description
            "Indicates the te-node-id of the protection group egress node
             when the APS state represents an external command (LoP, SF,
```

```
    MS) applied to it or a WTR timer running on it. If the
    external command is not applied to the ingress node or
    the WTR timer is not running on it, this attribute is not
    specified. If value 0.0.0.0 is used when the te-node-id of
    the protection group ingress node is unknown (e.g., because
    the ingress node is outside the scope of control of the
    server)";
  }
}

grouping protection-restoration-params_config {
  description "Protection and restoration parameters";
  container protection {
    description "Protection parameters";
    leaf enable {
      type boolean;
      default 'false';
      description
        "A flag to specify if LSP protection is enabled";
      reference "rfc4427";
    }
    leaf protection-type {
      type identityref {
        base te-types:lsp-protection-type;
      }
      description "LSP protection type.";
    }
    leaf protection-reversion-disable {
      type boolean;
      description "Disable protection reversion to working path";
    }
    leaf hold-off-time {
      type uint32;
      units "milli-seconds";
      default 0;
      description
        "The time between the declaration of an SF or SD condition
        and the initialization of the protection switching
        algorithm.";
    }
    leaf wait-to-revert {
      type uint16;
      units seconds;
      description
        "Time to wait before attempting LSP reversion";
    }
    leaf aps-signal-id {
      type uint8 {
```

```
        range "1..255";
    }
    description
        "The APS signal number used to reference the traffic of this
        tunnel. The default value for normal traffic is 1.
        The default value for extra-traffic is 255. If not specified,
        non-default values can be assigned by the server,
        if and only if, the server controls both endpoints.";
    reference
        "ITU-T G.808.1";
}
}
container restoration {
    description "Restoration parameters";
    leaf enable {
        type boolean;
        default 'false';
        description
            "A flag to specify if LSP restoration is enabled";
        reference "rfc4427";
    }
    leaf restoration-type {
        type identityref {
            base te-types:lsp-restoration-type;
        }
        description "LSP restoration type.";
    }
    leaf restoration-scheme {
        type identityref {
            base te-types:restoration-scheme-type;
        }
        description "LSP restoration scheme.";
    }
    leaf restoration-reversion-disable {
        type boolean;
        description "Disable restoration reversion to working path";
    }
    leaf hold-off-time {
        type uint32;
        units "milli-seconds";
        description
            "The time between the declaration of an SF or SD condition
            and the initialization of the protection switching
            algorithm.";
    }
    leaf wait-to-restore {
        type uint16;
        units seconds;
    }
}
```

```
        description
          "Time to wait before attempting LSP restoration";
      }
      leaf wait-to-revert {
        type uint16;
        units seconds;
        description
          "Time to wait before attempting LSP reversion";
      }
    }
  }
}

grouping p2p-dependency-tunnels_config {
  description
    "Grouping for tunnel dependency list of tunnels";
  container dependency-tunnels {
    description "Dependency tunnels list";
    list dependency-tunnel {
      key "name";
      description "Dependency tunnel entry";
      leaf name {
        type leafref {
          path "../../../../../tunnels/tunnel/name";
          require-instance false;
        }
        description "Dependency tunnel name";
      }
      leaf encoding {
        type identityref {
          base te-types:lsp-encoding-types;
        }
        description "LSP encoding type";
        reference "RFC3945";
      }
      leaf switching-type {
        type identityref {
          base te-types:switching-capabilities;
        }
        description "LSP switching type";
        reference "RFC3945";
      }
    }
  }
}

grouping tunnel-p2p-params_config {
  description
    "Configuration parameters relating to TE tunnel";
```

```
leaf name {
  type string;
  description "TE tunnel name.";
}
leaf identifier {
  type uint16;
  description
    "TE tunnel Identifier.";
}
leaf description {
  type string;
  description
    "Textual description for this TE tunnel";
}
leaf encoding {
  type identityref {
    base te-types:lsp-encoding-types;
  }
  description "LSP encoding type";
  reference "RFC3945";
}
leaf switching-type {
  type identityref {
    base te-types:switching-capabilities;
  }
  description "LSP switching type";
  reference "RFC3945";
}
leaf provisioning-state {
  type identityref {
    base te-types:tunnel-state-type;
  }
  default te-types:tunnel-state-up;
  description "TE tunnel administrative state.";
}
leaf preference {
  type uint8 {
    range "1..255";
  }
  description
    "Specifies a preference for this tunnel.
     A lower number signifies a better preference";
}
leaf reoptimize-timer {
  type uint16;
  units seconds;
  description
    "frequency of reoptimization of
```

```
        a traffic engineered LSP";
    }
    leaf source {
        type inet:ip-address;
        description
            "TE tunnel source address.";
    }
    leaf destination {
        type inet:ip-address;
        description
            "P2P tunnel destination address";
    }
    leaf src-tp-id {
        type binary;
        description
            "TE tunnel source termination point identifier.";
    }
    leaf dst-tp-id {
        type binary;
        description
            "TE tunnel destination termination point identifier.";
    }
    uses protection-restoration-params_config;
    uses te-types:tunnel-constraints_config;
    uses p2p-dependency-tunnels_config;
}

grouping tunnel-p2p-params_state {
    description
        "State parameters relating to TE tunnel";
    leaf operational-state {
        type identityref {
            base te-types:tunnel-state-type;
        }
        default te-types:tunnel-state-up;
        description "TE tunnel administrative state.";
    }
}

grouping access-segment-info {
    description
        "info related to a segment";
    container forward {
        description
            "for the forward direction of this tunnel";
        uses te-types:label-set-info;
    }
    container reverse {
```

```
        description
          "for the reverse direction of this tunnel";
          uses te-types:label-set-info;
      }
  }

  grouping path-access-segment-info {
    description
      "If an end-to-end tunnel crosses multiple domains using
       the same technology, some additional constraints have to be
       taken in consideration in each domain";
    container path-in-segment {
      presence
        "The end-to-end tunnel starts in a previous domain;
         this tunnel is a segment in the current domain.";
      description
        "This tunnel is a segment that needs to be coordinated
         with previous segment stitched on head-end side.";
      uses access-segment-info;
    }
    container path-out-segment {
      presence
        "The end-to-end tunnel is not terminated in this domain;
         this tunnel is a segment in the current domain.";
      description
        "This tunnel is a segment that needs to be coordinated
         with previous segment stitched on head-end side.";
      uses access-segment-info;
    }
  }

  /* TE tunnel configuration/state grouping */
  grouping tunnel-p2mp-properties {
    description
      "Top level grouping for P2MP tunnel properties.";
    uses tunnel-p2mp-params_config;
    container state {
      config false;
      description
        "Configuration applied parameters and state";
      leaf operational-state {
        type identityref {
          base te-types:tunnel-state-type;
        }
        default te-types:tunnel-state-up;
        description "TE tunnel administrative state.";
      }
    }
  }
}
```



```
}

grouping p2p-path-candidate-secondary-path-config {
  description
    "Configuration parameters relating to a secondary path which
    is a candidate for a particular primary path";

  leaf secondary-path {
    type leafref {
      path "../../../../../p2p-secondary-paths/" +
        "p2p-secondary-path/name";
    }
    description
      "A reference to the secondary path that should be utilised
      when the containing primary path option is in use";
  }

  leaf path-setup-protocol {
    type identityref {
      base te-types:path-signaling-type;
    }
    description
      "Signaling protocol used to set up this tunnel";
  }
}

grouping p2p-reverse-path-candidate-secondary-path-config {
  description
    "Configuration parameters relating to a secondary path which
    is a candidate for a particular primary path";

  leaf secondary-path {
    type leafref {
      path "../../../../../p2p-secondary-paths/" +
        "p2p-secondary-path/name";
    }
    description
      "A reference to the secondary path that should be utilised
      when the containing primary path option is in use";
  }

  leaf path-setup-protocol {
    type identityref {
      base te-types:path-signaling-type;
    }
    description
      "Signaling protocol used to set up this tunnel";
  }
}
```

```
}

grouping p2p-path-candidate-secondary-path-state {
  description
    "Operational state parameters relating to a secondary path
    which is a candidate for a particular primary path";

  leaf active {
    type boolean;
    description
      "Indicates the current active path option that has
      been selected of the candidate secondary paths";
  }
}

grouping tunnel-p2p-properties {
  description
    "Top level grouping for tunnel properties.";
  uses tunnel-p2p-params_config;
  container state {
    config false;
    description
      "Configuration applied parameters and state";
    uses tunnel-p2p-params_state;
  }
  uses bidir-assoc-properties;
  container p2p-primary-paths {
    description "Set of P2P primary aths container";
    list p2p-primary-path {
      key "name";
      description
        "List of primary paths for this tunnel.";
      uses p2p-primary-path-properties;
      uses p2p-reverse-primary-path-properties;
      container candidate-p2p-secondary-paths {
        description
          "The set of candidate secondary paths which may be used
          for this primary path. When secondary paths are specified
          in the list the path of the secondary LSP in use must be
          restricted to those path options referenced. The
          priority of the secondary paths is specified within the
          list. Higher priority values are less preferred - that is
          to say that a path with priority 0 is the most preferred
          path. In the case that the list is empty, any secondary
          path option may be utilised when the current primary path
          is in use.";
        list candidate-p2p-secondary-path {
          key "secondary-path";

```

```
        description
          "List of secondary paths for this tunnel.";
        uses p2p-path-candidate-secondary-path-config;

        container state {
          config false;
          description
            "Configuration applied parameters and state";
          uses p2p-path-candidate-secondary-path-state;
        }
      }
    }
  }
}

container p2p-secondary-paths {
  description "Set of P2P secondary paths container";
  list p2p-secondary-path {
    key "name";
    description
      "List of secondary paths for this tunnel.";
    uses p2p-secondary-path-properties;
  }
}

grouping shared-resources-tunnels_state {
  description
    "The specific tunnel that is using the shared secondary path
    resources";
  leaf lsp-shared-resources-tunnel {
    type te:tunnel-ref;
    description
      "Reference to the tunnel that sharing secondary path
      resources with this tunnel";
  }
}

grouping shared-resources-tunnels {
  description
    "Set of tunnels that share secondary path resources with
    this tunnel";
  container shared-resources-tunnels {
    description
      "Set of tunnels that share secondary path resources with
      this tunnel";
    leaf-list lsp-shared-resources-tunnel {
      type te:tunnel-ref;
      description
        "Reference to the tunnel that sharing secondary path
```

```
        resources with this tunnel";
    }
}

grouping tunnel-actions {
  description "Tunnel actions";
  action tunnel-action {
    description "Tunnel action";
    input {
      leaf action-type {
        type identityref {
          base te-types:tunnel-action-type;
        }
        description "Tunnel action type";
      }
    }
    output {
      leaf action-result {
        type identityref {
          base te-types:te-action-result;
        }
        description "The result of the RPC operation";
      }
    }
  }
}

grouping tunnel-protection-actions {
  description
    "Protection external command actions";
  action protection-external-commands {
    input {
      leaf protection-external-command {
        type identityref {
          base te-types:protection-external-commands;
        }
        description
          "Protection external command";
      }
      leaf protection-group-ingress-node-id {
        type te-types:te-node-id;
        description
          "Indicates the te-node-id of the protection group
          ingress node when the external command has to be
          applied to it. If the external command is not applied
          to the ingress node, this attribute is not specified.";
      }
      leaf protection-group-egress-node-id {
```

```
    type te-types:te-node-id;
    description
      "Indicates the te-node-id of the protection group egress
      node when the external command has to be applied to it.
      If the external command is not applied to the egress node,
      This attribute is not specified.";
  }
  leaf path-ref {
    type path-ref;
    description
      "Indicates to which path the external command applies to.";
  }
  leaf traffic-type {
    type enumeration {
      enum normal-traffic {
        description
          "The manual-switch or forced-switch command applies to
          the normal traffic (this Tunnel).";
      }
      enum null-traffic {
        description
          "The manual-switch or forced-switch command applies to
          the null traffic.";
      }
      enum extra-traffic {
        description
          "The manual-switch or forced-switch command applies to
          the extra traffic (the extra-traffic Tunnel sharing
          protection bandwidth with this Tunnel).";
      }
    }
    description
      "Indicates whether the manual-switch or forced-switch
      commands applies to the normal traffic, the null traffic
      or the extra-traffic.";
    reference
      "ITU-T G.808, RFC 4427";
  }
  leaf extra-traffic-tunnel-ref {
    type te:tunnel-ref;
    description
      "In case there are multiple extra-traffic tunnels sharing
      protection bandwidth with this Tunnel (m:n protection),
      represents which extra-traffic Tunnel the manual-switch or
      forced-switch to extra-traffic command applies to.";
  }
}
```

```
}

/**** End of TE tunnel groupings ****/

/**
 * LSP related generic groupings
 */
grouping lsp-record-route-information_state {
  description "recorded route information grouping";
  container lsp-record-route-subobjects {
    description "RSVP recorded route object information";
    list record-route-subobject {
      when "../../../origin-type = 'ingress'" {
        description "Applicable on non-ingress LSPs only";
      }
      key "index";
      description "Record route sub-object list";
      uses te-types:record-route-subobject_state;
    }
  }
}

grouping lsps-state-grouping {
  description
    "LSPs state operational data grouping";
  container lsps-state {
    config false;
    description "TE LSPs state container";
    list lsp {
      key
        "source destination tunnel-id lsp-id "+
        "extended-tunnel-id";
      description "List of LSPs associated with the tunnel.";
      uses lsp-properties_state;
      uses lsp-record-route-information_state;
    }
  }
}

/**** End of TE LSP groupings ****/

/**
 * TE global generic groupings
 */

/* Global named admin-groups configuration data */
grouping named-admin-groups_config {
  description
```

```
    "Global named administrative groups configuration
    grouping";
  leaf name {
    type string;
    description
      "A string name that uniquely identifies a TE
      interface named admin-group";
  }
  leaf bit-position {
    type uint32;
    description
      "Bit position representing the administrative group";
  }
}
grouping named-admin-groups {
  description
    "Global named administrative groups configuration
    grouping";
  container named-admin-groups {
    description "TE named admin groups container";
    list named-admin-group {
      if-feature te-types:extended-admin-groups;
      if-feature te-types:named-extended-admin-groups;
      key "name";
      description
        "List of named TE admin-groups";
      uses named-admin-groups_config;
    }
  }
}

/* Global named admin-srlgs configuration data */
grouping named-srlgs_config {
  description
    "Global named SRLGs configuration grouping";
  leaf name {
    type string;
    description
      "A string name that uniquely identifies a TE
      interface named srlg";
  }
  leaf group {
    type te-types:srlg;
    description "An SRLG value";
  }
  leaf cost {
    type uint32;
    description
```

```
        "SRLG associated cost. Used during path to append
        the path cost when traversing a link with this SRLG";
    }
}

grouping named-srlgs {
    description
        "Global named SRLGs configuration grouping";
    container named-srlgs {
        description "TE named SRLGs container";
        list named-srlg {
            if-feature te-types:named-srlg-groups;
            key "name";
            description
                "A list of named SRLG groups";
            uses named-srlgs_config;
        }
    }
}

/* Global named paths constraints configuration data */
grouping path-constraints_state {
    description
        "TE path constraints state";
    leaf bandwidth-generic_state {
        type te-types:te-bandwidth;
        description
            "A technology agnostic requested bandwidth to use
            for path computation";
    }
    leaf disjointness_state {
        type te-types:te-path-disjointness;
        description
            "The type of resource disjointness.";
    }
}

grouping path-constraints_config {
    description
        "Global named path constraints configuration
        grouping";
    uses te-types:common-constraints_config;
    uses te-types:generic-path-disjointness;
    uses te-types:generic-path-metric-bounds;
    uses path-affinities;
    uses path-srlgs;
    uses te-types:path-route-objects;
    uses shared-resources-tunnels {
```



```
        description
            "Set of tunnels that are allowed to share secondary path
            resources of this tunnel";
    }
    uses path-access-segment-info {
        description
            "Tunnel constraints induced by other segments.";
    }
}

grouping path-constraints {
    description "Per path constraints";
    uses path-constraints_config;
    container state {
        config false;
        description
            "Configuration applied parameters and state";
        uses path-constraints_state;
    }
}

grouping named-path-constraints {
    description
        "Global named path constraints configuration
        grouping";
    container named-path-constraints {
        description "TE named path constraints container";
        list named-path-constraint {
            if-feature te-types:named-path-constraints;
            key "name";
            leaf name {
                type string;
                description
                    "A string name that uniquely identifies a
                    path constraint set";
            }
            uses path-constraints;
            description
                "A list of named path constraints";
        }
    }
}

/* TE globals container data */
grouping globals-grouping {
    description
        "Globals TE system-wide configuration data grouping";
    container globals {
```

```
    description
      "Globals TE system-wide configuration data container";
    uses named-admin-groups;
    uses named-srlgs;
    uses named-path-constraints;
  }
}

/* TE tunnels container data */
grouping tunnels-grouping {
  description
    "Tunnels TE configuration data grouping";
  container tunnels {
    description
      "Tunnels TE configuration data container";

    list tunnel {
      key "name";
      description "P2P TE tunnels list.";
      uses tunnel-p2p-properties;
      uses tunnel-actions;
      uses tunnel-protection-actions;
    }
    list tunnel-p2mp {
      key "name";
      unique "identifier";
      description "P2MP TE tunnels list.";
      uses tunnel-p2mp-properties;
    }
  }
}

/* TE LSPs ephemeral state container data */
grouping lsp-properties_state {
  description
    "LSPs state operational data grouping";
  leaf source {
    type inet:ip-address;
    description
      "Tunnel sender address extracted from
       SENDER_TEMPLATE object";
    reference "RFC3209";
  }
  leaf destination {
    type inet:ip-address;
    description
      "Tunnel endpoint address extracted from
       SESSION object";
  }
}
```

```
    reference "RFC3209";
  }
  leaf tunnel-id {
    type uint16;
    description
      "Tunnel identifier used in the SESSION
       that remains constant over the life
       of the tunnel.";
    reference "RFC3209";
  }
  leaf lsp-id {
    type uint16;
    description
      "Identifier used in the SENDER_TEMPLATE
       and the FILTER_SPEC that can be changed
       to allow a sender to share resources with
       itself.";
    reference "RFC3209";
  }
  leaf extended-tunnel-id {
    type inet:ip-address;
    description
      "Extended Tunnel ID of the LSP.";
    reference "RFC3209";
  }
  leaf operational-state {
    type identityref {
      base te-types:lsp-state-type;
    }
    description "LSP operational state.";
  }
  leaf path-setup-protocol {
    type identityref {
      base te-types:path-signaling-type;
    }
    description
      "Signaling protocol used to set up this tunnel";
  }
  leaf origin-type {
    type enumeration {
      enum ingress {
        description
          "Origin ingress";
      }
      enum egress {
        description
          "Origin egress";
      }
    }
  }
```

```
        enum transit {
            description
                "transit";
        }
    }
    description
        "Origin type of LSP relative to the location
        of the local switch in the path.";
}

leaf lsp-resource-status {
    type enumeration {
        enum primary {
            description
                "A primary LSP is a fully established LSP for
                which the resource allocation has been committed
                at the data plane";
        }
        enum secondary {
            description
                "A secondary LSP is an LSP that has been provisioned
                in the control plane only; e.g. resource allocation
                has not been committed at the data plane";
        }
    }
    description "LSP resource allocation type";
    reference "rfc4872, section 4.2.1";
}

uses protection-restoration-params_state;
}
/**** End of TE global groupings ****/

/**
 * TE configurations container
 */
container te {
    presence "Enable TE feature.";
    description
        "TE global container.";

    /* TE Global Configuration Data */
    uses globals-grouping;

    /* TE Tunnel Configuration Data */
    uses tunnels-grouping;

    /* TE LSPs State Data */
}
```

```
    uses lsp-s-state-grouping;
}

/* TE Global RPCs/execution Data */
rpc globals-rpc {
  description
    "Execution data for TE global.";
}

/* TE interfaces RPCs/execution Data */
rpc interfaces-rpc {
  description
    "Execution data for TE interfaces.";
}

/* TE Tunnel RPCs/execution Data */
rpc tunnels-rpc {
  description "TE tunnels RPC nodes";
  input {
    container tunnel-info {
      description "Tunnel Identification";
      choice type {
        description "Tunnel information type";
        case tunnel-p2p {
          leaf p2p-id {
            type te:tunnel-ref;
            description "P2P TE tunnel";
          }
        }
        case tunnel-p2mp {
          leaf p2mp-id {
            type te:tunnel-p2mp-ref;
            description "P2MP TE tunnel";
          }
        }
      }
    }
  }
  output {
    container result {
      description
        "The container result of the RPC operation";
      leaf result {
        type enumeration {
          enum success {
            description "Origin ingress";
          }
        }
      }
    }
  }
}
```

```

        enum in-progress {
            description "Origin egress";
        }
        enum fail {
            description "transit";
        }
    }
    description "The result of the RPC operation";
}
}
}
}

/* TE Global Notification Data */
notification globals-notif {
    description
        "Notification messages for Global TE.";
}

/* TE Tunnel Notification Data */
notification tunnels-notif {
    description
        "Notification messages for TE tunnels.";
}
}
<CODE ENDS>

```

Figure 8: TE generic YANG module

```

<CODE BEGINS> file "ietf-te-device@2018-02-15.yang"
module ietf-te-device {

    namespace "urn:ietf:params:xml:ns:yang:ietf-te-device";

    /* Replace with IANA when assigned */
    prefix "te-dev";

    /* Import TE generic types */
    import ietf-te {
        prefix te;
    }

    /* Import TE generic types */
    import ietf-te-types {
        prefix te-types;
    }

    import ietf-interfaces {

```

```
    prefix if;
  }

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  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>

    WG Chair:   Lou Berger
                <mailto:lberger@labn.net>

    WG Chair:   Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

    Editor:     Tarek Saad
                <mailto:tsaad@cisco.com>

    Editor:     Rakesh Gandhi
                <mailto:rgandhi@cisco.com>

    Editor:     Vishnu Pavan Beeram
                <mailto:vbeeram@juniper.net>

    Editor:     Himanshu Shah
                <mailto:hshah@ciena.com>

    Editor:     Xufeng Liu
                <mailto:xufeng.liu@ericsson.com>

    Editor:     Xia Chen
                <mailto:jescia.chenxia@huawei.com>

    Editor:     Raqib Jones
                <mailto:raqib@Brocade.com>

    Editor:     Bin Wen
                <mailto:Bin_Wen@cable.comcast.com>";
```

```
description
  "YANG data module for TE device configurations,
  state, RPC and notifications.";

revision "2018-02-15" {
  description "Latest update to TE device YANG module.";
  reference "TBA";
}

/**
 * TE LSP device state grouping
 */
grouping lsp-device_state {
  description "TE LSP device state grouping";
  container lsp-timers {
    when "../te:origin-type = 'ingress'" {
      description "Applicable to ingress LSPs only";
    }
    description "Ingress LSP timers";
    leaf life-time {
      type uint32;
      units seconds;
      description
        "lsp life time";
    }

    leaf time-to-install {
      type uint32;
      units seconds;
      description
        "lsp installation delay time";
    }

    leaf time-to-destroy {
      type uint32;
      units seconds;
      description
        "lsp expiration delay time";
    }
  }
}

container downstream-info {
  when "../te:origin-type != 'egress'" {
    description "Applicable to ingress LSPs only";
  }
  description
    "downstream information";
}
```



```
    leaf nhop {
      type inet:ip-address;
      description
        "downstream nexthop.";
    }

    leaf outgoing-interface {
      type if:interface-ref;
      description
        "downstream interface.";
    }

    leaf neighbor {
      type inet:ip-address;
      description
        "downstream neighbor.";
    }

    leaf label {
      type rt-types:generalized-label;
      description
        "downstream label.";
    }
  }

  container upstream-info {
    when "../te:origin-type != 'ingress'" {
      description "Applicable to non-ingress LSPs only";
    }
    description
      "upstream information";

    leaf phop {
      type inet:ip-address;
      description
        "upstream nexthop or previous-hop.";
    }

    leaf neighbor {
      type inet:ip-address;
      description
        "upstream neighbor.";
    }

    leaf label {
      type rt-types:generalized-label;
      description
        "upstream label.";
    }
  }
}
```

```
    }  
  }  
}  
  
/**  
 * Device general groupings.  
 */  
grouping tunnel-device_config {  
  description "Device TE tunnel configs";  
  leaf path-invalidation-action {  
    type identityref {  
      base te-types:path-invalidation-action-type;  
    }  
    description "Tunnel path invalidtion action";  
  }  
}  
  
grouping lsp-device-timers_config {  
  description "Device TE LSP timers configs";  
  leaf lsp-install-interval {  
    type uint32;  
    units seconds;  
    description  
      "lsp installation delay time";  
  }  
  leaf lsp-cleanup-interval {  
    type uint32;  
    units seconds;  
    description  
      "lsp cleanup delay time";  
  }  
  leaf lsp-invalidation-interval {  
    type uint32;  
    units seconds;  
    description  
      "lsp path invalidation before taking action delay time";  
  }  
}  
grouping lsp-device-timers {  
  description "TE LSP timers configuration";  
  uses lsp-device-timers_config;  
}  
  
/**  
 * TE global device generic groupings  
 */  
  
/* TE interface container data */
```

```
grouping interfaces-grouping {
  description
    "Interface TE configuration data grouping";
  container interfaces {
    description
      "Configuration data model for TE interfaces.";
    uses te-all-attributes;
    list interface {
      key "interface";
      description "TE interfaces.";
      leaf interface {
        type if:interface-ref;
        description
          "TE interface name.";
      }
      /* TE interface parameters */
      uses te-attributes;
    }
  }
}

/**
 * TE interface device generic groupings
 */
grouping te-admin-groups_config {
  description
    "TE interface affinities grouping";
  choice admin-group-type {
    description
      "TE interface administrative groups
      representation type";
    case value-admin-groups {
      choice value-admin-group-type {
        description "choice of admin-groups";
        case admin-groups {
          description
            "Administrative group/Resource
            class/Color.";
          leaf admin-group {
            type te-types:admin-group;
            description
              "TE interface administrative group";
          }
        }
      }
    case extended-admin-groups {
      if-feature te-types:extended-admin-groups;
      description
        "Extended administrative group/Resource
```

```

        class/Color.";
    leaf extended-admin-group {
        type te-types:extended-admin-group;
        description
            "TE interface extended administrativei
            group";
    }
}
}
}
}
case named-admin-groups {
    list named-admin-groups {
        if-feature te-types:extended-admin-groups;
        if-feature te-types:named-extended-admin-groups;
        key named-admin-group;
        description
            "A list of named admin-group entries";
        leaf named-admin-group {
            type leafref {
                path "../../../../../te:globals/" +
                "te:named-admin-groups/te:named-admin-group/" +
                "te:name";
            }
            description "A named admin-group entry";
        }
    }
}
}
}
}

/* TE interface SRLGs */
grouping te-srlgs_config {
    description "TE interface SRLG grouping";
    choice srlg-type {
        description "Choice of SRLG configuration";
        case value-srlgs {
            list values {
                key "value";
                description "List of SRLG values that
                this link is part of.";
                leaf value {
                    type uint32 {
                        range "0..4294967295";
                    }
                    description
                        "Value of the SRLG";
                }
            }
        }
    }
}

```

```

    }
    case named-srlgs {
      list named-srlgs {
        if-feature te-types:named-srlg-groups;
        key named-srlg;
        description
          "A list of named SRLG entries";
        leaf named-srlg {
          type leafref {
            path "../../../../../te:globals/" +
              "te:named-srlgs/te:named-srlg/te:name";
          }
          description
            "A named SRLG entry";
        }
      }
    }
  }
}

grouping te-igp-flooding-bandwidth_config {
  description
    "Configurable items for igp flooding bandwidth
    threshold configuration.";
  leaf threshold-type {
    type enumeration {
      enum DELTA {
        description
          "DELTA indicates that the local
          system should flood IGP updates when a
          change in reserved bandwidth >= the specified
          delta occurs on the interface.";
      }
      enum THRESHOLD_CROSSED {
        description
          "THRESHOLD-CROSSED indicates that
          the local system should trigger an update (and
          hence flood) the reserved bandwidth when the
          reserved bandwidth changes such that it crosses,
          or becomes equal to one of the threshold values.";
      }
    }
  }
  description
    "The type of threshold that should be used to specify the
    values at which bandwidth is flooded. DELTA indicates that
    the local system should flood IGP updates when a change in
    reserved bandwidth >= the specified delta occurs on the
    interface. Where THRESHOLD_CROSSED is specified, the local

```

```
        system should trigger an update (and hence flood) the
        reserved bandwidth when the reserved bandwidth changes such
        that it crosses, or becomes equal to one of the threshold
        values";
    }

    leaf delta-percentage {
        when "../threshold-type = 'DELTA'" {
            description
                "The percentage delta can only be specified when the
                threshold type is specified to be a percentage delta of
                the reserved bandwidth";
        }
        type te-types:percentage;
        description
            "The percentage of the maximum-reservable-bandwidth
            considered as the delta that results in an IGP update
            being flooded";
    }
    leaf threshold-specification {
        when "../threshold-type = 'THRESHOLD_CROSSED'" {
            description
                "The selection of whether mirrored or separate threshold
                values are to be used requires user specified thresholds to
                be set";
        }
        type enumeration {
            enum MIRRORED_UP_DOWN {
                description
                    "MIRRORED_UP_DOWN indicates that a single set of
                    threshold values should be used for both increasing
                    and decreasing bandwidth when determining whether
                    to trigger updated bandwidth values to be flooded
                    in the IGP TE extensions.";
            }
            enum SEPARATE_UP_DOWN {
                description
                    "SEPARATE_UP_DOWN indicates that a separate
                    threshold values should be used for the increasing
                    and decreasing bandwidth when determining whether
                    to trigger updated bandwidth values to be flooded
                    in the IGP TE extensions.";
            }
        }
        description
            "This value specifies whether a single set of threshold
            values should be used for both increasing and decreasing
            bandwidth when determining whether to trigger updated
```

```
bandwidth values to be flooded in the IGP TE extensions.
MIRRORED-UP-DOWN indicates that a single value (or set of
values) should be used for both increasing and decreasing
values, where SEPARATE-UP-DOWN specifies that the increasing
and decreasing values will be separately specified";
}

leaf-list up-thresholds {
  when "../threshold-type = 'THRESHOLD_CROSSED'" +
    "and ../threshold-specification = 'SEPARATE_UP_DOWN'" {
    description
      "A list of up-thresholds can only be specified when the
      bandwidth update is triggered based on crossing a
      threshold and separate up and down thresholds are
      required";
  }
  type te-types:percentage;
  description
    "The thresholds (expressed as a percentage of the maximum
    reservable bandwidth) at which bandwidth updates are to be
    triggered when the bandwidth is increasing.";
}

leaf-list down-thresholds {
  when "../threshold-type = 'THRESHOLD_CROSSED'" +
    "and ../threshold-specification = 'SEPARATE_UP_DOWN'" {
    description
      "A list of down-thresholds can only be specified when the
      bandwidth update is triggered based on crossing a
      threshold and separate up and down thresholds are
      required";
  }
  type te-types:percentage;
  description
    "The thresholds (expressed as a percentage of the maximum
    reservable bandwidth) at which bandwidth updates are to be
    triggered when the bandwidth is decreasing.";
}

leaf-list up-down-thresholds {
  when "../threshold-type = 'THRESHOLD_CROSSED'" +
    "and ../threshold-specification = 'MIRRORED_UP_DOWN'" {
    description
      "A list of thresholds corresponding to both increasing
      and decreasing bandwidths can be specified only when an
      update is triggered based on crossing a threshold, and
      the same up and down thresholds are required.";
  }
}
```

```
    type te-types:percentage;
    description
      "The thresholds (expressed as a percentage of the maximum
       reservable bandwidth of the interface) at which bandwidth
       updates are flooded - used both when the bandwidth is
       increasing and decreasing";
  }
}

/* TE interface metric */
grouping te-metric_config {
  description "Interface TE metric grouping";
  leaf te-metric {
    type te-types:te-metric;
    description "Interface TE metric.";
  }
}

/* TE interface switching capabilities */
grouping te-switching-cap_config {
  description
    "TE interface switching capabilities";
  list switching-capabilities {
    key "switching-capability";
    description
      "List of interface capabilities for this interface";
    leaf switching-capability {
      type identityref {
        base te-types:switching-capabilities;
      }
      description
        "Switching Capability for this interface";
    }
    leaf encoding {
      type identityref {
        base te-types:lsp-encoding-types;
      }
      description
        "Encoding supported by this interface";
    }
  }
}

grouping te-advertisements_state {
  description
    "TE interface advertisements state grouping";
  container te-advertisements_state {
    description
```



```
    "TE interface advertisements state container";
  leaf flood-interval {
    type uint32;
    description
      "The periodic flooding interval";
  }
  leaf last-flooded-time {
    type uint32;
    units seconds;
    description
      "Time elapsed since last flooding in seconds";
  }
  leaf next-flooded-time {
    type uint32;
    units seconds;
    description
      "Time remained for next flooding in seconds";
  }
  leaf last-flooded-trigger {
    type enumeration {
      enum link-up {
        description "Link-up flooding trigger";
      }
      enum link-down {
        description "Link-up flooding trigger";
      }
      enum threshold-up {
        description
          "Bandwidth reservation up threshold";
      }
      enum threshold-down {
        description
          "Bandwidth reservation down threshold";
      }
      enum bandwidth-change {
        description "Banwidth capacity change";
      }
      enum user-initiated {
        description "Initiated by user";
      }
      enum srlg-change {
        description "SRLG property change";
      }
      enum periodic-timer {
        description "Periodic timer expired";
      }
    }
    description "Trigger for the last flood";
  }
```

```
    }
    list advertized-level-areas {
      key level-area;
      description
        "List of areas the TE interface is advertised
        in";
      leaf level-area {
        type uint32;
        description
          "The IGP area or level where the TE
          interface state is advertised in";
      }
    }
  }
}

/* TE interface attributes grouping */
grouping te-attributes {
  description "TE attributes configuration grouping";
  uses te-metric_config;
  uses te-admin-groups_config;
  uses te-srlgs_config;
  uses te-igp-flooding-bandwidth_config;
  uses te-switching-cap_config;
  container state {
    config false;
    description
      "State parameters for interface TE metric";
    uses te-advertisements_state;
  }
}

grouping te-all-attributes {
  description
    "TE attributes configuration grouping for all
    interfaces";
  uses te-igp-flooding-bandwidth_config;
}
/**** End of TE interfaces device groupings ****/

/**
 * TE device augmentations
 */
augment "/te:te" {
  description "TE global container.";
  /* TE Interface Configuration Data */
  uses interfaces-grouping;
}
```

```
}

/* TE globals device augmentation */
augment "/te:te/te:globals" {
  description
    "Global TE device specific configuration parameters";
  uses lsp-device-timers;
}

/* TE tunnels device configuration augmentation */
augment "/te:te/te:tunnels/te:tunnel" {
  description
    "Tunnel device dependent augmentation";
  uses lsp-device-timers_config;
}
augment "/te:te/te:tunnels/te:tunnel/te:state" {
  description
    "Tunnel device dependent augmentation";
  uses lsp-device-timers_config;
}

/* TE LSPs device state augmentation */
augment "/te:te/te:lsps-state/te:lsp" {
  description
    "LSP device dependent augmentation";
  uses lsps-device_state;
}

augment "/te:te/te:tunnels/te:tunnel/te:p2p-secondary-paths" +
  "/te:p2p-secondary-path/te:state/te:lsps/te:lsp" {
  description
    "LSP device dependent augmentation";
  uses lsps-device_state;
}

augment "/te:te/te:tunnels/te:tunnel/te:p2p-primary-paths" +
  "/te:p2p-primary-path/te:state/te:lsps/te:lsp" {
  description
    "LSP device dependent augmentation";
  uses lsps-device_state;
}

/* TE interfaces RPCs/execution Data */
rpc interfaces-rpc {
  description
    "Execution data for TE interfaces.";
}
```

```
/* TE Interfaces Notification Data */
notification interfaces-notif {
  description
    "Notification messages for TE interfaces.";
}
}
<CODE ENDS>
```

Figure 9: TE MPLS specific types YANG module

```
<CODE BEGINS> file "ietf-te-mpls@2018-02-15.yang"
module ietf-te-mpls {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te-mpls";

  /* Replace with IANA when assigned */
  prefix "te-mpls";

  /* Import TE base model */
  import ietf-te {
    prefix te;
  }

  /* Import TE MPLS types */
  import ietf-te-mpls-types {
    prefix "te-mpls-types";
  }

  /* Import TE generic types */
  import ietf-te-types {
    prefix te-types;
  }

  /* Import routing types */
  import ietf-routing-types {
    prefix "rt-types";
  }

  import ietf-mpls-static {
    prefix mpls-static;
  }

  import ietf-inet-types {
    prefix inet;
  }

  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>

  WG Chair:   Lou Berger
              <mailto:lberger@labn.net>

  WG Chair:   Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Tarek Saad
              <mailto:tsaad@cisco.com>

  Editor:     Rakesh Gandhi
              <mailto:rgandhi@cisco.com>

  Editor:     Vishnu Pavan Beeram
              <mailto:vbeeram@juniper.net>

  Editor:     Himanshu Shah
              <mailto:hshah@ciena.com>

  Editor:     Xufeng Liu
              <mailto:xufeng.liu@ericsson.com>

  Editor:     Xia Chen
              <mailto:jescia.chenxia@huawei.com>

  Editor:     Raqib Jones
              <mailto:raqib@Brocade.com>

  Editor:     Bin Wen
              <mailto:Bin_Wen@cable.comcast.com>";

description
  "YANG data module for MPLS TE configurations,
  state, RPC and notifications.";

revision "2018-02-15" {
  description "Latest update to MPLS TE YANG module.";
  reference "TBD";
}

/* MPLS TE tunnel properties*/

grouping tunnel-igp-shortcut_config {
```

```
description "TE tunnel IGP shortcut configs";
leaf shortcut-eligible {
  type boolean;
  default "true";
  description
    "Whether this LSP is considered to be eligible for us as a
    shortcut in the IGP. In the case that this leaf is set to
    true, the IGP SPF calculation uses the metric specified to
    determine whether traffic should be carried over this LSP";
}
leaf metric-type {
  type identityref {
    base te-types:LSP_METRIC_TYPE;
  }
  default te-types:LSP_METRIC_INHERITED;
  description
    "The type of metric specification that should be used to set
    the LSP(s) metric";
}
leaf metric {
  type int32;
  description
    "The value of the metric that should be specified. The value
    supplied in this leaf is used in conjunction with the metric
    type to determine the value of the metric used by the system.
    Where the metric-type is set to LSP_METRIC_ABSOLUTE - the
    value of this leaf is used directly; where it is set to
    LSP_METRIC_RELATIVE, the relevant (positive or negative)
    offset is used to formulate the metric; where metric-type
    is LSP_METRIC_INHERITED, the value of this leaf is not
    utilised";
}
leaf-list routing-afs {
  type inet:ip-version;
  description
    "Address families";
}
}

grouping tunnel-igp-shortcuts {
  description
    "TE tunnel IGP shortcut grouping";
  container tunnel-igp-shortcut {
    description
      "Tunnel IGP shortcut properties";
    uses tunnel-igp-shortcut_config;
  }
}
```

```
grouping tunnel-forwarding-adjacency_configs {
  description "Tunnel forwarding adjacency grouping";
  leaf binding-label {
    type rt-types:mpls-label;
    description "MPLS tunnel binding label";
  }
  leaf load-share {
    type uint32 {
      range "1..4294967295";
    }
    description "ECMP tunnel forwarding
      load-share factor.";
  }
  leaf policy-class {
    type uint8 {
      range "1..7";
    }
    description
      "The class associated with this tunnel";
  }
}

grouping tunnel-forwarding-adjacency {
  description "Properties for using tunnel in forwarding.";
  container forwarding {
    description
      "Tunnel forwarding properties container";
    uses tunnel-forwarding-adjacency_configs;
  }
}

/** End of MPLS TE tunnel configuration/state */
grouping te-lsp-auto-bandwidth_config {
  description
    "Configuration parameters related to autobandwidth";

  leaf enabled {
    type boolean;
    default false;
    description
      "enables mpls auto-bandwidth on the
        lsp";
  }

  leaf min-bw {
    type te-mpls-types:bandwidth-kbps;
    description
      "set the minimum bandwidth in Kbps for an
```

```
        auto-bandwidth LSP";
    }

    leaf max-bw {
        type te-mpls-types:bandwidth-kbps;
        description
            "set the maximum bandwidth in Kbps for an
            auto-bandwidth LSP";
    }

    leaf adjust-interval {
        type uint32;
        description
            "time in seconds between adjustments to
            LSP bandwidth";
    }

    leaf adjust-threshold {
        type te-types:percentage;
        description
            "percentage difference between the LSP's
            specified bandwidth and its current bandwidth
            allocation -- if the difference is greater than the
            specified percentage, auto-bandwidth adjustment is
            triggered";
    }
}

grouping te-lsp-overflow_config {
    description
        "configuration for mpls lsp bandwidth
        overflow adjustment";

    leaf enabled {
        type boolean;
        default false;
        description
            "enables mpls lsp bandwidth overflow
            adjustment on the lsp";
    }

    leaf overflow-threshold {
        type te-types:percentage;
        description
            "bandwidth percentage change to trigger
            an overflow event";
    }
}
```



```
    leaf trigger-event-count {
      type uint16;
      description
        "number of consecutive overflow sample
        events needed to trigger an overflow adjustment";
    }
  }

  grouping te-lsp-underflow_config {
    description
      "configuration for mpls lsp bandwidth
      underflow adjustment";

    leaf enabled {
      type boolean;
      default false;
      description
        "enables bandwidth underflow
        adjustment on the lsp";
    }

    leaf underflow-threshold {
      type te-types:percentage;
      description
        "bandwidth percentage change to trigger
        and underflow event";
    }

    leaf trigger-event-count {
      type uint16;
      description
        "number of consecutive underflow sample
        events needed to trigger an underflow adjustment";
    }
  }

  grouping te-tunnel-bandwidth_config {
    description
      "Configuration parameters related to bandwidth for a tunnel";

    leaf specification-type {
      type te-mpls-types:te-bandwidth-type;
      default SPECIFIED;
      description
        "The method used for settign the bandwidth, either explicitly
        specified or configured";
    }

    leaf set-bandwidth {
```

```
    when "../specification-type = 'SPECIFIED'" {
      description
        "The bandwidth value when bandwidth is explicitly
        specified";
    }
    type te-mpls-types:bandwidth-kbps;
    description
      "set bandwidth explicitly, e.g., using
      offline calculation";
  }
  leaf class-type {
    type te-types:te-ds-class;
    description
      "The Class-Type of traffic transported by the LSP.";
    reference "RFC4124: section-4.3.1";
  }
}

grouping te-tunnel-bandwidth_state {
  description
    "Operational state parameters relating to bandwidth for a tunnel";

  leaf signaled-bandwidth {
    type te-mpls-types:bandwidth-kbps;
    description
      "The currently signaled bandwidth of the LSP. In the case where
      the bandwidth is specified explicitly, then this will match the
      value of the set-bandwidth leaf; in cases where the bandwidth is
      dynamically computed by the system, the current value of the
      bandwidth should be reflected.";
  }
}

grouping tunnel-bandwidth_top {
  description
    "Top level grouping for specifying bandwidth for a tunnel";

  container bandwidth-mpls {
    description
      "Bandwidth configuration for TE LSPs";

    uses te-tunnel-bandwidth_config;

    container state {
      config false;
      description
        "State parameters related to bandwidth
        configuration of TE tunnels";
    }
  }
}
```

```
    uses te-tunnel-bandwidth_state;
  }

  container auto-bandwidth {
    when "../specification-type = 'AUTO'" {
      description
        "Include this container for auto bandwidth
        specific configuration";
    }
    description
      "Parameters related to auto-bandwidth";

    uses te-lsp-auto-bandwidth_config;

    container overflow {
      description
        "configuration of MPLS overflow bandwidth
        adjustment for the LSP";

      uses te-lsp-overflow_config;
    }

    container underflow {
      description
        "configuration of MPLS underflow bandwidth
        adjustment for the LSP";

      uses te-lsp-underflow_config;
    }
  }
}

grouping te-path-bandwidth_top {
  description
    "Top level grouping for specifying bandwidth for a TE path";

  container bandwidth {
    description
      "Bandwidth configuration for TE LSPs";

    uses te-tunnel-bandwidth_config;
    container state {
      config false;
      description
        "State parameters related to bandwidth
        configuration of TE tunnels";
      uses te-tunnel-bandwidth_state;
    }
  }
}
```

```
    }
  }
}

/**
 * MPLS TE augmentations
 */

/* MPLS TE tunnel augmentations */
augment "/te:te/te:tunnels/te:tunnel" {
  description "MPLS TE tunnel config augmentations";
  uses tunnel-igp-shortcuts;
  uses tunnel-forwarding-adjacency;
  uses tunnel-bandwidth_top;
}

/* MPLS TE LSPs augmentations */
augment "/te:te/te:tunnels/te:tunnel/" +
  "te:p2p-primary-paths/te:p2p-primary-path" {
  when "/te:te/te:tunnels/te:tunnel" +
    "/te:p2p-primary-paths/te:p2p-primary-path" +
    "/te:path-setup-protocol = 'te-types:path-setup-static'" {
    description
      "When the path is statically provisioned";
  }
  description "MPLS TE LSP augmentation";
  leaf static-lsp-name {
    type mpls-static:static-lsp-ref;
    description "Static LSP name";
  }
}

augment "/te:te/te:tunnels/te:tunnel/" +
  "te:p2p-primary-paths/te:p2p-primary-path/" +
  "te:state" {
  description "MPLS TE LSP augmentation";
  leaf static-lsp-name {
    type mpls-static:static-lsp-ref;
    description "Static LSP name";
  }
}

augment "/te:te/te:tunnels/te:tunnel/" +
  "te:p2p-secondary-paths/te:p2p-secondary-path" {
  when "/te:te/te:tunnels/te:tunnel" +
    "/te:p2p-secondary-paths/te:p2p-secondary-path/" +
    "te:path-setup-protocol = 'te-types:path-setup-static'" {
```

```

        description
          "When the path is statically provisioned";
      }
      description "MPLS TE LSP augmentation";
      leaf static-lsp-name {
        type mpls-static:static-lsp-ref;
        description "Static LSP name";
      }
    }
    augment "/te:te/te:tunnels/te:tunnel/" +
      "te:p2p-secondary-paths/te:p2p-secondary-path/" +
      "te:state" {
      description "MPLS TE LSP augmentation";
      leaf static-lsp-name {
        type mpls-static:static-lsp-ref;
        description "Static LSP name";
      }
    }

    augment "/te:te/te:globals/te:named-path-constraints/" +
      "te:named-path-constraint" {
      description "foo";
      uses te-path-bandwidth_top;
    }
  }
}
<CODE ENDS>

```

Figure 10: TE MPLS YANG module

```

<CODE BEGINS> file "ietf-te-mpls-types@2018-02-15.yang"
module ietf-te-mpls-types {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te-mpls-types";

  /* Replace with IANA when assigned */
  prefix "te-mpls-types";

  organization
    "IETF TEAS Working Group";

  contact "Fill me";

  description
    "This module contains a collection of generally
    useful TE specific YANG data type defintions.";

  revision "2018-02-15" {
    description "Latest revision of TE MPLS types";
  }
}

```

```
    reference "RFC3209";
}

identity backup-protection-type {
    description
        "Base identity for backup protection type";
}

identity backup-protection-link {
    base backup-protection-type;
    description
        "backup provides link protection only";
}

identity backup-protection-node-link {
    base backup-protection-type;
    description
        "backup offers node (preferred) or link protection";
}

identity bc-model-type {
    description
        "Base identity for Diffserv-TE bandwidth constraint
        model type";
}

identity bc-model-rdm {
    base bc-model-type;
    description
        "Russian Doll bandwidth constraint model type.";
}

identity bc-model-mam {
    base bc-model-type;
    description
        "Maximum Allocation bandwidth constraint
        model type.";
}

identity bc-model-mar {
    base bc-model-type;
    description
        "Maximum Allocation with Reservation
        bandwidth constraint model type.";
}

typedef bandwidth-kbps {
    type uint64;
```

```
    units "Kbps";
    description
        "Bandwidth values expressed in kilobits per second";
}

typedef bandwidth-mbps {
    type uint64;
    units "Mbps";
    description
        "Bandwidth values expressed in megabits per second";
}

typedef bandwidth-gbps {
    type uint64;
    units "Gbps";
    description
        "Bandwidth values expressed in gigabits per second";
}

typedef te-bandwidth-type {
    type enumeration {
        enum SPECIFIED {
            description
                "Bandwidth is explicitly specified";
        }
        enum AUTO {
            description
                "Bandwidth is automatically computed";
        }
    }
    description
        "enumerated type for specifying whether bandwidth is
        explicitly specified or automatically computed";
}

typedef bfd-type {
    type enumeration {
        enum classical {
            description "BFD classical session type.";
        }
        enum seamless {
            description "BFD seamless session type.";
        }
    }
    default "classical";
    description
        "Type of BFD session";
}
```

```
typedef bfd-encap-mode-type {  
  type enumeration {  
    enum gal {  
      description  
        "BFD with GAL mode";  
    }  
    enum ip {  
      description  
        "BFD with IP mode";  
    }  
  }  
  default ip;  
  description  
    "Possible BFD transport modes when running over TE  
    LSPs.";  
}  
}  
<CODE ENDS>
```

Figure 11: TE MPLS types YANG module

```
<CODE BEGINS> file "ietf-te-sr-mpls@2018-02-15.yang"  
module ietf-te-sr-mpls {  
  
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls";  
  
  /* Replace with IANA when assigned */  
  prefix "te-sr-mpls";  
  
  /* Import TE generic types */  
  import ietf-te {  
    prefix te;  
  }  
  
  /* Import TE generic types */  
  import ietf-te-types {  
    prefix te-types;  
  }  
  
  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>  
  
    WG Chair: Lou Berger
```


<mailto:lberger@labn.net>

WG Chair: Vishnu Pavan Beeram
<mailto:vbeeram@juniper.net>

Editor: Tarek Saad
<mailto:tsaad@cisco.com>

Editor: Rakesh Gandhi
<mailto:rgandhi@cisco.com>

Editor: Vishnu Pavan Beeram
<mailto:vbeeram@juniper.net>

Editor: Himanshu Shah
<mailto:hshah@ciena.com>

Editor: Xufeng Liu
<mailto:xufeng.liu@ericsson.com>

Editor: Xia Chen
<mailto:jescia.chenxia@huawei.com>

Editor: Raqib Jones
<mailto:raqib@Brocade.com>

Editor: Bin Wen
<mailto:Bin_Wen@cable.comcast.com>;

description

"YANG data module for MPLS TE configurations,
state, RPC and notifications.";

```
revision "2018-02-15" {  
  description "Latest update to MPLS TE YANG module.";  
  reference "TBD";  
}
```

```
identity sr-protection-type {  
  description  
    "The Adj-SID base protection types";  
}
```

```
identity sr-protection-type-protected {  
  base sr-protection-type;  
  description  
    "The Adj-SID is eligible if protected";  
}
```

```
identity sr-protection-type-unprotected {
  base sr-protection-type;
  description
    "The Adj-SID is eligible if unprotected";
}

identity sr-protection-type-any {
  base sr-protection-type;
  description
    "The Adj-SID is eligible if protected or unprotected";
}

typedef te-sid-selection-mode {
  type enumeration {
    enum ADJ_SID_ONLY {
      description
        "The SR-TE tunnel should only use adjacency SIDs
        to build the SID stack to be pushed for the LSP";
    }
    enum MIXED_MODE {
      description
        "The SR-TE tunnel can use a mix of adjacency
        and prefix SIDs to build the SID stack to be pushed
        to the LSP";
    }
  }
  description "SID selection mode type";
}

/* MPLS SR-TE tunnel properties*/
grouping tunnel-sr-mpls-properties_config {
  description "MPLS TE SR tunnel properties";
  leaf path-signaling-type {
    type identityref {
      base te-types:path-signaling-type;
    }
    description "TE tunnel path signaling type";
  }
}

grouping te-sr-named-path-constraints_config {
  description
    "Configuration parameters relating to SR-TE LSPs";

  leaf sid-selection-mode {
    type te-sid-selection-mode;
    default MIXED_MODE;
    description

```

```

        "The restrictions placed on the SIDs to be selected by the
        calculation method for the explicit path when it is
        instantiated for a SR-TE LSP";
    }

    leaf sid-protection {
        type identityref {
            base sr-protection-type;
        }
        default sr-protection-type-any;
        description
            "When set to protected only SIDs that are
            protected are to be selected by the calculating method
            when the explicit path is instantiated by a SR-TE LSP.";
    }
}

grouping te-sr-named-path-constraints {
    description "Named TE SR path constraints grouping";
    uses te-sr-named-path-constraints_config;
}

/** End of MPLS SR-TE tunnel configuration/state */

/**
 * MPLS TE augmentations
 */
augment "/te:te/te:globals/te:named-path-constraints" +
    "/te:named-path-constraint" {
    description
        "Augmentations for MPLS SR-TE config named constraints";
    uses te-sr-named-path-constraints;
}

/* MPLS TE tunnel augmentations */

/* MPLS TE LSPs augmentations */
}
<CODE ENDS>

```

Figure 12: SR TE MPLS YANG module

5. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-te XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-device XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-mpls XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-types XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-te-mpls-types XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-te namespace: urn:ietf:params:xml:ns:yang:ietf-te prefix: ietf-te reference: RFC3209

name: ietf-te-device namespace: urn:ietf:params:xml:ns:yang:ietf-te prefix: ietf-te-device reference: RFC3209

name: ietf-te-mpls namespace: urn:ietf:params:xml:ns:yang:ietf-te-mpls prefix: ietf-te-mpls reference: RFC3209

name: ietf-te-sr-mpls namespace: urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls prefix: ietf-te-sr-mpls

name: ietf-te-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-types prefix: ietf-te-types reference: RFC3209

name: ietf-te-mpls-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-mpls-types prefix: ietf-te-mpls-types reference: RFC3209

6. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF

users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which 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. Following are the subtrees and data nodes and their sensitivity/vulnerability:

"/te/globals": This module specifies the global TE configurations on a device. Unauthorized access to this container could cause the device to ignore packets it should receive and process.

"/te/tunnels": This list specifies the configured TE tunnels on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

"/te/lsp-state": This list specifies the state derived LSPs. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

"/te/interfaces": This list specifies the configured TE interfaces on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

7. Acknowledgement

The authors would like to thank the members of the multi-vendor YANG design team who are involved in the definition of this model.

The authors would also like to thank Loa Andersson, Lou Berger, Sergio Belotti, Italo Busi, Carlo Perocchio, Francesco Lazzeri, Aihua Guo, Dhruv Dhody, Anurag Sharma, and Xian Zhang for their comments and providing valuable feedback on this document.

8. Contributors

Xia Chen
Huawei Technologies

Email: jescia.chenxia@huawei.com

Raqib Jones
Brocade

Email: raqib@Brocade.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com

9. Normative References

- [I-D.ietf-teas-yang-rsvp]
Beeram, V., Saad, T., Gandhi, R., Liu, X., Bryskin, I.,
and H. Shah, "A YANG Data Model for Resource Reservation
Protocol (RSVP)", draft-ietf-teas-yang-rsvp-08 (work in
progress), October 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP
Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001,
<<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label
Switching (GMPLS) Signaling Resource ReserVation Protocol-
Traffic Engineering (RSVP-TE) Extensions", RFC 3473,
DOI 10.17487/RFC3473, January 2003,
<<https://www.rfc-editor.org/info/rfc3473>>.
- [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>>.
- [RFC6107] Shiimoto, K., Ed. and A. Farrel, Ed., "Procedures for Dynamically Signaled Hierarchical Label Switched Paths", RFC 6107, DOI 10.17487/RFC6107, February 2011, <<https://www.rfc-editor.org/info/rfc6107>>.
- [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>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [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>>.

Authors' Addresses

Tarek Saad (editor)
Cisco Systems Inc

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems Inc

Email: rgandhi@cisco.com

Xufeng Liu
Jabil

Email: Xufeng_Liu@jabil.com

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Himanshu Shah
Ciena

Email: hshah@ciena.com

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

TEAS Working Group
Internet Draft
Intended Status: Standard Track
Expires: August 27, 2018

Y. Lee (Editor)
Dhruv Dhody
Huawei
D. Ceccarelli
Ericsson
Igor Bryskin
Huawei
Bin Yeong Yoon
ETRI

February 27, 2018

A Yang Data Model for ACTN VN Operation

draft-lee-teas-actn-vn-yang-12

Abstract

This document provides a YANG data model for the Abstraction and Control of Traffic Engineered (TE) networks (ACTN) Virtual Network Service (VNS) operation.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on August 27, 2018.

Copyright Notice
Lee, et al.

Expires August 2018

[Page 1]

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction.....	3
1.1. Terminology.....	4
2. ACTN CMI context.....	4
2.1. Type 1 VN.....	4
2.2. Type 2 VN.....	5
3. High-Level Control Flows with Examples.....	6
3.1. Type 1 VN Illustration.....	6
3.2. Type 2 VN Illustration.....	8
4. Justification of the ACTN VN Model on the CMI.....	10
4.1. Customer view of VN.....	10
4.2. Innovative Services.....	10
4.2.1. VN Compute.....	10
4.2.2. Multi-sources and Multi-destinations.....	11
4.2.3. Others.....	11
4.3. Summary.....	12
5. ACTN VN YANG Model (Tree Structure).....	12
6. ACTN-VN YANG Code.....	15
7. JSON Example.....	27
7.1. ACTN VN JSON.....	28
7.2. TE-topology JSON.....	33
8. Security Considerations.....	49
9. IANA Considerations.....	50
10. Acknowledgments.....	50
11. References.....	51
11.1. Normative References.....	51
11.2. Informative References.....	51
12. Contributors.....	52

Authors' Addresses.....	52
-------------------------	----

1. Introduction

This document provides a YANG data model for the Abstraction and Control of Traffic Engineered (TE) networks (ACTN) Virtual Network Service (VNS) operation that is going to be implemented for the Customer Network Controller (CNC)- Multi-Domain Service Coordinator (MSDC) interface (CMI).

The YANG model on the CMI is also known as customer service model in [Service-YANG]. The YANG model discussed in this document is used to operate customer-driven VNs during the VN computation, VN instantiation and its life-cycle management and operations.

The YANG model discussed in this document basically provides the following:

- o Characteristics of Access Points (APs) that describe customer's end point characteristics;
- o Characteristics of Virtual Network Access Points (VNAP) that describe How an AP is partitioned for multiple VNs sharing the AP and its reference to a Link Termination Point (LTP) of the Provider Edge (PE) Node;
- o Characteristics of Virtual Networks (VNs) that describe the customer's VNs in terms of VN Members comprising a VN, multi-source and/or multi-destination characteristics of VN Member, the VN's reference to TE-topology's Abstract Node;

The actual VN instantiation is performed with Connectivity Matrices sub-module of TE-Topology Model [TE-Topo] which interacts with the VN YANG module presented in this draft. Once TE-topology Model is used in triggering VN instantiation over the networks, TE-tunnel [TE-tunnel] Model will inevitably interact with TE-Topology model for setting up actual tunnels and LSPs under the tunnels.

The ACTN VN operational state is included in the same tree as the configuration consistent with Network Management Datastore Architecture (NMDA) [NMDA]. The origin of the data is indicated as per the origin metadata annotation.

1.1. Terminology

Refer to [ACTN-Frame] and [RFC7926] for the key terms used in this document.

2. ACTN CMI context

The model presented in this document has the following ACTN context.

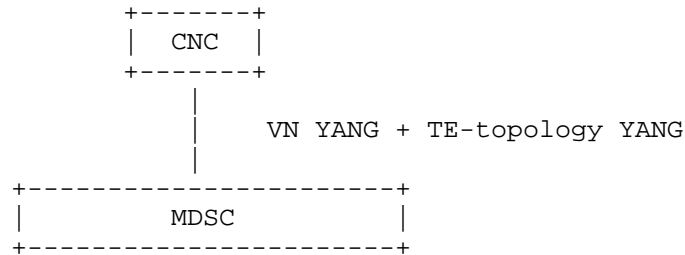


Figure 1. ACTN CMI

Both ACTN VN YANG and TE-topology models are used over the CMI to establish a VN over TE networks.

2.1. Type 1 VN

As defined in [ACTN-FW], a Virtual Network is a customer view of the TE network. To recapitulate VN types from [ACTN-FW], Type 1 VN is defined as follows:

The VN can be seen as a set of edge-to-edge links (a Type 1 VN). Each link is referred to as a VN member and is formed as an end-to-end tunnel across the underlying networks. Such tunnels may be constructed by recursive slicing or abstraction of paths in the underlying networks and can encompass edge points of the customer's network, access links, intra-domain paths, and inter-domain links.

If we were to create a VN where we have four VN-members as follows:

VN-Member 1	L1-L4
VN-Member 2	L1-L7
VN-Member 3	L2-L4
VN-Member 4	L3-L8

Where L1, L2, L3, L4, L7 and L8 correspond to a Customer End-Point, respectively.

This VN can be modeled as one abstract node representation as follows in Figure 2:

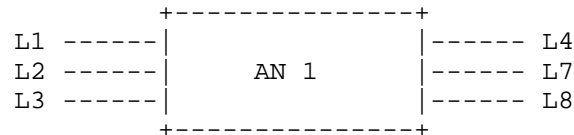


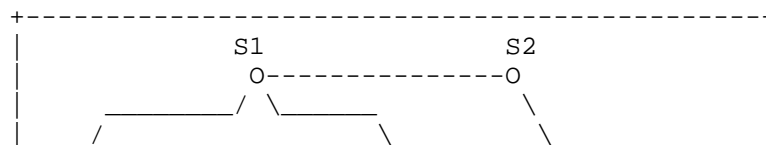
Figure 2. Abstract Node (One node topology)

Modeling a VN as one abstract node is the easiest way for customers to express their end-to-end connectivity; however, customers are not limited to express their VN only with one abstract node. In some cases, more than one abstract nodes can be employed to express their VN.

2.2. Type 2 VN

For some VN members of a VN, the customers are allowed to configure the actual path (i.e., detailed virtual nodes and virtual links) over the VN/abstract topology agreed mutually between CNC and MDSC prior to or a topology created by the MDSC as part of VN instantiation. Type 2 VN is always built on top of a Type 1 VN.

If a Type 2 VN is desired for some or all of VN members of a type 1 VN (see the example in Section 2.1), the TE-topology model can provide the following abstract topology (that consists of virtual nodes and virtual links) which is built on top of the Type 1 VN.



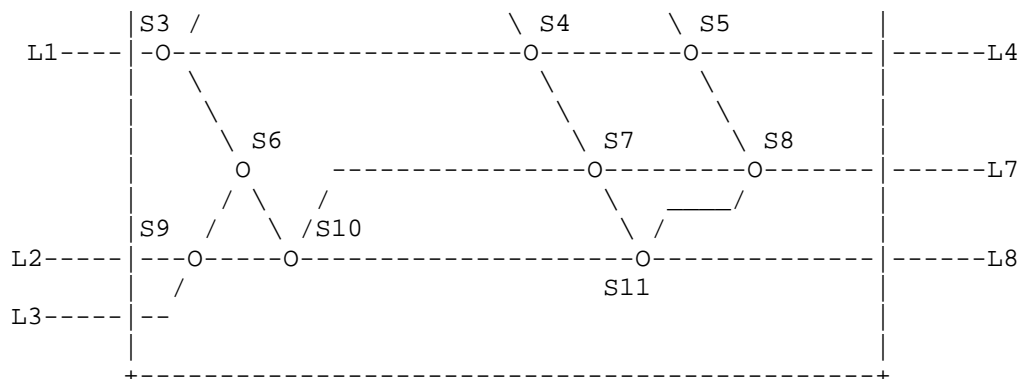


Figure 3. Type 2 topology

As you see from Figure 3, the Type 1 abstract node is depicted as a Type 1 abstract topology comprising of detailed virtual nodes and virtual links.

As an example, if VN-member 1 (L1-L4) is chosen to configure its own path over Type 2 topology, it can select, say, a path that consists of the ERO {S3,S4,S5} based on the topology and its service requirement. This capability is enacted via TE-topology configuration by the customer.

3. High-Level Control Flows with Examples

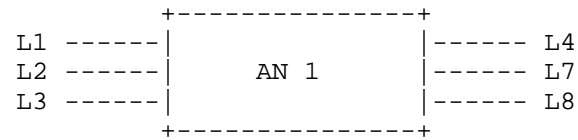
3.1. Type 1 VN Illustration

If we were to create a VN where we have four VN-members as follows:

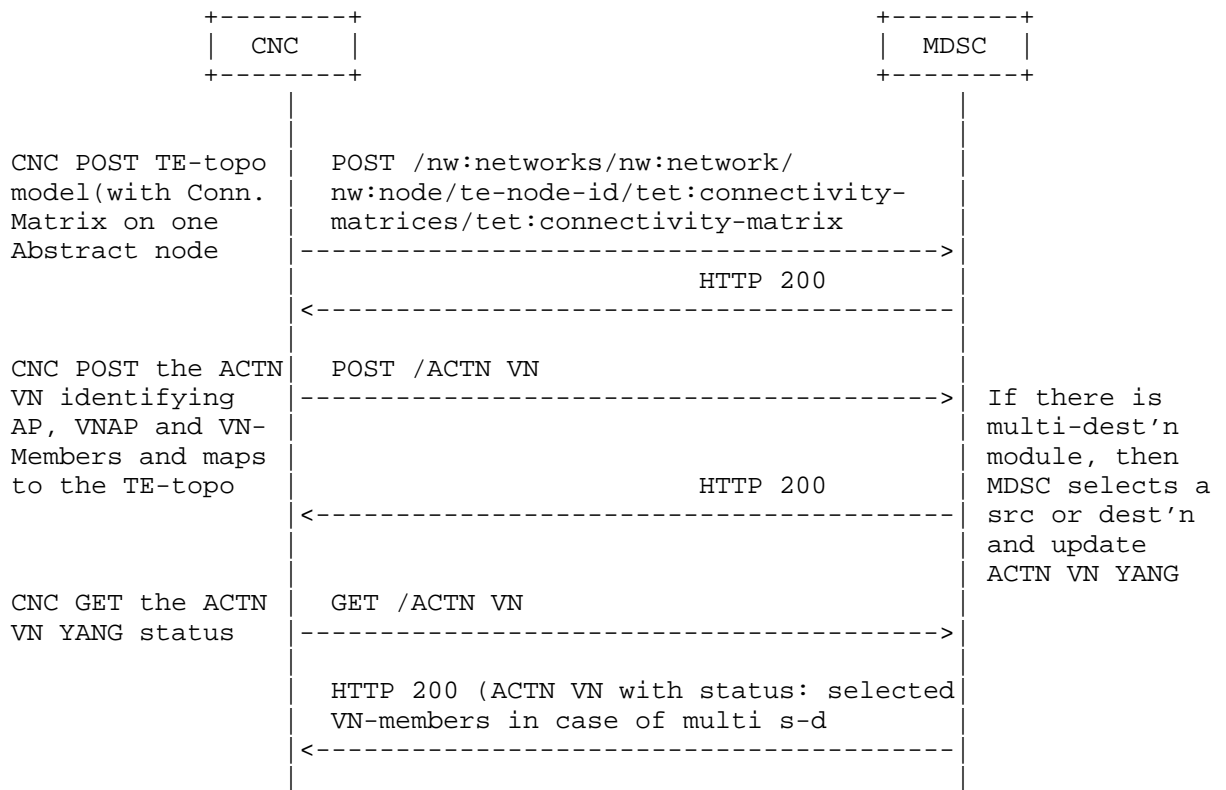
VN-Member 1	L1-L4
VN-Member 2	L1-L7
VN-Member 3	L2-L4
VN-Member 4	L3-L8

Where L1, L2, L3, L4, L7 and L8 correspond to Customer End-Point, respectively.

This VN can be modeled as one abstract node representation as follows:



If this VN is Type 1, the following diagram shows the message flow between CNC and MDSC to instantiate this VN using ACTN VN and TE-Topology Model.

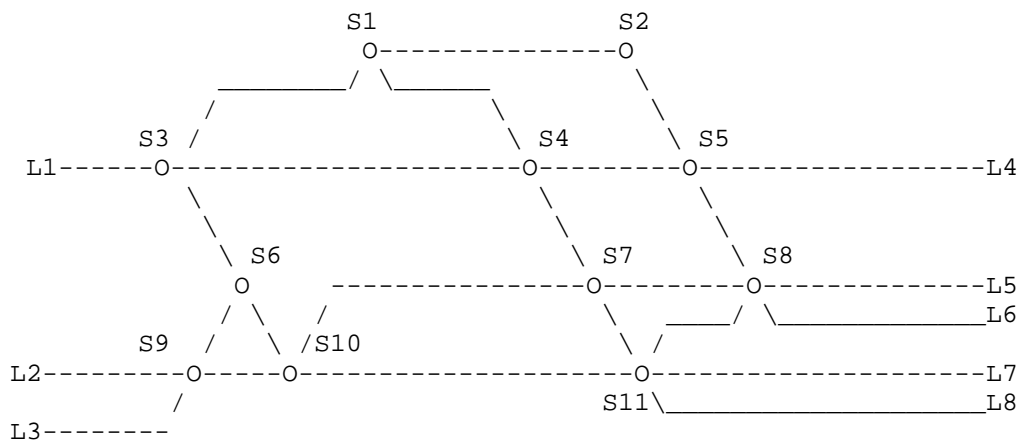


3.2. Type 2 VN Illustration

For some VN members, the customer may want to "configure" explicit routes over the path that connects its two end-points. Let us consider the following example.

VN-Member 1	L1-L4
VN-Member 2	L1-L7 (via S4 and S7)
VN-Member 3	L2-L4
VN-Member 4	L3-L8 (via S10)

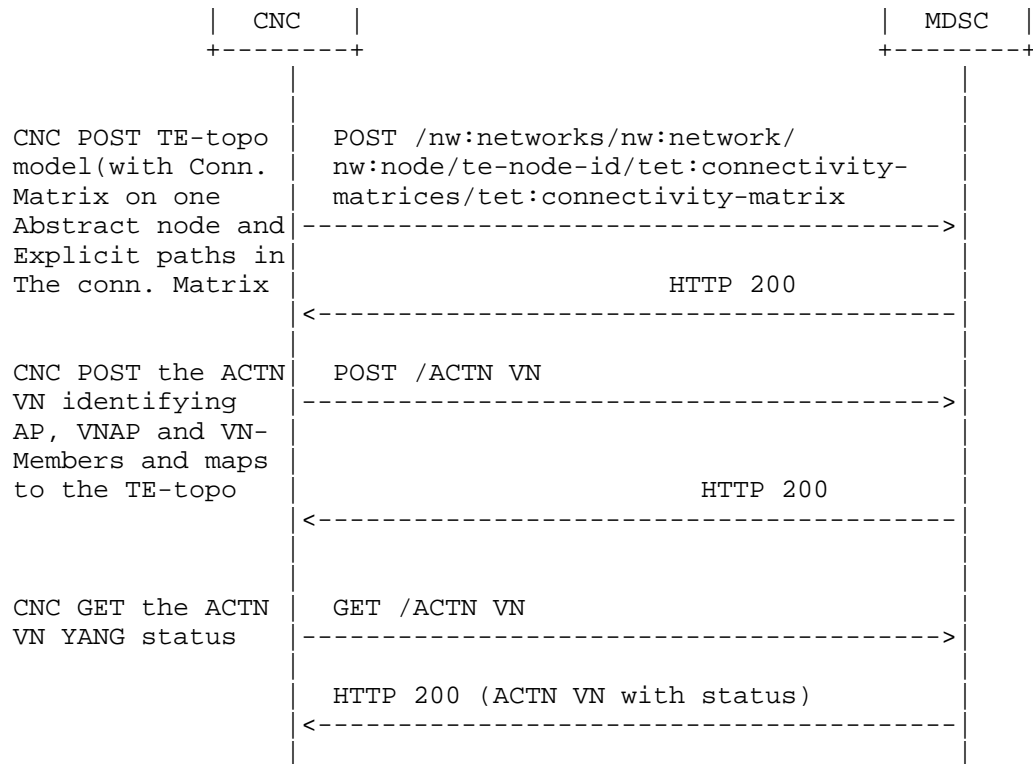
Where the following topology is the underlay for Abstraction Node 1 (AN1).



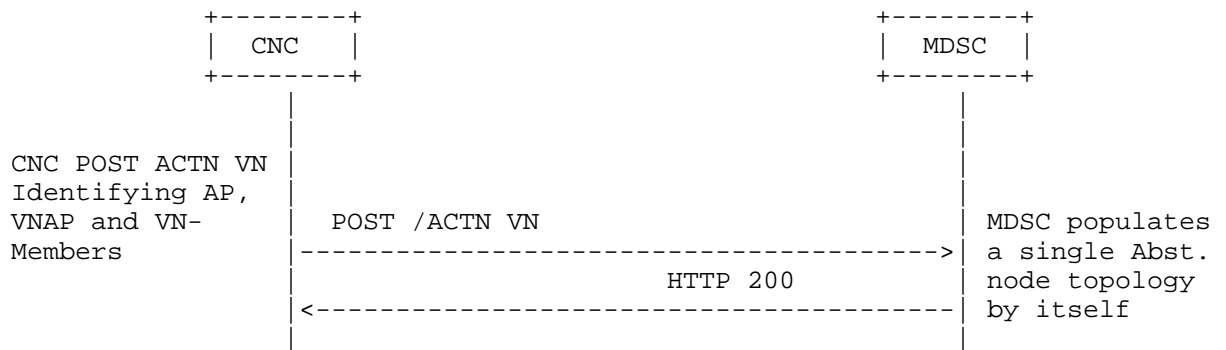
If CNC creates the single abstract topology, the following diagram shows the message flow between CNC and MDSC to instantiate this VN using ACTN VN and TE-Topology Model.

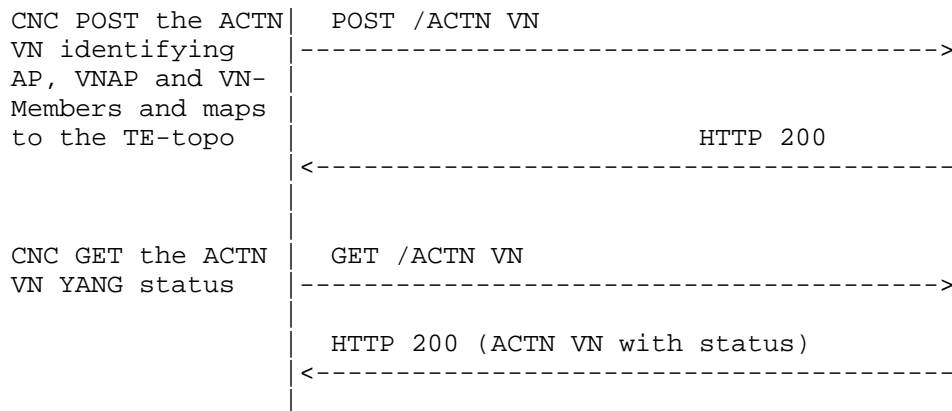
+-----+

+-----+



On the other hand, if MDSC create single node topology based ACTN VN YANG posted by the CNC, the following diagram shows the message flow between CNC and MDSC to instantiate this VN using ACTN VN and TE-Topology Model.





4. Justification of the ACTN VN Model on the CMI.

4.1. Customer view of VN

The VN-Yang model allows to define a customer view, and allows the customer to communicate using the VN constructs as described in the [ACTN-INFO]. It also allows to group the set of edge-to-edge links (i.e., VN members) under a common umbrella of VN. This allows the customer to instantiate and view the VN as one entity, making it easier for some customers to work on VN without worrying about the details of the provider based YANG models.

This is similar to the benefits of having a separate YANG model for the customer services as described in [SERVICE-YANG], which states that service models do not make any assumption of how a service is actually engineered and delivered for a customer.

4.2. Innovative Services

4.2.1. VN Compute

ACTN VN supports VN compute (pre-instantiation mode) to view the full VN as a single entity before instantiation. Achieving this via path computation or "compute only" tunnel setup does not provide the same functionality.

4.2.2. Multi-sources and Multi-destinations

In creating a virtual network, the list of sources or destinations or both may not be pre-determined by the customer. For instance, for a given source, there may be a list of multiple-destinations to which the optimal destination may be chosen depending on the network resource situations. Likewise, for a given destination, there may also be multiple-sources from which the optimal source may be chosen. In some cases, there may be a pool of multiple sources and destinations from which the optimal source-destination may be chosen. The following YANG module is shown for describing source container and destination container. The following YANG tree shows how to model multi-sources and multi-destinations.

```

+--rw actn
  . . .
  +--rw vn
    +--rw vn-list* [vn-id]
      +--rw vn-id          uint32
      +--rw vn-name?       string
      +--rw vn-topology-id? te-types:te-topology-id
      +--rw abstract-node?  -> /nw:networks/network/node/tet:te-node-id
      +--rw vn-member-list* [vn-member-id]
        | +--rw vn-member-id          uint32
        | +--rw src
        | | +--rw src?                -> /actn/ap/access-point-list/access-po
int-id      | | +--rw src-vn-ap-id?    -> /actn/ap/access-point-list/vn-ap/vn-
ap-id      | | +--rw multi-src?        boolean {multi-src-dest}?
            | | +--rw dest
            | | +--rw dest?            -> /actn/ap/access-point-list/access-p
oint-      | | +--rw dest-vn-ap-id?    -> /actn/ap/access-point-list/vn-ap/vn
id         | | +--rw multi-dest?        boolean {multi-src-dest}?
            | | +--rw connetivity-matrix-id? -> /nw:networks/network/node/tet:
te/te-      | +--ro oper-status?        identityref
node-attributes/connectivity-matrices/connectivity-matrix/id
            | +--ro if-selected?        boolean {multi-src-dest}?
            +--rw admin-status?        identityref
            +--ro oper-status?        identityref

```

4.2.3. Others

The VN Yang model can be easily augmented to support the mapping of VN to the Services such as L3SM and L2SM as described in [TE-MAP].

The VN Yang model can be extended to support telemetry, performance monitoring and network autonomies as described in [ACTN-PM].

4.3. Summary

This section summarizes the innovative service features of the ACTN VN Yang.

- o Maintenance of AP and VNAP along with VN.
- o VN construct to group of edge-to-edge links
- o VN Compute (pre-instantiate)
- o Multi-Source / Multi-Destination
- o Ability to support various VN and VNS Types
 - * VN Type 1: Customer configures the VN as a set of VN Members.
No other details need to be set by customer, making for a simplified operations for the customer.
 - * VN Type 2: Along with VN Members, the customer could also provide an abstract topology, this topology is provided by the Abstract TE Topology Yang Model.

5. ACTN VN YANG Model (Tree Structure)

```
module: ietf-actn-vn
  +--rw actn
```

```

+--rw ap
|   +--rw access-point-list* [access-point-id]
|   |   +--rw access-point-id      uint32
|   |   +--rw access-point-name?   string
|   |   +--rw max-bandwidth?       te-types:te-bandwidth
|   |   +--rw avl-bandwidth?       te-types:te-bandwidth
|   |   +--rw vn-ap* [vn-ap-id]
|   |   |   +--rw vn-ap-id          uint32
|   |   |   +--rw vn?               -> /actn/vn/vn-list/vn-id
|   |   |   +--rw abstract-node?   ->
|   |   +--rw ltp?                 te-types:te-tp-id
|   +--rw vn
|   |   +--rw vn-list* [vn-id]
|   |   |   +--rw vn-id              uint32
|   |   |   +--rw vn-name?           string
|   |   |   +--rw vn-topology-id?    te-types:te-topology-id
|   |   |   +--rw abstract-node?     ->
|   |   +--rw vn-member-list* [vn-member-id]
|   |   |   +--rw vn-member-id        uint32
|   |   |   +--rw src
|   |   |   |   +--rw src?           -> /actn/ap/access-point-
|   |   |   |   |   +--rw src-vn-ap-id? -> /actn/ap/access-point-
|   |   |   |   |   |   +--rw multi-src? boolean {multi-src-dest}?
|   |   |   |   |   |   +--rw dest
|   |   |   |   |   |   |   +--rw dest? -> /actn/ap/access-point-
|   |   |   |   |   |   |   |   +--rw dest-vn-ap-id? -> /actn/ap/access-point-
|   |   |   |   |   |   |   |   |   +--rw multi-dest? boolean {multi-src-dest}?
|   |   |   |   |   |   |   |   |   +--rw connetivity-matrix-id? ->
|   |   |   |   |   |   |   |   |   |   +--ro oper-status?          identityref
|   |   |   |   |   |   |   |   |   |   +--ro if-selected?          boolean {multi-src-dest}?
|   |   |   |   |   |   |   |   |   |   +--rw admin-status?          identityref
|   |   |   |   |   |   |   |   |   |   +--ro oper-status?          identityref
|   |   |   |   |   |   |   |   |   |   +--rw vn-level-diversity?    vn-disjointness
|   |   +--rw vn-level-diversity?    vn-disjointness
|   +--rw vn-level-diversity?    vn-disjointness
+--rw vn-level-diversity?    vn-disjointness

```

```

rpcs:
  +---x vn-compute
    +---w input
      | +---w abstract-node?      ->
/nw:networks/network/node/tet:te-node-id
      | +---w vn-member-list* [vn-member-id]
      | | +---w vn-member-id      uint32
      | | +---w src
      | | | +---w src?            -> /actn/ap/access-point-
list/access-point-id
      | | | +---w src-vn-ap-id?   -> /actn/ap/access-point-
list/vn-ap/vn-ap-id
      | | | +---w multi-src?      boolean {multi-src-dest}?
      | | | +---w dest
      | | | +---w dest?          -> /actn/ap/access-point-
list/access-point-id
      | | | +---w dest-vn-ap-id?  -> /actn/ap/access-point-
list/vn-ap/vn-ap-id
      | | | +---w multi-dest?     boolean {multi-src-dest}?
      | | | +---w connetivity-matrix-id? ->
/nw:networks/network/node/tet:te/te-node-attributes/connectivity-
matrices/connectivity-matrix/id
      | +---w vn-level-diversity?  vn-disjointness
  +--ro output
    +--ro vn-member-list* [vn-member-id]
    +--ro vn-member-id      uint32
    +--ro src
    | +--ro src?            -> /actn/ap/access-point-
list/access-point-id
    | +--ro src-vn-ap-id?   -> /actn/ap/access-point-
list/vn-ap/vn-ap-id
    | +--ro multi-src?      boolean {multi-src-dest}?
    +--ro dest
    | +--ro dest?          -> /actn/ap/access-point-
list/access-point-id
    | +--ro dest-vn-ap-id?  -> /actn/ap/access-point-
list/vn-ap/vn-ap-id
    | +--ro multi-dest?     boolean {multi-src-dest}?

```

```

        +--ro connetivity-matrix-id?    ->
/nw:networks/network/node/tet:te/te-node-attributes/connectivity-
matrices/connectivity-matrix/id
        +--ro if-selected?              boolean {multi-src-
dest}?
        +--ro compute-status?           identityref

```

6. ACTN-VN YANG Code

The YANG code is as follows:

```

<CODE BEGINS> file "ietf-actn-vn@2018-02-27.yang"

module ietf-actn-vn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-actn-vn";
  prefix "vn";

  /* Import network */
  import ietf-network {
    prefix "nw";
  }

  /* Import TE generic types */
  import ietf-te-types {
    prefix "te-types";
  }

  /* Import Abstract TE Topology */
  import ietf-te-topology {
    prefix "tet";
  }

  organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
    Working Group";
  contact
    "Editor: Young Lee <leeyoung@huawei.com>
    : Dhruv Dhody <dhruv.ietf@gmail.com>";
  description
    "This module contains a YANG module for the ACTN VN. It
    describes a VN operation module that takes place in the
    context of the CNC-MDSC Interface (CMI) of the ACTN
    architecture where the CNC is the actor of a VN

```

```
    Instantiation/modification /deletion.";
revision 2018-02-27 {
    description
        "initial version.";
    reference
        "TBD";
}
/*
 * Features
 */
feature multi-src-dest {
    description
        "Support for selection of one src or destination
        among multiple.";
}

/*identity path-metric-delay {
    base te-types:path-metric-type;
    description
        "delay path metric";
}
identity path-metric-delay-variation {
    base te-types:path-metric-type;
    description
        "delay-variation path metric";
}
identity path-metric-loss {
    base te-types:path-metric-type;
    description
        "loss path metric";
}*/

identity vn-state-type {
    description
        "Base identity for VN state";
}
identity vn-state-up {
    base vn-state-type;
    description "VN state up";
}
identity vn-state-down {
    base vn-state-type;
    description "VN state down";
}
identity vn-admin-state-type {
```



```
        description
            "Base identity for VN admin states";
    }
    identity vn-admin-state-up {
        base vn-admin-state-type;
        description "VN administratively state up";
    }
    identity vn-admin-state-down {
        base vn-admin-state-type;
        description "VN administratively state down";
    }
    identity vn-compute-state-type {
        description
            "Base identity for compute states";
    }
    identity vn-compute-state-computing {
        base vn-compute-state-type;
        description
            "State path compute in progress";
    }
    identity vn-compute-state-computation-ok {
        base vn-compute-state-type;
        description
            "State path compute successful";
    }
    identity vn-compute-state-computatione-failed {
        base vn-compute-state-type;
        description
            "State path compute failed";
    }
}
/*
 * Groupings
 */

typedef vn-disjointness {
    type bits {
        bit node {
            position 0;
            description "node disjoint";
        }
        bit link {
            position 1;
            description "link disjoint";
        }
        bit srlg {
```

```
        position 2;
        description "srlg disjoint";
    }
}
description
    "type of the resource disjointness for
    VN level applied across all VN members
    in a VN";
}

grouping vn-ap {
    description
        "VNAP related information";
    leaf vn-ap-id {
        type uint32;
        description
            "unique identifier for the referred
            VNAP";
    }
    leaf vn {
        type leafref {
            path "/actn/vn/vn-list/vn-id";
        }
        description
            "reference to the VN";
    }
    leaf abstract-node {
        type leafref {
            path "/nw:networks/nw:network/nw:node/"
                + "tet:te-node-id";
        }
        description
            "a reference to the abstract node in TE
            Topology";
    }
    leaf ltp {
        type te-types:te-tp-id;
        description
            "Reference LTP in the TE-topology";
    }
}

grouping access-point {
    description
        "AP related information";
    leaf access-point-id {
```

```
        type uint32;
        description
            "unique identifier for the referred
            access point";
    }
    leaf access-point-name {
        type string;
        description
            "ap name";
    }

    leaf max-bandwidth {
        type te-types:te-bandwidth;
        description
            "max bandwidth of the AP";
    }
    leaf avl-bandwidth {
        type te-types:te-bandwidth;
        description
            "available bandwidth of the AP";
    }
    /*add details and any other properties of AP,
    not associated by a VN
    CE port, PE port etc.
    */
    list vn-ap {
        key vn-ap-id;
        uses vn-ap;
        description
            "list of VNAP in this AP";
    }
} //access-point
grouping vn-member {
    description
        "vn-member is described by this container";
    leaf vn-member-id {
        type uint32;
        description
            "vn-member identifier";
    }
}
container src
{
    description
        "the source of VN Member";
    leaf src {
```

```
        type leafref {
            path "/actn/ap/access-point-list/access-point-id";
        }
        description
            "reference to source AP";
    }
    leaf src-vn-ap-id{
        type leafref {
            path "/actn/ap/access-point-list/vn-ap/vn-ap-id";
        }
        description
            "reference to source VNAP";
    }
    leaf multi-src {
        if-feature multi-src-dest;
        type boolean;
        description
            "Is source part of multi-source, where
            only one of the source is enabled";
    }
}
container dest
{
    description
        "the destination of VN Member";
    leaf dest {
        type leafref {
            path "/actn/ap/access-point-list/access-point-id";
        }
        description
            "reference to destination AP";
    }
    leaf dest-vn-ap-id{
        type leafref {
            path "/actn/ap/access-point-list/vn-ap/vn-ap-id";
        }
        description
            "reference to dest VNAP";
    }
    leaf multi-dest {
        if-feature multi-src-dest;
        type boolean;
        description
            "Is destination part of multi-destination, where
            only one of the destination is enabled";
    }
}
```

```
    }
  }
  leaf connectivity-matrix-id{
    type leafref {
      path "/nw:networks/nw:network/nw:node/tet:te/"
        + "tet:te-node-attributes/"
        + "tet:connectivity-matrices/"
        + "tet:connectivity-matrix/tet:id";
    }
    description
      "reference to connectivity-matrix";
  }
} //vn-member
/*
grouping policy {
  description
    "policy related to vn-member-id";
  leaf local-reroute {
    type boolean;
    description
      "Policy to state if reroute
        can be done locally";
  }
  leaf push-allowed {
    type boolean;
    description
      "Policy to state if changes
        can be pushed to the customer";
  }
  leaf incremental-update {
    type boolean;
    description
      "Policy to allow only the
        changes to be reported";
  }
}
} //policy
*/
grouping vn-policy {
  description
    "policy for VN-level diverisity";
  leaf vn-level-diversity {
    type vn-disjointness;
    description
      "the type of disjointness on the VN level
        (i.e., across all VN members)";
  }
}
```

```
    }
  }
  /*
  grouping metrics-op {
    description
      "metric related information";
    list metric{
      key "metric-type";
      config false;
      description
        "The list of metrics for VN";
      leaf metric-type {
        type identityref {
          base te-types:path-metric-type;
        }
        description
          "The VN metric type.";
      }
      leaf value{
        type uint32;
        description
          "The limit value";
      }
    }
  }
  */
  /*
  grouping metrics {
    description
      "metric related information";
    list metric{
      key "metric-type";
      description
        "The list of metrics for VN";
      uses te:path-metrics-bounds_config;
      container optimize{
        description
          "optimizing constraints";
        leaf enabled{
          type boolean;
          description
            "Metric to optimize";
        }
        leaf value{
          type uint32;
        }
      }
    }
  }
  */
```

```

        description
            "The computed value";
    }
}
}
}
/*
/*
grouping service-metric {
    description
        "service-metric";
    uses te:path-objective-function_config;
    uses metrics;
    uses te-types:common-constraints_config;
    uses te:protection-restoration-params_config;
    uses policy;
} //service-metric
/*
/*
* Configuration data nodes
*/
container actn {
    description
        "actn is described by this container";
    container ap {
        description
            "AP configurations";
        list access-point-list {
            key "access-point-id";
            description
                "access-point identifier";
            uses access-point {
                description
                    "access-point information";
            }
        }
    }
}
container vn {
    description
        "VN configurations";
    list vn-list {
        key "vn-id";
        description
            "a virtual network is identified by a vn-id";
        leaf vn-id {

```

```
        type uint32;
        description
            "a unique vn identifier";
    }
    leaf vn-name {
        type string;
        description "vn name";
    }
    leaf vn-topology-id{
        type te-types:te-topology-id;
        description
            "An optional identifier to the TE Topology
            Model where the abstract nodes and links
            of the Topology can be found for Type 2
            VNS";
    }
    leaf abstract-node {
        type leafref {
            path "/nw:networks/nw:network/nw:node/"
                + "tet:te-node-id";
        }
        description
            "a reference to the abstract node in TE
            Topology";
    }
    list vn-member-list{
        key "vn-member-id";
        description
            "List of VN-members in a VN";
        uses vn-member;
        /*uses metrics-op;*/
        leaf oper-status {
            type identityref {
                base vn-state-type;
            }
            config false;
            description
                "VN-member operational state.";
        }
    }
}
leaf if-selected{
    if-feature multi-src-dest;
    type boolean;
    default false;
```



```
        config false;
        description
            "Is the vn-member is selected among the
            multi-src/dest options";
    }
/*
container multi-src-dest{
    if-feature multi-src-dest;
    config false;
    description
        "The selected VN Member when multi-src
        and/or mult-destination is enabled.";
    leaf selected-vn-member{
        type leafref {
            path "/actn/vn/vn-list/vn-member-list"
                + "/vn-member-id";
        }
        description
            "The selected VN Member along the set
            of source and destination configured
            with multi-source and/or multi-destination";
    }
}
*/
/*uses service-metric;*/
leaf admin-status {
    type identityref {
        base vn-admin-state-type;
    }
    default vn-admin-state-up;
    description "VN administrative state.";
}
leaf oper-status {
    type identityref {
        base vn-state-type;
    }
    config false;
    description "VN operational state.";
}
    uses vn-policy;
} //vn-list
} //vn
} //actn
/*
* Notifications - TBD
```

```
*/
/*
* RPC
*/
rpc vn-compute{
  description
    "The VN computation without actual
    instantiation";
  input {
    leaf abstract-node {
      type leafref {
        path "/nw:networks/nw:network/nw:node/"
          + "tet:te-node-id";
      }
      description
        "a reference to the abstract node in TE
        Topology";
    }
    list vn-member-list{
      key "vn-member-id";
      description
        "List of VN-members in a VN";
      uses vn-member;
    }
    uses vn-policy;
    /*uses service-metric;*/
  }
  output {
    list vn-member-list{
      key "vn-member-id";
      description
        "List of VN-members in a VN";
      uses vn-member;
      leaf if-selected{
        if-feature multi-src-dest;
        type boolean;
        default false;
        description
          "Is the vn-member is selected among
          the multi-src/dest options";
      }
      /*uses metrics-op;*/
      leaf compute-status {
        type identityref {
          base vn-compute-state-type;
        }
      }
    }
  }
}
```

```

    }
    description
      "VN-member compute state.";
  }
}
/*
container multi-src-dest{
  if-feature multi-src-dest;
  description
    "The selected VN Member when multi-src
    and/or mult-destination is enabled.";
  leaf selected-vn-member-id{
    type uint32;
    description
      "The selected VN Member-id from the
      input";
  }
}*/
}
}
}

```

<CODE ENDS>

7. JSON Example

This section provides json implementation examples as to how ACTN VN YANG model and TE topology model are used together to instantiate virtual networks.

The example in this section includes following VN

- o VN1 (Type 1): Which maps to the single node topology abstract1 (node D1) and consist of VN Members 104 (L1 to L4), 107 (L1 to L7), 204 (L2 to L4), 308 (L3 to L8) and 108 (L1 to L8). We also show how disjointness (node, link, srlg) is supported in the example on the global level (i.e., connectivity matrices level).

- o VN2 (Type 2): Which maps to the single node topology abstract2 (node D2), this topology has an underlay topology (absolute) (see figure in section 3.2). This VN has a single VN member 105 (L1 to L5) and an underlay path (S4 and S7) has been set in the connectivity matrix of abstract2 topology;
- o VN3 (Type 1): This VN has a multi-source, multi-destination feature enable for VN Member 104 (L1 to L4)/107 (L1 to L7) [multi-src] and VN Member 204 (L2 to L4)/304 (L3 to L4) [multi-dest] usecase. The selected VN-member is known via the field "if-selected" and the corresponding connectivity-matrix-id.

Note that the ACTN VN YANG model also include the AP and VNAP which shows various VN using the same AP.

7.1. ACTN VN JSON

```
{
  "actn": {
    "ap": {
      "access-point-list": [
        {
          "access-point-id": 101,
          "access-point-name": "101",
          "vn-ap": [
            {
              "vn-ap-id": 10101,
              "vn": 1,
              "abstract-node": "D1",
              "ltp": "1-0-1"
            },
            {
              "vn-ap-id": 10102,
              "vn": 2,
              "abstract-node": "D2",
              "ltp": "1-0-1"
            },
            {
              "vn-ap-id": 10103,
              "vn": 3,
              "abstract-node": "D3",
              "ltp": "1-0-1"
            }
          ]
        },
        {
          "access-point-id": 202,
          "access-point-name": "202",
          "vn-ap": [

```

```

        {
            "vn-ap-id": 20201,
            "vn": 1,
            "abstract-node": "D1",
            "ltp": "2-0-2"
        }
    ],
},
{
    "access-point-id": 303,
    "access-point-name": "303",
    "vn-ap": [
        {
            "vn-ap-id": 30301,
            "vn": 1,
            "abstract-node": "D1",
            "ltp": "3-0-3"
        },
        {
            "vn-ap-id": 30303,
            "vn": 3,
            "abstract-node": "D3",
            "ltp": "3-0-3"
        }
    ]
},
{
    "access-point-id": 440,
    "access-point-name": "440",
    "vn-ap": [
        {
            "vn-ap-id": 44001,
            "vn": 1,
            "abstract-node": "D1",
            "ltp": "4-4-0"
        }
    ]
},
{
    "access-point-id": 550,
    "access-point-name": "550",
    "vn-ap": [
        {
            "vn-ap-id": 55002,
            "vn": 2,
            "abstract-node": "D2",
            "ltp": "5-5-0"
        }
    ]
}
]
```

```

    },
    {
      "access-point-id": 770,
      "access-point-name": "770",
      "vn-ap": [
        {
          "vn-ap-id": 77001,
          "vn": 1,
          "abstract-node": "D1",
          "ltp": "7-7-0"
        },
        {
          "vn-ap-id": 77003,
          "vn": 3,
          "abstract-node": "D3",
          "ltp": "7-7-0"
        }
      ]
    },
    {
      "access-point-id": 880,
      "access-point-name": "880",
      "vn-ap": [
        {
          "vn-ap-id": 88001,
          "vn": 1,
          "abstract-node": "D1",
          "ltp": "8-8-0"
        },
        {
          "vn-ap-id": 88003,
          "vn": 3,
          "abstract-node": "D3",
          "ltp": "8-8-0"
        }
      ]
    }
  ],
  "vn": {
    "vn-list": [
      {
        "vn-id": 1,
        "vn-name": "vn1",
        "vn-topology-id": "te-topology:abstract1",
        "abstract-node": "D1",
        "vn-member-list": [
          {
            "vn-member-id": 104,

```

```
"src": {
  "src": 101,
  "src-vn-ap-id": 10101,
},
"dest": {
  "dest": 440,
  "dest-vn-ap-id": 44001,
},
"connectivity-matrix-id": 104
},
{
  "vn-member-id": 107,
  "src": {
    "src": 101,
    "src-vn-ap-id": 10101,
  },
  "dest": {
    "dest": 770,
    "dest-vn-ap-id": 77001,
  },
  "connectivity-matrix-id": 107
},
{
  "vn-member-id": 204,
  "src": {
    "src": 202,
    "dest-vn-ap-id": 20401,
  },
  "dest": {
    "dest": 440,
    "dest-vn-ap-id": 44001,
  },
  "connectivity-matrix-id": 204
},
{
  "vn-member-id": 308,
  "src": {
    "src": 303,
    "src-vn-ap-id": 30301,
  },
  "dest": {
    "dest": 880,
    "src-vn-ap-id": 88001,
  },
  "connectivity-matrix-id": 308
},
{
  "vn-member-id": 108,
  "src": {
```

```

        "src": 101,
        "src-vn-ap-id": 10101,
      },
      "dest": {
        "dest": 880,
        "dest-vn-ap-id": 88001,
      },
      "connectivity-matrix-id": 108
    }
  ]
},
{
  "vn-id": 2,
  "vn-name": "vn2",
  "vn-topology-id": "te-topology:abstract2",
  "abstract-node": "D2",
  "vn-member-list": [
    {
      "vn-member-id": 105,
      "src": {
        "src": 101,
        "src-vn-ap-id": 10102,
      },
      "dest": {
        "dest": 550,
        "dest-vn-ap-id": 55002,
      },
      "connectivity-matrix-id": 105
    }
  ]
},
{
  "vn-id": 3,
  "vn-name": "vn3",
  "vn-topology-id": "te-topology:abstract3",
  "abstract-node": "D3",
  "vn-member-list": [
    {
      "vn-member-id": 104,
      "src": {
        "src": 101,
      },
      "dest": {
        "dest": 440,
        "multi-dest": true
      }
    }
  ],
  {
    "vn-member-id": 107,

```



```

      "src": {
        "src": 101,
        "src-vn-ap-id": 10103,
      },
      "dest": {
        "dest": 770,
        "dest-vn-ap-id": 77003,
        "multi-dest": true
      },
      "connectivity-matrix-id": 107,
      "if-selected": true,
    },
    {
      "vn-member-id": 204,
      "src": {
        "src": 202,
        "multi-src": true,
      },
      "dest": {
        "dest": 440,
      },
    },
    {
      "vn-member-id": 304,
      "src": {
        "src": 303,
        "src-vn-ap-id": 30303,
        "multi-src": true,
      },
      "dest": {
        "dest": 440,
        "src-vn-ap-id": 44003,
      },
      "connectivity-matrix-id": 304,
      "if-selected": true,
    },
  ],
},
]
}
}
}

```

7.2. TE-topology JSON

```

{
  "networks": {

```

```
"network": [
  {
    "network-types": {
      "te-topology": {}
    },
    "network-id": "abstract1",
    "provider-id": 201,
    "client-id": 600,
    "te-topology-id": "te-topology:abstract1",
    "node": [
      {
        "node-id": "D1",
        "te-node-id": "2.0.1.1",
        "te": {
          "te-node-attributes": {
            "domain-id": 1,
            "is-abstract": [null],
            "connectivity-matrices": {
              "is-allowed": true,
              "path-constraints": {
                "bandwidth-generic": {
                  "te-bandwidth": {
                    "generic": [
                      {
                        "generic": "0x1p10",
                      }
                    ]
                  }
                }
              }
            }
          }
        },
        "disjointness": "node link srlg",
      },
    ],
    "connectivity-matrix": [
      {
        "id": 104,
        "from": "1-0-1",
        "to": "4-4-0"
      },
      {
        "id": 107,
        "from": "1-0-1",
        "to": "7-7-0"
      },
      {
        "id": 204,
        "from": "2-0-2",
        "to": "4-4-0"
      },
    ],
  }
]
```

```

        "id": 308,
        "from": "3-0-3",
        "to": "8-8-0"
      },
      {
        "id": 108,
        "from": "1-0-1",
        "to": "8-8-0"
      }
    ],
  }
},
"termination-point": [
  {
    "tp-id": "1-0-1",
    "te-tp-id": 10001,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "1-1-0",
    "te-tp-id": 10100,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "2-0-2",
    "te-tp-id": 20002,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
]
}

```

```
    },
    {
      "tp-id": "2-2-0",
      "te-tp-id": 20200,
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    }
  ],
  {
    "tp-id": "3-0-3",
    "te-tp-id": 30003,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
],
{
  "tp-id": "3-3-0",
  "te-tp-id": 30300,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "4-0-4",
  "te-tp-id": 40004,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
]
```



```
"tp-id": "6-6-0",
"te-tp-id": 60600,
"te": {
  "interface-switching-capability": [
    {
      "switching-capability": "switching-otn",
      "encoding": "lsp-encoding-oduk"
    }
  ]
},
{
  "tp-id": "7-0-7",
  "te-tp-id": 70007,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "7-7-0",
  "te-tp-id": 70700,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "8-0-8",
  "te-tp-id": 80008,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "8-8-0",
```

```

        "te-tp-id": 80800,
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-otn",
                    "encoding": "lsp-encoding-oduk"
                }
            ]
        }
    ]
},
{
    "network-types": {
        "te-topology": {}
    },
    "network-id": "abstract2",
    "provider-id": 201,
    "client-id": 600,
    "te-topology-id": "te-topology:abstract2",
    "node": [
        {
            "node-id": "D2",
            "te-node-id": "2.0.1.2",
            "te": {
                "te-node-attributes": {
                    "domain-id": 1,
                    "is-abstract": [null],
                    "connectivity-matrices": {
                        "is-allowed": true,
                        "underlay": {
                            "enabled": true
                        }
                    },
                    "path-constraints": {
                        "bandwidth-generic": {
                            "te-bandwidth": {
                                "generic": [
                                    {
                                        "generic": "0x1p10"
                                    }
                                ]
                            }
                        }
                    }
                },
                "optimizations": {
                    "objective-function": {

```

```

    "objective-function-type": "of-maximize-residual-
bandwidth"
    }
  },
  "connectivity-matrix": [
    {
      "id": 105,
      "from": "1-0-1",
      "to": "5-5-0",
      "underlay": {
        "enabled": true,
        "primary-path": {
          "network-ref": "absolute",
          "path-element": [
            {
              "path-element-id": 1,
              "index": 1,
              "numbered-hop": {
                "address": "4.4.4.4",
                "hop-type": "STRICT"
              }
            },
            {
              "path-element-id": 2,
              "index": 2,
              "numbered-hop": {
                "address": "7.7.7.7",
                "hop-type": "STRICT"
              }
            }
          ]
        }
      }
    }
  ]
},
"termination-point": [
  {
    "tp-id": "1-0-1",
    "te-tp-id": 10001,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
]

```



```
    }  
  },  
  {  
    "tp-id": "1-1-0",  
    "te-tp-id": 10100,  
    "te": {  
      "interface-switching-capability": [  
        {  
          "switching-capability": "switching-otn",  
          "encoding": "lsp-encoding-oduk"  
        }  
      ]  
    }  
  },  
  {  
    "tp-id": "2-0-2",  
    "te-tp-id": 20002,  
    "te": {  
      "interface-switching-capability": [  
        {  
          "switching-capability": "switching-otn",  
          "encoding": "lsp-encoding-oduk"  
        }  
      ]  
    }  
  },  
  {  
    "tp-id": "2-2-0",  
    "te-tp-id": 20200,  
    "te": {  
      "interface-switching-capability": [  
        {  
          "switching-capability": "switching-otn",  
          "encoding": "lsp-encoding-oduk"  
        }  
      ]  
    }  
  },  
  {  
    "tp-id": "3-0-3",  
    "te-tp-id": 30003,  
    "te": {  
      "interface-switching-capability": [  
        {  
          "switching-capability": "switching-otn",  
          "encoding": "lsp-encoding-oduk"  
        }  
      ]  
    }  
  }  
}
```

```
    },
    {
      "tp-id": "3-3-0",
      "te-tp-id": 30300,
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    }
  ],
  {
    "tp-id": "4-0-4",
    "te-tp-id": 40004,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
],
{
  "tp-id": "4-4-0",
  "te-tp-id": 40400,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "5-0-5",
  "te-tp-id": 50005,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
}
```

```
{
  "tp-id": "5-5-0",
  "te-tp-id": 50500,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "6-0-6",
  "te-tp-id": 60006,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "6-6-0",
  "te-tp-id": 60600,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "7-0-7",
  "te-tp-id": 70007,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "7-7-0",
  "te-tp-id": 70700,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
}
```

```

        "tp-id": "7-7-0",
        "te-tp-id": 70700,
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-otn",
                    "encoding": "lsp-encoding-oduk"
                }
            ]
        }
    },
    {
        "tp-id": "8-0-8",
        "te-tp-id": 80008,
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-otn",
                    "encoding": "lsp-encoding-oduk"
                }
            ]
        }
    },
    {
        "tp-id": "8-8-0",
        "te-tp-id": 80800,
        "te": {
            "interface-switching-capability": [
                {
                    "switching-capability": "switching-otn",
                    "encoding": "lsp-encoding-oduk"
                }
            ]
        }
    }
]
},
{
    "network-types": {
        "te-topology": {}
    },
    "network-id": "abstract3",
    "provider-id": 201,
    "client-id": 600,
    "te-topology-id": "te-topology:abstract3",
    "node": [
        {

```

```

"node-id": "D3",
"te-node-id": "3.0.1.1",
"te": {
  "te-node-attributes": {
    "domain-id" : 3,
    "is-abstract": [null],
    "connectivity-matrices": {
      "is-allowed": true,
      "path-constraints": {
        "bandwidth-generic": {
          "te-bandwidth": {
            "generic": [
              {
                "generic": "0x1p10",
              }
            ]
          }
        }
      },
      "connectivity-matrix": [
        {
          "id": 107,
          "from": "1-0-1",
          "to": "7-7-0"
        },
        {
          "id": 308,
          "from": "3-0-3",
          "to": "8-8-0"
        }
      ],
    }
  },
  "termination-point": [
    {
      "tp-id": "1-0-1",
      "te-tp-id": 10001,
      "te": {
        "interface-switching-capability": [
          {
            "switching-capability": "switching-otn",
            "encoding": "lsp-encoding-oduk"
          }
        ]
      }
    },
    {
      "tp-id": "1-1-0",

```

```
"te-tp-id": 10100,
"te": {
  "interface-switching-capability": [
    {
      "switching-capability": "switching-otn",
      "encoding": "lsp-encoding-oduk"
    }
  ]
},
{
  "tp-id": "2-0-2",
  "te-tp-id": 20002,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "2-2-0",
  "te-tp-id": 20200,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "3-0-3",
  "te-tp-id": 30003,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "3-3-0",
  "te-tp-id": 30300,
```

```
"te": {
  "interface-switching-capability": [
    {
      "switching-capability": "switching-otn",
      "encoding": "lsp-encoding-oduk"
    }
  ]
},
{
  "tp-id": "4-0-4",
  "te-tp-id": 40004,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "4-4-0",
  "te-tp-id": 40400,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "5-0-5",
  "te-tp-id": 50005,
  "te": {
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  }
},
{
  "tp-id": "5-5-0",
  "te-tp-id": 50500,
  "te": {
```

```
    "interface-switching-capability": [
      {
        "switching-capability": "switching-otn",
        "encoding": "lsp-encoding-oduk"
      }
    ]
  },
  {
    "tp-id": "6-0-6",
    "te-tp-id": 60006,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "6-6-0",
    "te-tp-id": 60600,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "7-0-7",
    "te-tp-id": 70007,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "7-7-0",
    "te-tp-id": 70700,
    "te": {
      "interface-switching-capability": [
```



```

        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "8-0-8",
    "te-tp-id": 80008,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  },
  {
    "tp-id": "8-8-0",
    "te-tp-id": 80800,
    "te": {
      "interface-switching-capability": [
        {
          "switching-capability": "switching-otn",
          "encoding": "lsp-encoding-oduk"
        }
      ]
    }
  }
]
}

```

8. Security Considerations

TDB

9. IANA Considerations

TDB

10. Acknowledgments

The authors would like to thank Xufeng Liu for his helpful comments and valuable suggestions.

11. References

11.1. Normative References

- [TE-TOPO] X. Liu, et al., "YANG Data Model for TE Topologies", work in progress: draft-ietf-teas-yang-te-topo.
- [TE-tunnel] T. Saad, et al., "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", work in progress: draft-ietf-teas-yang-te.

11.2. Informative References

- [RFC7926] A. Farrel (Ed.), "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", RFC 7926, July 2016.
- [ACTN-REQ] Lee, et al., "Requirements for Abstraction and Control of TE Networks", draft-ietf-teas-actn-requirements, work in progress.
- [ACTN-FWK] D. Ceccarelli, Y. Lee [Editors], "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ceccarelli-teas-actn-framework, work in progress.
- [TE-MAP] Y. Lee, D. Dhody, and D. Ceccarelli, "Traffic Engineering and Service Mapping Yang Model", draft-lee-teas-te-service-mapping-yang, work in progress.
- [SERVICE-YANG] Q. Wu, W. Liu and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained, work in progress.
- [ACTN-PM] Y. Lee, et al., "YANG models for ACTN TE Performance Monitoring Telemetry and Network Autonomics", draft-lee-teas-actn-pm-telemetry-autonomics, work in progress.
- [OIF-VTNS] Virtual Transport Network Services 1.0 Specification, IA OIF-VTNS-1.0, April 2017.

12. Contributors

Contributor's Addresses

Haomian Zheng
Huawei Technologies
Email: zhenghaomian@huawei.com

Xian Zhang
Huawei Technologies
Email: zhang.xian@huawei.com

Sergio Belotti
Nokia
Email: sergio.belotti@nokia.com

Qin Wu
Huawei Technologies
Email: bill.wu@huawei.com

Takuya Miyasaka
KDDI
Email: ta-miyasaka@kddi.com

Peter Park
KT
Email: peter.park@kt.com

Authors' Addresses

Young Lee (ed.)
Huawei Technologies
Email: leeyoung@huawei.com

Dhruv Dhody
Huawei Technologies
Email: dhruv.ietf@gmail.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan, 48
Stockholm, Sweden
Email: daniele.ceccarelli@ericsson.com

Igor Bryskin
Huawei
Email: Igor.Bryskin@huawei.com

Bin Yeong Yoon
ETRI
Email: byyun@etri.re.kr

TEAS WG
Internet Draft
Intended status: standard track
Expires: August 21, 2018

Young Lee
Dhruv Dhody
Huawei

Daniele Ceccarelli
Ericsson

Jeff Tantsura
Huawei

Giuseppe Fioccola
Telecom Italia

February 25, 2018

Traffic Engineering and Service Mapping Yang Model

draft-lee-teas-te-service-mapping-yang-06

Abstract

This document provides a YANG data model to map service model (e.g., L3SM) and Traffic Engineering model (e.g., TE Tunnel or ACTN VN model). This model is referred to as TE service Mapping Model. This model is applicable to the operation's need for a seamless control and management of their VPN services with TE tunnel support.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 25, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction.....	2
2. L3VPN Architecture in ACTN context.....	5
3. TE-Service Mapping Model.....	7
4. YANG Data Tree.....	7
5. Yang Data Model.....	8
6. Security.....	14
7. IANA Considerations.....	15
8. Acknowledgements.....	15
9. References.....	16
9.1. Informative References.....	16
10. Contributors.....	17
Authors' Addresses.....	17

1. Introduction

Data models are a representation of objects that can be configured or monitored within a system. Within the IETF, YANG [RFC6020] is the language of choice for documenting data models, and YANG models have been produced to allow configuration or modeling of a variety of

network devices, protocol instances, and network services. YANG data models have been classified in [Netmod-Yang-Model-Classification] and [Service-YANG].

[RFC4110] provides a framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs). [L3SM-YANG] provides a L3VPN service delivery YANG model for PE-based VPNs. The scope of this draft is limited to a set of domain under the same network operators to deliver services requiring TE tunnels.

[ACTN-VN-YANG] describes how customers or end to end orchestrators can request and/or instantiate a generic virtual network service. [ACTN-Applicability] describes a connection between IETF YANG model classifications to ACTN interfaces. In particular, it describes the customer service model can be mapped into the CMI (CNC-MDSC Interface) of the ACTN architecture.

The YANG model on the ACTN CMI is known as customer service model in [Service-YANG]. The YANG model developed in this document describes how operator's end to end orchestrator interacts with the MDSC so that the MDSC then can coordinate the control and management of L3VPN MPLS TE tunnels that traverse both IP/MPLS and Transport networks. In addition, the YANG model described in this document also supports the mapping with TE tunnels for other VPN models such as L1VPN and L2VPN, etc.

While IP/MPLS PNC is responsible for provisioning the VPN service on the PE nodes, the MDSC can coordinate how to map the VPN services with TE tunnels. This is consistent with the two of the core functions of the MDSC specified in [ACTN-Frame]:

- . Customer mapping/translation function: This function is to map customer requests/commands into network provisioning requests that can be sent to the Physical Network Controller (PNC) according to business policies provisioned statically or dynamically. Specifically, it provides mapping and translation of a customer's service request into a set of parameters that are specific to a network type and technology such that network configuration process is made possible.
- . Virtual service coordination function: This function translates customer service-related information into virtual network service operations in order to seamlessly operate virtual networks while meeting a customer's service requirements. In the context of ACTN, service/virtual service coordination includes a number of service orchestration functions such as multi-destination load balancing, guarantees of service

quality, bandwidth and throughput. It also includes notifications for service fault and performance degradation and so forth.

In some cases, under the confines of service policy, dynamic TE tunnel creation may need to be supported for the VPN service. This may occur when there are no suitable existing TE tunnels that can support VPN service requirements. Or the operator would like to dynamically create and bind tunnels to the VPN, which could not be shared for network slicing.

To summarize there are three mode of operations, but not limited to:

- o New VN/Tunnel Binding - Customer could request an L3VPN service [L3SM-YANG] with a new VN/Tunnel not shared with other existing services. This is to meet VPN isolation requirement. Further the mapping yang model described in Section 5 of this document is used to set this mapping between the L3VPN service and the ACTN VN. Note that this could be done dynamically. The VN (and TE tunnels) could be bound to the L3VPN and not used for any other VPN.
- o VN/Tunnel Selection - Customer could request an L3VPN service [L3SM-Yang], and with this model as input, the PNC configures the different network elements to deliver the service. Each network element would select a tunnel based on the configuration. With this mode, new tunnels (or VN) are not created for each VPN. Thus, the tunnels can be shared across multiple VPN. Further the mapping yang model described in Section 5 of this document is used to get the mapping between the L3VPN and the tunnels in use. No modification is allowed when an existing tunnel is selected.
- o VN/Tunnel Modify - This mode allows the modification of the properties of the existing VN/tunnel (e.g., bandwidth) when VN/Tunnel Selection Mode is applied.

Other mode of operations could be easily added to the current model in the future.

[Editor's note - A future version of the document can be updated to add more modes or policy.]

The YANG model described in this document provides an ACTN TE-service mapping model that enables a seamless service mapping across L1/2/3 VPN, ACTN VN and TE-tunnel models at the controllers.

2. L3VPN Architecture in ACTN context

Figure 1 shows the architectural context of this document.

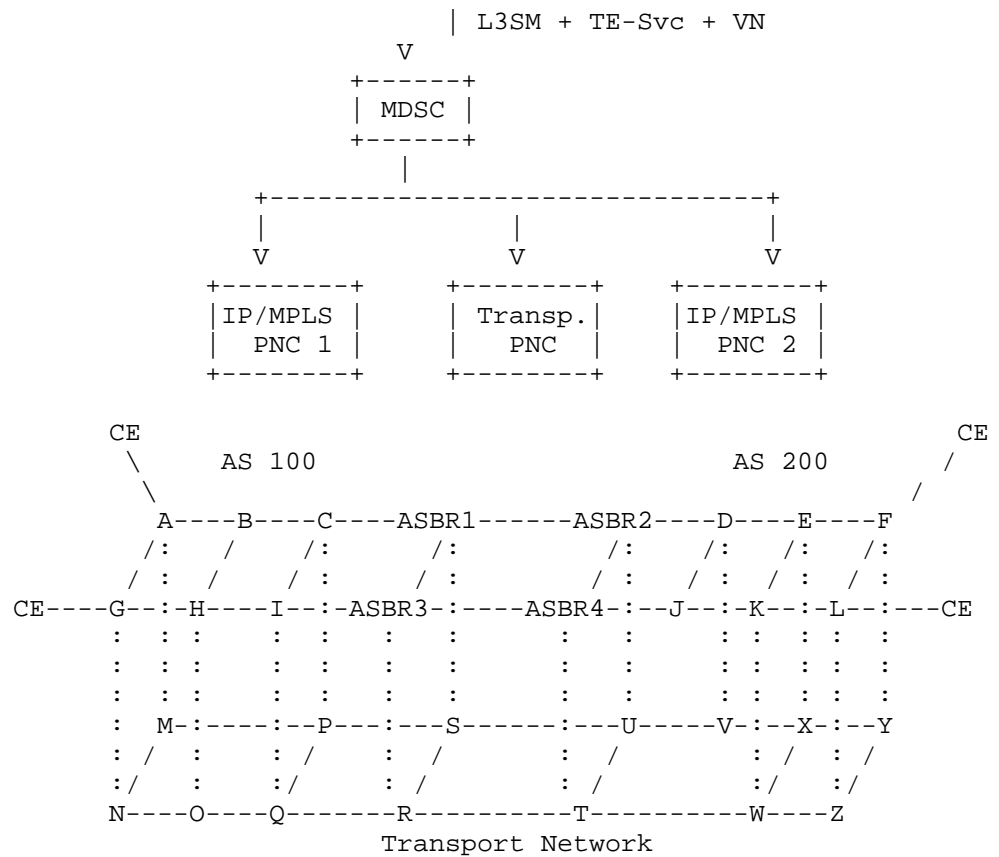


Figure 1: L3VPN Architecture from the IP+Optical Network Perspective

There are three main entities in the architecture.

- . MDSC: This entity is responsible for coordinating a L3VPN service request (expressed in L3SM) with the IP PNC and the Transport PNC.

One of the key responsibilities of the MDSC for TE services is to coordinate with both the IP PNC and the Transport PNC for the mapping of L3VPN Service Model and ACTN VN model. With the VN/TE-tunnel binding case, the MDSC will need to coordinate with the Transport PNC to dynamically create the TE-tunnel(s) in the Transport network as needed. These tunnels are added as links in the IP Layer topology. The MDSC coordinates with IP PNC to create the TE-tunnel(s) in the IP layer, as part of the ACTN VN creation.

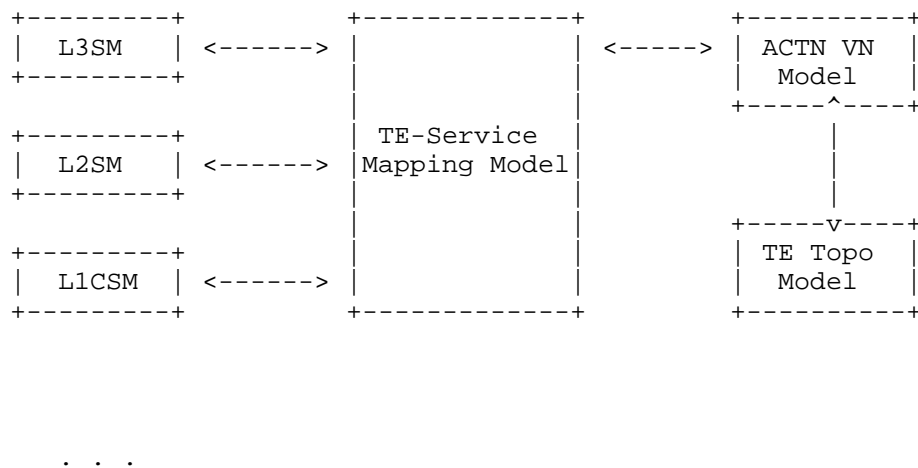
- . IP/MPLS PNC: This entity is responsible for device configuration to create PE-PE L3VPN tunnels for the VPN customer and for the configuration of the L3VPN VRF on the PE nodes. Each network element would select a tunnel based on the configuration.
- . Transport PNC: This entity is responsible for device configuration for TE tunnels in the transport networks.

High-Level Control Flows

1. Customer asks for a L3VPN between CE1 and CE2 with TE constraints using L3SM model. The customer can provide tunnel creation policy where it allows dynamic VN/TE tunnel creation or not. Under this policy, dynamic VN/TE tunnels can be created when there are no proper VN/TE-tunnels that can support L3VPN tunnels or when there is a strict isolation requirement for the VPN service, e.g., no sharing with other tunnels is allowed.
2. The MDSC determines if it needs to create a new VN, and if that is the case, ACTN VN YANG [ACTN-VN-YANG] is used to configure a new VN based on this VPN and map the VPN service to ACTN VN. In case an existing tunnel is to be used, each device will select which tunnel to use and populates this mapping information.
3. The MDSC interacts with both the IP/MPLS PNC and the Transport PNC to create a PE-PE tunnel in the IP network mapped to a TE tunnel in the transport network by providing the inter-layer access points and tunnel requirements. The specific service information are passed to the IP/MPLS PNC for the actual VPN configuration and activation.
 - a. The Transport PNC creates the corresponding TE tunnel matching with the access point and egress point.
 - b. The IP/MPLS PNC maps the VPN ID with the corresponding TE tunnel ID to bind these two IDs.
4. The IP/MPLS PNC creates/updates a VRF instance for this VPN customer. This is not in the scope of this document..

3. TE-Service Mapping Model

The role of TE-service Mapping model is to create a binding relationship across L3SM, L2SM [L2SM-YANG] and L1CSM [L1CSM-YANG] and ACTN VN Model. The ACTN VN YANG model is a generic virtual network service model that allows customers (internal or external) to create a VN that meets the customer's service objective with various constraints via TE-topology model [TE-topo]. The TE-service mapping model is needed to bind LxVPN specific service model with TE-specific parameters. This binding will facilitate a seamless service operation with underlay-TE network visibility. The TE-service model developed in this document can also be extended to support other services beyond L3SM, L2SM and L1CSM.



4. YANG Data Tree

```

module: ietf-te-service-mapping
  +--rw te-service-mapping
    +--rw service-mapping
      +--rw mapping-list* [map-id]
        +--rw map-id          uint32
        +--rw map-type?       map-type
        +--rw (service)?
          +--:(l3vpn)
  
```

```

| | | +--rw l3vpn-ref? -> /l3:l3vpn-svc/vpn-services/vpn-
service/vpn-id | | +---:(l2vpn)
| | | +--rw l2vpn-ref? -> /l2:l2vpn-svc/vpn-services/vpn-
service/vpn-id | | +---:(l1vpn)
| | | +--rw l1vpn-ref? -> /l1:l1cs/service/service-
list/subscriber-l1vc-id
| | +--rw actn-vn-ref? -> /vn:actn/vn/vn-list/vn-id
+--rw site-mapping
+--rw mapping-list* [map-id]
+--rw map-id uint32
+--rw (service)?
| +---:(l3vpn)
| | +--rw l3vpn-ref? -> /l3:l3vpn-svc/sites/site/site-id
| +---:(l2vpn)
| | +--rw l2vpn-ref? -> /l2:l2vpn-svc/sites/site/site-id
| +---:(l1vpn)
| | +--rw l1vpn-ref? -> /l1:l1cs/access/uni-list/UNI-ID
+--rw actn-ap-ref? -> /vn:actn/ap/access-point-list/access-
point-id

```

5. Yang Data Model

The YANG code is as follows:

<CODE BEGINS> file "ietf-te-service-mapping@2018-02-24.yang"

```

module ietf-te-service-mapping {

    namespace "urn:ietf:params:xml:ns:yang:ietf-te-service-mapping";

    prefix "tm";

    import ietf-l3vpn-svc {
        prefix "l3";
    }

    import ietf-l2vpn-svc {
        prefix "l2";
    }

    import ietf-l1csm {
        prefix "l1";
    }

```

```
}

import ietf-actn-vn {
    prefix "vn";
}

organization
    "IETF Traffic Engineering Architecture and Signaling (TEAS)
    Working Group";

contact
    "Editor: Young Lee <leeyoung@huawei.com>
    Dhruv Dhody <dhruv.ietf@gmail.com>";

description
    "This module contains a YANG module for the mapping of
    service (e.g. L3VPN) to the TE tunnels or ACTN VN.";

revision 2018-02-24 {
    description
        "initial version.";
    reference
        "TBD";
}

/*
 * Identities
 */
identity service-type {
    description
        "Base identity from which specific service types are
        derived.";
}

identity l3vpn-service {
    base service-type;
    description
        "L3VPN service type.";
}

identity l2vpn-service {
    base service-type;
    description
        "L2VPN service type.";
}
```

```

identity l3vpn-service {
    base service-type;
    description
        "L3VPN connectivity service type.";
}
/*
 * Enum
 */
typedef map-type {
    type enumeration {
        enum "new" {
            description
                "The new VN/tunnels are binded to the service. Customer
could request a VN with a new VN/Tunnel not shared with other existing
services. This is to meet VPN isolation requirement. ";
        }
        enum "select" {
            description
                "The VPN service selects an existing tunnel with no
modification";
        }
        enum "modify" {
            description
                "The VPN service selects an existing tunnel and allows
to modify the properties of the tunnel (e.g., b/w) ";
        }
    }
    description
        "The map-type";
}

/*
 * Groupings
 */
grouping service-ref{
    description
        "The reference to the service.";
    choice service {
        description
            "The service";
        case l3vpn {
            leaf l3vpn-ref {
                type leafref {
                    path "/l3:l3vpn-svc/l3:vpn-services/"
                    + "l3:vpn-service/l3:vpn-id";
                }
            }
        }
    }
}

```



```
        }
        description
            "The reference to L3VPN Service Yang Model";
    }
}
case l2vpn {
    leaf l2vpn-ref {
        type leafref {
            path "/l2:l2vpn-svc/l2:vpn-services/"
                + "l2:vpn-service/l2:vpn-id";
        }
        description
            "The reference to L2VPN Service Yang Model";
    }
}
case l1vpn {
    leaf l1vpn-ref {
        type leafref {
            path "/l1:l1cs/l1:service/"
                + "l1:service-list/l1:subscriber-l1vc-id";
        }
        description
            "The reference to L1VPN Service Yang Model";
    }
}
}
}

grouping site-ref {
    description
        "The reference to the site.";
    choice service {
        description
            "The service choice";
        case l3vpn {
            leaf l3vpn-ref {
                type leafref {
                    path "/l3:l3vpn-svc/l3:sites/l3:site/"
                        + "l3:site-id";
                }
                description
                    "The reference to L3VPN Service Yang Model";
            }
        }
    }
}
```

```

    }
    case l2vpn {
      leaf l2vpn-ref {
        type leafref {
          path "/l2:l2vpn-svc/l2:sites/l2:site/"
            + "l2:site-id";
        }
        description
          "The reference to L2VPN Service Yang Model";
      }
    }
  }
  case l1vpn {
    leaf l1vpn-ref {
      type leafref {
        path "/l1:l1cs/l1:access/l1:uni-list/"
          + "l1:UNI-ID";
      }
      description
        "The reference to L1VPN Connectivity Service Yang
Model";
    }
  }
}

grouping actn-vn-ref {
  description
    "The reference to ACTN VN.";
  leaf actn-vn-ref {
    type leafref {
      path "/vn:actn/vn:vn/vn:vn-list/vn:vn-id";
    }
    description
      "The reference to ACTN VN";
  }
}

grouping actn-ap-ref {
  description
    "The reference to ACTN endpoints (AP).";
  leaf actn-ap-ref {
    type leafref {

```

```
        path "/vn:actn/vn:ap/vn:access-point-list"
            + "/vn:access-point-id";
    }
    description
        "The reference to ACTN AP";
}

grouping service-mapping {
    description
        "Mapping between Services and TE";
    container service-mapping {
        description
            "Mapping between Services and TE";

        list mapping-list {
            key "map-id";
            description
                "Mapping identified via a map-id";
            leaf map-id {
                type uint32;
                description
                    "a unique mapping identifier";
            }
            leaf map-type {
                type map-type;
                description
                    "Tunnel Bind or Tunnel Selection";
            }
            uses service-ref;

            uses actn-vn-ref ;
        }
    }
}

grouping site-mapping {
    description
        "Mapping between VPN access site and ACTN AP";
    container site-mapping {
        description
            "Mapping between VPN access site and ACTN AP";
        list mapping-list {
            key "map-id";
            description
                "Mapping identified via a map-id";
```

```
        leaf map-id {
            type uint32;
            description
                "a unique mapping identifier";
        }
        uses site-ref;

        uses actn-ap-ref ;
    }
}

/*
 * Configuration data nodes
 */
container te-service-mapping {
    description
        "Mapping between Services and TE";

    uses service-mapping;

    uses site-mapping;
}

}
```

<CODE ENDS>

6. Security

The configuration, state, and action data defined in this document are designed to be accessed via a management protocol with a secure transport layer, such as NETCONF [RFC6241]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a preconfigured subset of all available NETCONF protocol operations and content.

A number of configuration data nodes defined in this document are writable/deletable (i.e., "config true") These data nodes may be considered sensitive or vulnerable in some network environments.

7. IANA Considerations

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

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

This document registers the following YANG modules in the YANG Module

Names registry [RFC7950]:

```
-----  
name:          ietf-te-service-mapping  
namespace:     urn:ietf:params:xml:ns:yang:ietf-te-service-mapping  
reference:     RFC XXXX (TDB)  
-----
```

8. Acknowledgements

We thank Diego Caviglia and Igor Bryskin for useful discussions and motivation for this work.

9. References

9.1. Informative References

- [RFC4110] R. Callon and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, July 2005.
- [RFC6020] M. Bjorklund, Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

- [Service-YANG] Q. Wu, W. Liu and A. Farrel, "Service Models Explained", draft-wu-opsawg-service-model-explained, work in progress.
- [Netmod-Yang-Model-Classification] D. Bogdanovic, B. Claise, and C. Moberg, "YANG Module Classification", draft-ietf-netmod-yang-model-classification, work in progress.
- [Netconf] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241.
- [Restconf] A. Bierman, M. Bjorklund, and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf, work in progress.
- [ACTN-Frame] D. Cecarelli and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework, work in progress.
- [TE-Topology] X. Liu, et. al., "YANG Data Model for TE Topologies", draft-ietf-teas-yang-te-topo, work in progress.
- [TE-Tunnel] T. Saad (Editor), "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.
- [ACTN-VN-YANG] Y. Lee (Editor), "A Yang Data Model for ACTN VN Operation", draft-lee-teas-actn-vn-yang, work in progress.
- [L3SM-YANG] S. Litkowski, L. Tomotaki, and K. Ogaki, "YANG Data Model for L3VPN service delivery", draft-ietf-l3sm-l3vpn-service-model, work in progress.
- [L2SM-YANG] B. Wen, et al, "A YANG Data Model for L2VPN Service Delivery", draft-ietf-l2sm-l2vpn-service-model, work in progress.
- [L1CSM-YANG] G. Fioccola, et al, "A Yang Data Model for L1 Connectivity Service Model (L1CSM)", draft-fioccola-ccamp-l1csm-yang, work in progress.

10. Contributors

Authors' Addresses

Young Lee
Huawei Technologies
5340 Legacy Drive
Plano, TX 75023, USA
Phone: (469)277-5838

Email: leeyoung@huawei.com

Dhruv Dhody
Huawei Technologies

Email: dhruv.ietf@gmail.com

Daniele Ceccarelli
Ericsson
Torshamnsgatan, 48
Stockholm, Sweden

Email: daniele.ceccarelli@ericsson.com

Jeff Tantsura
Huawei

Email: jefftant@gmail.com

Giuseppe Fioccola
Telecom Italia
Email: giuseppe.fioccola@telecomitalia.it

TEAS Working Group
Internet-Draft
Intended status: Informational
Expires: September 2, 2018

Z. Li
D. Dhody
H. Chen
Huawei Technologies
March 1, 2018

Hierarchy of IP Controllers (HIC)
draft-li-teas-hierarchy-ip-controllers-00

Abstract

This document describes the interactions between various IP controllers in a hierarchical fashion to provide various IP services. It describes how the Abstraction and Control of Traffic Engineered Networks (ACTN) framework is applied to the Hierarchy of IP controllers (HIC) as well as document the interactions with other protocols like BGP, Path Computation Element Communication Protocol (PCEP) to provide end to end dynamic services spanning multiple domains and controllers (e.g. Layer 3 Virtual Private Network (L3VPN), Seamless MPLS etc).

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 September 2, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Overview	4
2.1. Mapping to ACTN	4
2.2. Interface between Super Controller and Domain Controller in HIC	5
3. Key Concepts	6
3.1. Topology	6
3.2. Path Computation/Path instantiation	7
3.3. BGP considerations	8
4. VPN Service	8
4.1. Seamless MPLS	8
4.2. L3VPN	10
4.3. L2VPN and EVPN service	11
5. Possible Features/Extensions	11
6. Other Considerations	12
6.1. Control Plane	12
6.1.1. PCE / PCEP	12
6.1.2. BGP	13
6.2. Management Plane	13
6.2.1. YANG Models	13
6.2.2. Protocol Considerations	14
7. IANA Considerations	14
8. Security Considerations	14
9. Acknowledgments	15
10. References	15
10.1. Normative References	15
10.2. Informative References	15
Authors' Addresses	19

1. Introduction

Software-Defined Networking (SDN) refers to a separation between the control elements and the forwarding components so that software running in a centralized system called a controller, can act to program the devices in the network to behave in specific ways. A required element in an SDN architecture is a component that plans how the network resources will be used and how the devices will be programmed. It is possible to view this component as performing specific computations to place flows within the network given knowledge of the availability of network resources, how other

forwarding devices are programmed, and the way that other flows are routed. The Application-Based Network Operation (ABNO) [RFC7491] describes how various components and technologies fit together.

A domain [RFC4655] is any collection of network elements within a common sphere of address management or path computation responsibility. Specifically within this document we mean a part of an operator's network that is under common management. Network elements will often be grouped into domains based on technology types, vendor profiles, and geographic proximity and under a domain controller.

Multiple such domains in the network are interconnected, and a path is established through a series of connected domains to form an end-to-end path over which various services are offered. Each domain is under the control of the domain controller (or lower-level controller), and a "super controller" (or high-level controller) takes responsibility for a high-level view of the network before distributing tasks to domain controllers (or lower-level controllers). It is possible for each of the domain to use a different tunneling mechanism (eg RSVP-TE, Segment Routing (SR) etc).

[I-D.ietf-teas-actn-framework] describes the framework for Abstraction and Control of Traffic Engineered Networks (ACTN) as well as a set of management and control functions used to operate multiple TE networks. This documents would apply the ACTN principles to Hierarchy of IP controllers (HIC) and focus on the applicability and interactions with other protocol and technologies (specific to IP packet domains).

Sometimes, service (such as Layer 3 Virtual Private Network (L3VPN), Layer 2 Virtual Private Network (L2VPN), Ethernet VPN (EVPN), Seamless MPLS) require sites attached to different domains (under the control of different domain controller) to be interconnected as part of the VPN service. This require multi-domain coordination between domain controllers to compute and setup E2E path for the VPN service.

This document describes the interactions between various IP controllers in a hierarchical fashion to provide various IP services. It describes how the Abstraction and Control of Traffic Engineered Networks (ACTN) framework is applied to the Hierarchy of IP controllers (HIC) as well as document the interactions with control plane protocols (like BGP, Path Computation Element Communication Protocol (PCEP)) and management plane aspects (Yang models) to provide end to end dynamic services spanning multiple domains and controllers (e.g. L3VPN, Seamless MPLS etc).

2. Overview

Figure 1 show examples of multi-domain IP domains under hierarchy of IP controllers.

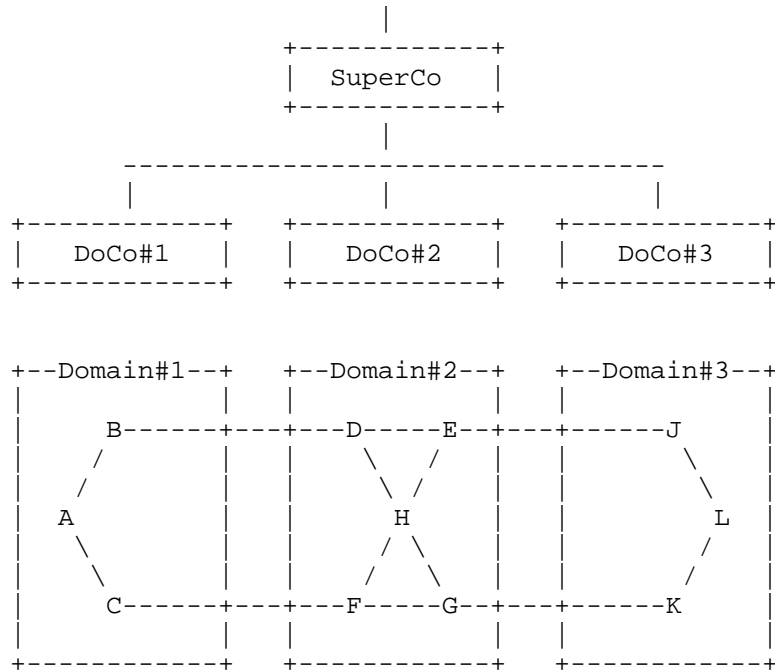


Figure 1: Example: Hierarchy of IP controllers (HIC)

The IP "Super Controller" receives request from the network/service orchestrator to setup dynamic services spanning multiple domains. The IP "Super Controller" breaks down and assigns tasks to the domain controllers, responsible for communicating to network devices in the domain. It further coordinates between the controller to provide a unified view of the multi-domain network.

2.1. Mapping to ACTN

As per [I-D.ietf-teas-actn-framework], ACTN has following main functions -

- o Multi-domain coordination
- o Virtualization/Abstraction

- o Customer mapping/translation
- o Virtual service coordination

These functions are part of Multi Domain Service Coordinator (MDSC) and/or Provisioning Network Controller (PNC). Further these functions are part of the controller / orchestrator.

The HIC is an instantiation of ACTN framework for IP packet network. The IP domain (lower-level) controllers implements the PNC functionalities for configuring, controlling and monitoring the IP domain. The "super controller" (high-level controller) implements the MDSC functionalities for coordination between multiple domains as well as maintaining an abstracted view of multiple domains. It also takes care of the service related functionalities of customer mapping/translation and virtual service coordination.

The ACTN functions are part of the IP controllers and responsible for the TE topology and E2E path computation/setup. There are other functions along with ACTN that are needed to manage multiple IP domain networks.

2.2. Interface between Super Controller and Domain Controller in HIC

The interaction between super controller and domain controller in HIC is a combination of Control Plane and Management Plane interface as shown in Figure 2. BGP [RFC4271] and PCEP [RFC5440] are example of the control plane interface; where as NETCONF [RFC6241] and RESTCONF [RFC8040] are example of management plane interface.

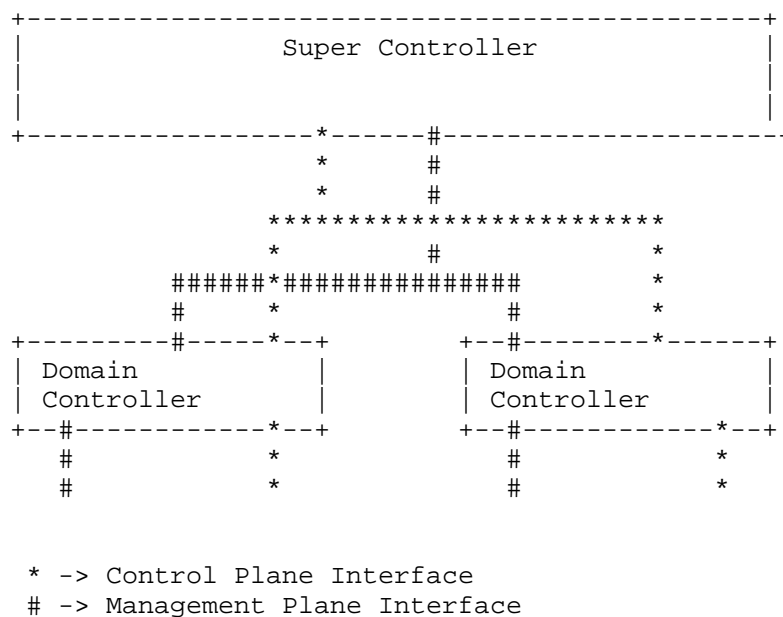


Figure 2: Interface between Super Controller and Domain Controller

Note that ACTN's MDSC-PNC Interface (MPI) could be implemented via management plane interface using Yang models [I-D.ietf-teas-actn-yang] or via PCEP control plane interface [I-D.ietf-pce-applicability-actn].

3. Key Concepts

3.1. Topology

The Domain Controller is expected to be aware of the topology of the network devices in its domain. The domain controller could participate in the IGP ([RFC3630] and [RFC5305]) or use BGP-LS [RFC7752] by which link-state and TE information is collected and shared with domain controller using the BGP routing protocol.

An alternate approach would be to rely on the management plane interface which uses the YANG model for network/TE Topology as per [I-D.ietf-i2rs-yang-network-topo] and [I-D.ietf-teas-yang-te-topo].

The domain controller is expected to share the domain topology to the Super Controller as aligned to ACTN (where PNC abstract the topology towards MDSC). A level of abstraction is usually applied while

presenting the topology to a higher level controller. Topology abstraction is described in [RFC7926] as well as [I-D.ietf-teas-actn-framework]. BGP-LS, PCEP-LS [I-D.dhodylee-pce-pcep-ls] or management plane interface based on the abstracted network/TE Topology could be used to carry the abstract topology to the super-controller. At minimum the border nodes and inter-domain links are exposed to the super-controller.

Further [I-D.ietf-teas-actn-framework] defines three types of topology abstraction - (1) Native/White Topology; (2) Black Topology; and (3) Grey Topology. Based on the local policy, the domain controller would share the domain topology to the Super Controller based on the abstraction type. Note that any of the control plane or management plane mechanism could be used to carry abstracted domain topology. The Super Controller's MDSC function is expected to manage a E2E topology by coordinating the abstracted domain topology received from the domain controllers.

3.2. Path Computation/Path instantiation

The Domain Controller is responsible for computing and setup of path when the source and destination is in the same domain, otherwise the Super Controller coordinates the multi-domain path computation and setup with the help of the domain controller. This is aligned to ACTN.

PCEP [RFC5440] provides mechanisms for Path Computation Elements (PCEs) [RFC4655] to perform path computations in response to Path Computation Clients (PCCs) requests. Since then, the role and function of the PCE has grown to allow delegated control [RFC8231] and PCE-initiated use of network resources [RFC8281].

Further, [RFC6805] and [I-D.ietf-pce-stateful-hpce] describes a hierarchy of PCE with Parent PCE coordinating multi-domain path computation function between Child PCE(s). This fits well with HIC as described in this document.

Note that a management plane interface which uses the YANG model for path computation/setup ([I-D.ietf-teas-yang-path-computation] and [I-D.ietf-teas-yang-te]) could be used in place of PCEP.

In case there is a need to stitch per domain tunnels into an E2E tunnel, mechanism are described in [I-D.lee-pce-lsp-stitching-hpce] and [I-D.dugeon-pce-stateful-interdomain].

3.3. BGP considerations

[RFC4456] describes the concept of route-reflection where a "route reflector" (RR) reflects the routes to avoid full mesh connection between Internal BGP (IBGP) peers. The IP domain controller can play the role of RR in its domain. The super controller can further act as RR to towards the domain controller.

[Editor's Note: To do - BGP Policy, BGP Flowspec. More information will be added in the next version]

[Editor's Note: Need to evaluate a role of BMP]

4. VPN Service

4.1. Seamless MPLS

Seamless MPLS [I-D.ietf-mpls-seamless-mpls] describes an architecture which can be used to extend MPLS networks to integrate access and core/aggregation networks into a single MPLS domain. In the seamless MPLS for mobile backhaul, since there are multiple domains including the core network and multiple mobile backhaul networks, for each domain there is a domain controller. In order to implement the end-to-end network service provision, there should be coordination among multiple domain controllers.

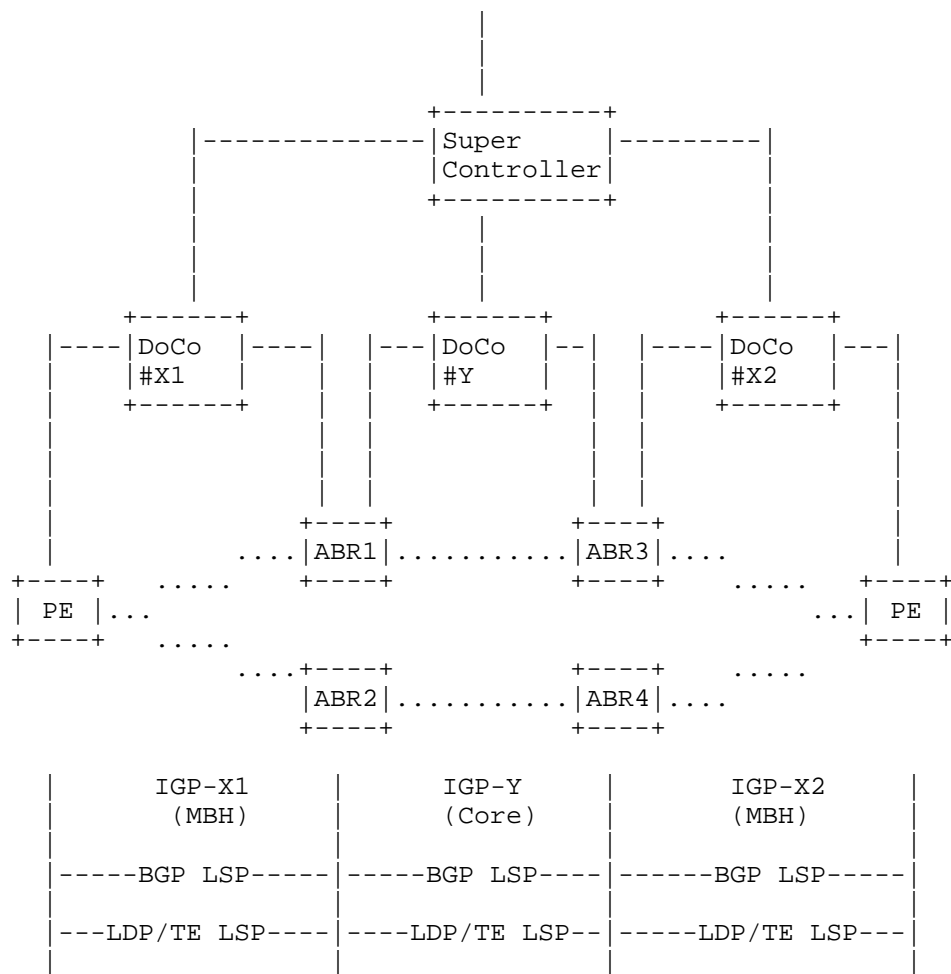


Figure 3: Seamless MPLS

Super Controller is responsible for setting the seamless MPLS service. It should break down the service model to network configuration model [RFC8309] and the domain controller further break it to the device configuration model to the PE/ASBR to make the E2E seamless MPLS service. The selection of appropriate ASBRs and handling of intra-domain tunnels is coordinated by the Super Controller in the similar fashion as shown in Section 4.2.

By enabling BGP sessions between Domain Controller and Super Controller, BGP labeled routes can also be learned at Super

Controller. As Super Controller is aware of the (abstract) topology, it could make intelligent decisions regarding E2E BGP LSP to optimize based on the overall traffic information.

4.2. L3VPN

A Layer 3 IP VPN service is a collection of sites that are authorized to exchange traffic between each other over a shared IP infrastructure. [RFC4110] provides a framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs). [RFC8299] provides a L3VPN service delivery YANG model for PE-based VPNs. The Super controller is expected to implement the L3SM model and translate it to network models towards the domain controller, which in turns translate it to the device model. See [RFC8309] for more details.

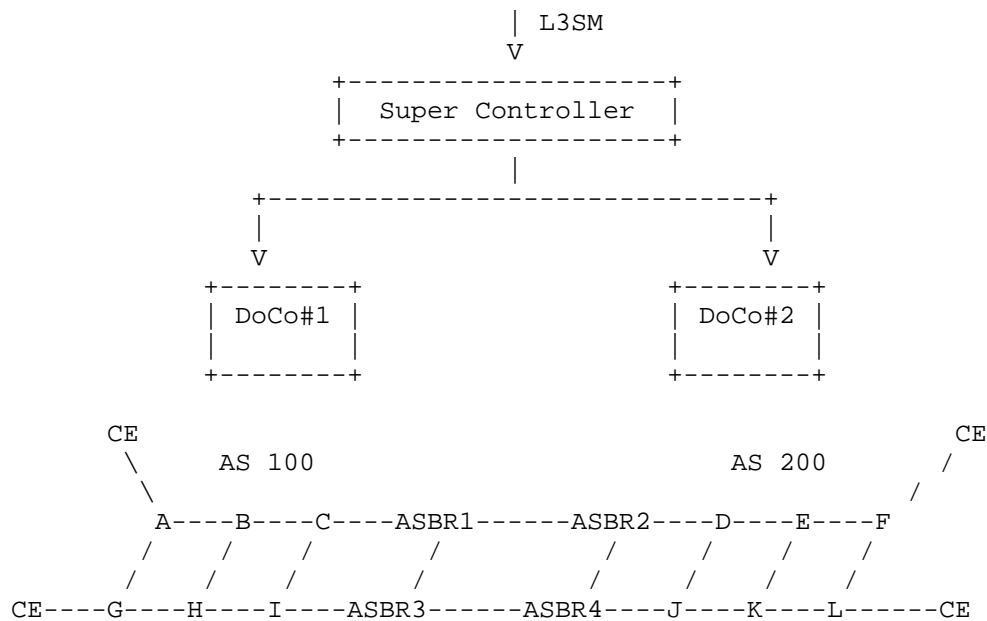


Figure 4: L3VPN

Based on the user data in L3SM model, the network configurations need to be trickle down to the network device to setup the L3VPN.

Based on the QoS or Policy requirement for the L3VPN service, the Super Controller may -

- o Set the tunnel selection policy at the PE/ASBR routers so that they could select the existing tunnels

- o Select an existing tunnels at the controller level and bind it to the VPN service
- o Initiate the process of creating a new tunnel based on the QoS requirement and bind it the VPN service
- o Initiate the process of creating a new tunnel based on the the policy

Refer [I-D.lee-teas-te-service-mapping-yang] for more details from ACTN perspective.

Apart from the Management plane interface based on respective YANG models, the control plane interface PCEP could be used for path computation and setup.

4.3. L2VPN and EVPN service

There are two fundamentally different kinds of Layer 2 VPN service that a service provider could offer to a customer: Virtual Private Wire Service (VPWS) and Virtual Private LAN Service (VPLS) [RFC4664]. A VPWS is a VPN service that supplies an L2 point-to-point service. A VPLS is an L2 service that emulates LAN service across a Wide Area Network (WAN). A BGP MPLS-based Ethernet VPN (EVPN) [RFC7432] addresses some of the limitations when it comes to multihoming and redundancy, multicast optimization, provisioning simplicity, flow-based load balancing, and multipathing etc.

The handling of L2VPN/EVPN service is done in a similar fashion as shown in Section 4.2.

5. Possible Features/Extensions

This sections list some of the possible features or protocol extensions that could be worked on to deploy HIC in a multi-domain packet network.

1. Simplify the initial configurations needed to setup the relationship between the super controller and the domain controllers. Note that this could be done via exchanges during initial session establishment, discovery via other protocols, service discovery (such as DNS) etc.
2. The (higher-level controller, lower-level controller) relationship or the the role of the controller.
3. The learning and handling of various capabilities of the Super Controller and Domain Controller.

4. Handling of multiple instances of controller at each level for high availability.

[Editor's Note - This list is expected to be updated in next version with more details]

6. Other Considerations

6.1. Control Plane

6.1.1. PCE / PCEP

The Path Computation Element communication Protocol (PCEP) [RFC5440] provides mechanisms for Path Computation Elements (PCEs) [RFC4655] to perform path computations in response to Path Computation Clients (PCCs) requests.

The ability to compute shortest constrained TE LSPs in Multiprotocol Label Switching (MPLS) and Generalized MPLS (GMPLS) networks across multiple domains has been identified as a key motivation for PCE development.

A stateful PCE [RFC8231] is capable of considering, for the purposes of path computation, not only the network state in terms of links and nodes (referred to as the Traffic Engineering Database or TED) but also the status of active services (previously computed paths, and currently reserved resources, stored in the Label Switched Paths Database (LSPDB).

[RFC8051] describes general considerations for a stateful PCE deployment and examines its applicability and benefits, as well as its challenges and limitations through a number of use cases.

[RFC8231] describes a set of extensions to PCEP to provide stateful control. A stateful PCE has access to not only the information carried by the network's Interior Gateway Protocol (IGP), but also the set of active paths and their reserved resources for its computations. The additional state allows the PCE to compute constrained paths while considering individual LSPs and their interactions. [RFC8281] describes the setup, maintenance and teardown of PCE-initiated LSPs under the stateful PCE model.

[RFC8231] also describes the active stateful PCE. The active PCE functionality allows a PCE to reroute an existing LSP or make changes to the attributes of an existing LSP, or a PCC to delegate control of specific LSPs to a new PCE.

Computing paths across large multi-domain environments require special computational components and cooperation between entities in different domains capable of complex path computation. The PCE provides an architecture and a set of functional components to address this problem space. A PCE may be used to compute end-to-end paths across multi-domain environments using a per-domain path computation technique [RFC5152]. The Backward recursive PCE based path computation (BRPC) mechanism [RFC5441] defines a PCE-based path computation procedure to compute inter-domain constrained MPLS and GMPLS TE networks. However, both per-domain and BRPC techniques assume that the sequence of domains to be crossed from source to destination is known, either fixed by the network operator or obtained by other means.

[RFC6805] describes a Hierarchical PCE (H-PCE) architecture which can be used for computing end-to-end paths for inter-domain MPLS Traffic Engineering (TE) and GMPLS Label Switched Paths (LSPs) when the domain sequence is not known. Within the Hierarchical PCE (H-PCE) architecture, the Parent PCE (P-PCE) is used to compute a multi-domain path based on the domain connectivity information. A Child PCE (C-PCE) may be responsible for a single domain or multiple domains, it is used to compute the intra-domain path based on its domain topology information.

[I-D.ietf-pce-stateful-hpce] state the considerations for stateful PCE(s) in hierarchical PCE architecture. In particular, the behavior changes and additions to the existing stateful PCE mechanisms (including PCE- initiated LSP setup and active PCE usage) in the context of networks using the H-PCE architecture.

[I-D.ietf-pce-applicability-actn] examines the applicability of PCE/ PCEP to the ACTN framework in detail.

6.1.2. BGP

[Editor's Note - TBD, More details on BGP-LS, BGP-Flowspec, RR handling, BGP Policy etc to be added in the next revision]

6.2. Management Plane

6.2.1. YANG Models

This is an non-exhaustive list of possible yang models developed or in-development that could be used for HIC.

Topology: [I-D.ietf-i2rs-yang-network-topo] defines a generic YANG data model for network topology. [I-D.ietf-teas-yang-te-topo]

defines a YANG data model for representing, retrieving and manipulating Traffic Engineering (TE) Topologies.

Tunnel: [I-D.ietf-teas-yang-te] defines a YANG data model for the configuration and management of Traffic Engineering (TE) interfaces, tunnels and Label Switched Paths (LSPs).

L3VPN: The Layer 3 service model (L3SM) is defined in [RFC8299], which is a YANG data model that can be used for communication between customers and network operators and to deliver a Layer 3 provider-provisioned VPN service. [I-D.ietf-bess-l3vpn-yang] defines a YANG data model that can be used to configure and manage BGP Layer 3 VPNs at the device. Note that a network configuration model at the Domain Controller level needs to be developed.

L2VPN/EVPN: [I-D.ietf-l2sm-l2vpn-service-model] defines a YANG data model that can be used to configure a Layer 2 Provider Provisioned VPN service. This model is intended to be instantiated at management system to deliver the overall service. [I-D.ietf-bess-l2vpn-yang] and [I-D.ietf-bess-evpn-yang] defines a YANG data model to configure and manage L2VPN and EVPN service respectively. Note that a network configuration model at the Domain Controller level needs to be developed.

OAM: TBD

[Editor's Note - the above list should be extended.]

6.2.2. Protocol Considerations

The Network Configuration Protocol (NETCONF) [RFC6241] provides mechanisms to install, manipulate, and delete the configuration of network devices. The RESTCONF [RFC8040] describes an HTTP-based protocol that provides a programmatic interface for accessing data defined in YANG, using the data-store concepts defined in NETCONF.

Some other mechanism like gRPC/gNMI could also be used between controllers using the same YANG data models.

7. IANA Considerations

There are no IANA concerns in this document.

8. Security Considerations

There are no new security concerns in this document.

9. Acknowledgments

10. References

10.1. Normative References

[I-D.ietf-teas-actn-framework]

Ceccarelli, D. and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-framework-11 (work in progress), October 2017.

10.2. Informative References

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.

[RFC4110] Callon, R. and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, DOI 10.17487/RFC4110, July 2005, <<https://www.rfc-editor.org/info/rfc4110>>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/info/rfc4655>>.

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

[RFC5152] Vasseur, JP., Ed., Ayyangar, A., Ed., and R. Zhang, "A Per-Domain Path Computation Method for Establishing Inter-Domain Traffic Engineering (TE) Label Switched Paths (LSPs)", RFC 5152, DOI 10.17487/RFC5152, February 2008, <<https://www.rfc-editor.org/info/rfc5152>>.

- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.
- [RFC5441] Vasseur, JP., Ed., Zhang, R., Bitar, N., and JL. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths", RFC 5441, DOI 10.17487/RFC5441, April 2009, <<https://www.rfc-editor.org/info/rfc5441>>.
- [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>>.
- [RFC6805] King, D., Ed. and A. Farrel, Ed., "The Application of the Path Computation Element Architecture to the Determination of a Sequence of Domains in MPLS and GMPLS", RFC 6805, DOI 10.17487/RFC6805, November 2012, <<https://www.rfc-editor.org/info/rfc6805>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, DOI 10.17487/RFC7491, March 2015, <<https://www.rfc-editor.org/info/rfc7491>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.

- [RFC7926] Farrel, A., Ed., Drake, J., Bitar, N., Swallow, G., Ceccarelli, D., and X. Zhang, "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", BCP 206, RFC 7926, DOI 10.17487/RFC7926, July 2016, <<https://www.rfc-editor.org/info/rfc7926>>.
- [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>>.
- [RFC8051] Zhang, X., Ed. and I. Minei, Ed., "Applicability of a Stateful Path Computation Element (PCE)", RFC 8051, DOI 10.17487/RFC8051, January 2017, <<https://www.rfc-editor.org/info/rfc8051>>.
- [RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <<https://www.rfc-editor.org/info/rfc8231>>.
- [RFC8281] Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for PCE-Initiated LSP Setup in a Stateful PCE Model", RFC 8281, DOI 10.17487/RFC8281, December 2017, <<https://www.rfc-editor.org/info/rfc8281>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.
- [RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <<https://www.rfc-editor.org/info/rfc8309>>.
- [I-D.ietf-teas-actn-yang]
Lee, Y., zhenghaomian@huawei.com, z., Ceccarelli, D., Yoon, B., and S. Belotti, "Applicability of YANG models for Abstraction and Control of Traffic Engineered Networks", draft-ietf-teas-actn-yang-01 (work in progress), February 2018.

- [I-D.ietf-pce-applicability-actn]
Dhody, D., Lee, Y., and D. Ceccarelli, "Applicability of Path Computation Element (PCE) for Abstraction and Control of TE Networks (ACTN)", draft-ietf-pce-applicability-actn-03 (work in progress), March 2018.
- [I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-12 (work in progress), February 2018.
- [I-D.ietf-teas-yang-te-topo]
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-15 (work in progress), February 2018.
- [I-D.ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo-20 (work in progress), December 2017.
- [I-D.ietf-pce-stateful-hpce]
Dhody, D., Lee, Y., Ceccarelli, D., Shin, J., King, D., and O. Dios, "Hierarchical Stateful Path Computation Element (PCE).", draft-ietf-pce-stateful-hpce-02 (work in progress), October 2017.
- [I-D.ietf-teas-yang-path-computation]
Busi, I., Belotti, S., Lopezalvarez, V., Dios, O., ansharma@infinera.com, a., Shi, Y., Vilata, R., Sethuraman, K., Scharf, M., and D. Ceccarelli, "Yang model for requesting Path Computation", draft-ietf-teas-yang-path-computation-00 (work in progress), November 2017.
- [I-D.ietf-mpls-seamless-mpls]
Leymann, N., Decraene, B., Filsfils, C., Konstantynowicz, M., and D. Steinberg, "Seamless MPLS Architecture", draft-ietf-mpls-seamless-mpls-07 (work in progress), June 2014.
- [I-D.ietf-bess-evpn-yang]
Brissette, P., Shah, H., Hussain, I., Tiruveedhula, K., and J. Rabadan, "Yang Data Model for EVPN", draft-ietf-bess-evpn-yang-05 (work in progress), February 2018.

- [I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B.,
and K. Tiruveedhula, "YANG Data Model for MPLS-based
L2VPN", draft-ietf-bess-l2vpn-yang-08 (work in progress),
February 2018.
- [I-D.ietf-bess-l3vpn-yang]
Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S.,
Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model
for BGP/MPLS L3 VPNs", draft-ietf-bess-l3vpn-yang-02 (work
in progress), October 2017.
- [I-D.ietf-l2sm-l2vpn-service-model]
Wen, B., Fioccola, G., Xie, C., and L. Jalil, "A YANG Data
Model for L2VPN Service Delivery", draft-ietf-l2sm-l2vpn-
service-model-08 (work in progress), February 2018.
- [I-D.dhodylee-pce-pcep-ls]
Dhody, D., Lee, Y., and D. Ceccarelli, "PCEP Extension for
Distribution of Link-State and TE Information.", draft-
dhodylee-pce-pcep-ls-09 (work in progress), January 2018.
- [I-D.lee-teas-te-service-mapping-yang]
Lee, Y., Dhody, D., Ceccarelli, D., Tantsura, J., and G.
Fioccola, "Traffic Engineering and Service Mapping Yang
Model", draft-lee-teas-te-service-mapping-yang-06 (work in
progress), February 2018.
- [I-D.lee-pce-lsp-stitching-hpce]
Lee, Y., Dhody, D., and D. Ceccarelli, "PCEP Extensions
for Stitching LSPs in Hierarchical Stateful PCE Model",
draft-lee-pce-lsp-stitching-hpce-01 (work in progress),
December 2017.
- [I-D.dugeon-pce-stateful-interdomain]
Dugeon, O. and J. Meuric, "PCEP Extension for Stateful
Inter-Domain Tunnels", draft-dugeon-pce-stateful-
interdomain-00 (work in progress), October 2017.

Authors' Addresses

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

EMail: lizhenbin@huawei.com

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: dhruv.ietf@gmail.com

Huaimo Chen
Huawei Technologies
Boston, MA
USA

EMail: huaimo.chen@huawei.com