TEAS Working Group
Internet Draft
Intended status: Standards Track

Xufeng Liu
Ericsson
Igor Bryskin
ADVA Optical Networking
Vishnu Pavan Beeram
Juniper Networks
Tarek Saad
Cisco Systems Inc
Himanshu Shah
Ciena
Oscar Gonzalez De Dios
Telefonica

Expires: September 09, 2015 March 09, 2015

YANG Data Model for TE Topologies
draft-liu-teas-yang-te-topo-00

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/1id-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 09, 2015.

Copyright Notice

Copyright (c) 2015 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 defines a YANG data model for representing and manipulating TE Topologies.

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. Introduction....2
   1.1. Tree Structure - Legend....3
   1.2. Prefixes in Data Node Names....4
2. Design Considerations....4
   2.1. Extensible Model....4
   2.2. Overlay/Underlay Relationship....4
   2.3. Scheduling Parameters....4
   2.4. Provider vs Abstract TE Topologies....5
   2.5. Open Items....6
3. Tree Structure....6
4. TE Topology Yang Module....42
5. Security Considerations....68
6. IANA Considerations....68
7. Normative References....69
8. Acknowledgments....69

1. Introduction

YANG [RFC6020] 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 interface, such as CLI and programmatic APIs.

This document defines a YANG data model for representing and manipulating TE Topologies. This model contains technology agnostic TE Topology building blocks that can be augmented and used by other technology-specific TE Topology models.

1.1. Tree Structure - Legend

A simplified graphical representation of the data model is presented in Section 3 of this document. 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.

1.2. 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]

Table 1: Prefixes and corresponding YANG modules

2. Design Considerations

2.1. Extensible Model

The TE Topology model proposed in this document is meant to be technology agnostic. Other technology specific TE Topology models can augment and use the building blocks provided by the proposed model.

2.2. Overlay/Underlay Relationship

The model captures overlay and underlay relationship for TE nodes/links. For example - in networks where multiple TE Topologies are built hierarchically, this model allows the user to start from a specific topological element in the top most topology and traverse all the way down to the supporting topological elements in the bottom most topology.

2.3. Scheduling Parameters

The model allows time scheduling parameters to be specified for each topological element. These parameters allow the provider to present different topological views to the client at different time slots.

```
+--rw schedules* [schedule-id]
|  +--rw schedule-id          uint32
|  +--rw start?               yang:date-and-time
|  +--rw schedule-duration?   string
```

```
      |  +--rw repeat-interval?     string
```

2.4. Provider vs Abstract TE Topologies

The TE Topology exported by the provider network to a client could be a provider TE Topology or an abstract TE Topology. Provider TE Topology is the TE topology exported by the provider network exactly as the topology is known to the provider itself in an unmodified way. The nodes/links that make up a provider TE topology are named from the provider name space and are private to that topology.

```
 +--:(provider)
   +--rw provider
   +--rw node* [te-node-id]
   |  +--rw te-node-id            te-node-id
   |      :
   +--rw link* [source-te-node-id source-te-link-id dest-te-node-id
dest-te-link-id]
      +-rw source-te-node-id      leafref
      +--rw source-te-link-id     leafref
      +--rw dest-te-node-id       leafref
      +--rw dest-te-link-id       leafref
          :
```

An abstract TE Topology allows the provider to present the network in abstract terms on per client basis. These topologies are typically decoupled from the provider network topology. The nodes/links that constitute an abstract TE Topology are picked from a global pool of nodes/links. The nodes/links in the global pool can be used in more than one abstract TE Topologies.

```
+--:(abstract)
   +--rw abstract
   +--rw node* [te-node-id]
   |  +--rw te-node-uu-id    leafref
   |  +--rw te-node-id       leafref
   +--rw link* [source-te-node-id source-te-link-id dest-te-node-id
dest-te-link-id]
      +--rw source-te-node-id    leafref
      +--rw source-te-link-id    leafref
      +--rw dest-te-node-id      leafref
      +--rw dest-te-link-id      leafref
```

2.5. Open Items

- Relationship with "generic network topology" model: The generic network topology building blocks are discussed in [YANG-NET-TOPO]. This version of the document does not use any of those building blocks. The authors would like to explore the possibility of doing that in future revisions of this document.

- Incremental notifications: The model proposed in this version does not cover incremental notifications. The authors intend to add this in future revisions of the document.

- Terminology: A provider network can export its TE Topology as is to the client or it could export an abstract representation of that TE Topology. This version of the document uses the term "Provider TE Topology" for the former and the term "Abstract TE Topology" for the latter. The authors would like to get wider consensus on the use of these terms (and if necessary, find alternatives) before the next revision.

3. Tree Structure

```
module: ietf-te-topology

   +--rw te-topologies
   |  +--rw topology* [te-topology-id]
   |  |  +--rw te-topology-id    te-topology-id
   |  |  +--rw topology-types
   |  |  |  +--rw te-topology!
   |  |  +--rw (abstract-type)?
   |  |     +--:(abstract)
   |  |     |  +--rw abstract
   |  |     |     +--rw node* [te-node-id]
   |  |     |     |  +--rw te-node-uu-id    leafref
   |  |     |     |  +--rw te-node-id       leafref
   |  |     |     +--rw link* [source-te-node-id source-te-link-id
dest-te-node-id dest-te-link-id]
   |  |     |        +--rw source-te-node-id    leafref
   |  |     |        +--rw source-te-link-id    leafref
   |  |     |        +--rw dest-te-node-id      leafref
   |  |     |        +--rw dest-te-link-id      leafref
   |  |     +--:(provider)
   |  |        +--rw provider
   |  |           +--rw node* [te-node-id]
   |  |           |  +--rw te-node-id            te-node-id
   |  |           |  +--rw te-node-attributes
```

```
      |  |           |     +--rw schedules* [schedule-id]
      |  |           |     |  +--rw schedule-id          uint32
      |  |           |     |  +--rw start?               yang:date-and-
   time
      |  |           |     |  +--rw schedule-duration?   string
      |  |           |     |  +--rw repeat-interval?     string
      |  |           |     +--rw name?                  inet:domain-
   name
      |  |           |     +--rw signaling-address*     inet:ip-address
      |  |           |     +--rw flag*                  flag-type
      |  |           |     +--rw is-abstract?           boolean
      |  |           |     +--rw underlay-topology?     leafref {te-
   topology-hierarchy}?
      |  |           |     +--rw te-link* [te-link-id]
      |  |           |     |  +--rw te-link-id         te-link-id
      |  |           |     |  +--rw (stack-level)?
      |  |           |     |     +--:(bundle)
      |  |           |     |     |  +--rw bundled-links
      |  |           |     |     |     +--rw bundled-link* [sequence]
      |  |           |     |     |        +--rw sequence       uint32
      |  |           |     |     |        +--rw te-link-ref?   leafref
      |  |           |     |     +--:(component)
      |  |           |     |        +--rw component-links
      |  |           |     |           +--rw component-link* [sequence]
      |  |           |     |              +--rw sequence
   uint32
      |  |           |     |              +--rw component-link-ref?
   leafref
      |  |           |     +--rw connectivity-matrix* [id]
      |  |           |     |  +--rw id            uint32
      |  |           |     |  +--rw from-link
      |  |           |     |  |  +--rw node-ref?       leafref
      |  |           |     |  |  +--rw link-end-ref?   leafref
      |  |           |     |  +--rw to-link
      |  |           |     |  |  +--rw node-ref?       leafref
      |  |           |     |  |  +--rw link-end-ref?   leafref
      |  |           |     |  +--rw is-allowed?   boolean
      |  |           |     +--rw ted
      |  |           |        +--rw te-router-id-ipv4?    inet:ipv4-
   address
      |  |           |        +--rw te-router-id-ipv6?    inet:ipv6-
   address
      |  |           |        +--rw ipv4-local-address* [ipv4-prefix]
      |  |           |        |  +--rw ipv4-prefix    inet:ipv4-prefix
      |  |           |        +--rw ipv6-local-address* [ipv6-prefix]
```

```
      |  |           |        |  +--rw ipv6-prefix      inet:ipv6-
   prefix
      |  |           |        |  +--rw prefix-option?   uint8
      |  |           |        +--rw pcc-capabilities?     pcc-
   capabilities
      |  |           +--rw link* [source-te-node-id source-te-link-id
   dest-te-node-id dest-te-link-id]
      |  |              +--rw source-te-node-id     leafref
      |  |              +--rw source-te-link-id     leafref
      |  |              +--rw dest-te-node-id       leafref
      |  |              +--rw dest-te-link-id       leafref
      |  |              +--rw te-link-attributes
      |  |                 +--rw schedules* [schedule-id]
      |  |                 |  +--rw schedule-id          uint32
      |  |                 |  +--rw start?               yang:date-and-
   time
      |  |                 |  +--rw schedule-duration?   string
      |  |                 |  +--rw repeat-interval?     string
      |  |                 +--rw name?          string
      |  |                 +--rw flag*          flag-type
      |  |                 +--rw is-abstract?   boolean
      |  |                 +--rw underlay! {te-topology-hierarchy}?
      |  |                 |  +--rw underlay-path
      |  |                 |  |  +--rw topology-id?    leafref
      |  |                 |  |  +--rw path-element* [path-element-id]
      |  |                 |  |     +--rw path-element-id    uint32
      |  |                 |  |     +--rw loose?             boolean
      |  |                 |  |     +--rw (element-type)?
      |  |                 |  |        +--:(numbered-link)
      |  |                 |  |        |  +--rw link-ip-address?
   inet:ip-address
      |  |                 |  |        +--:(unnumbered-link)
      |  |                 |  |        |  +--rw link-node-id?
   uint32
      |  |                 |  |        |  +--rw te-link-id?
   uint32
      |  |                 |  |        +--:(node)
      |  |                 |  |        |  +--rw te-node-id?
   uint32
      |  |                 |  |        +--:(label)
      |  |                 |  |           +--rw label?
   uint32
      |  |                 |  +--rw underlay-backup-path
      |  |                 |  |  +--rw topology-id?    leafref
      |  |                 |  |  +--rw path-element* [path-element-id]
      |  |                 |  |     +--rw path-element-id    uint32
```

```
      |  |                 |  |     +--rw loose?             boolean
      |  |                 |  |     +--rw (element-type)?
      |  |                 |  |        +--:(numbered-link)
      |  |                 |  |        |  +--rw link-ip-address?
   inet:ip-address
      |  |                 |  |        +--:(unnumbered-link)
      |  |                 |  |        |  +--rw link-node-id?
   uint32
      |  |                 |  |        |  +--rw te-link-id?
   uint32
      |  |                 |  |        +--:(node)
      |  |                 |  |        |  +--rw te-node-id?
   uint32
      |  |                 |  |        +--:(label)
      |  |                 |  |           +--rw label?
   uint32
      |  |                 |  +--rw underlay-protection-type?   uint16
      |  |                 |  +--rw underlay-trail-src
      |  |                 |  |  +--rw node-ref?       leafref
      |  |                 |  |  +--rw link-end-ref?   leafref
      |  |                 |  +--rw underlay-trail-des
      |  |                 |     +--rw node-ref?       leafref
      |  |                 |     +--rw link-end-ref?   leafref
      |  |                 +--rw ted
      |  |                    +--rw admin-status?
   enumeration
      |  |                    +--rw oper-status?
   enumeration
      |  |                    +--rw area-id?
   binary
      |  |                    +--rw performance-metric-throttle {te-
   performance-metric}?
      |  |                    |  +--rw unidirectional-delay-offset?
   uint32
      |  |                    |  +--rw measure-interval?
   uint32
      |  |                    |  +--rw advertisement-interval?
   uint32
      |  |                    |  +--rw suppression-interval?
   uint32
      |  |                    |  +--rw threshold-out
      |  |                    |  |  +--rw unidirectional-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-min-delay?
   uint32
```

```
      |  |                    |  |  +--rw unidirectional-max-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-delay-
   variation?       uint32
      |  |                    |  |  +--rw unidirectional-packet-loss?
   decimal64
      |  |                    |  |  +--rw unidirectional-residual-
   bandwidth?    decimal64
      |  |                    |  |  +--rw unidirectional-available-
   bandwidth?   decimal64
      |  |                    |  |  +--rw unidirectional-utilized-
   bandwidth?    decimal64
      |  |                    |  +--rw threshold-in
      |  |                    |  |  +--rw unidirectional-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-min-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-max-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-delay-
   variation?       uint32
      |  |                    |  |  +--rw unidirectional-packet-loss?
   decimal64
      |  |                    |  |  +--rw unidirectional-residual-
   bandwidth?    decimal64
      |  |                    |  |  +--rw unidirectional-available-
   bandwidth?   decimal64
      |  |                    |  |  +--rw unidirectional-utilized-
   bandwidth?    decimal64
      |  |                    |  +--rw threshold-accelerated-
   advertisement
      |  |                    |     +--rw unidirectional-delay?
   uint32
      |  |                    |     +--rw unidirectional-min-delay?
   uint32
      |  |                    |     +--rw unidirectional-max-delay?
   uint32
      |  |                    |     +--rw unidirectional-delay-
   variation?       uint32
      |  |                    |     +--rw unidirectional-packet-loss?
   decimal64
      |  |                    |     +--rw unidirectional-residual-
   bandwidth?    decimal64
      |  |                    |     +--rw unidirectional-available-
   bandwidth?   decimal64
```

```
      |  |                    |     +--rw unidirectional-utilized-
   bandwidth?    decimal64
      |  |                    +--rw information-source?
   enumeration
      |  |                    +--rw credibility-preference?
   uint16
      |  |                    +--rw link-index?
   uint64
      |  |                    +--rw administrative-group* [sequence]
      |  |                    |  +--rw sequence      uint32
      |  |                    |  +--rw ag-element?   uint32
      |  |                    +--rw max-link-bandwidth?
   decimal64
      |  |                    +--rw max-resv-link-bandwidth?
   decimal64
      |  |                    +--rw unreserved-bandwidth* [priority]
      |  |                    |  +--rw priority     uint8
      |  |                    |  +--rw bandwidth?   decimal64
      |  |                    +--rw te-default-metric?
   uint32
      |  |                    +--rw performance-metric {te-performance-
   metric}?
      |  |                    |  +--rw measurement
      |  |                    |  |  +--rw unidirectional-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-min-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-max-delay?
   uint32
      |  |                    |  |  +--rw unidirectional-delay-
   variation?       uint32
      |  |                    |  |  +--rw unidirectional-packet-loss?
   decimal64
      |  |                    |  |  +--rw unidirectional-residual-
   bandwidth?    decimal64
      |  |                    |  |  +--rw unidirectional-available-
   bandwidth?   decimal64
      |  |                    |  |  +--rw unidirectional-utilized-
   bandwidth?    decimal64
      |  |                    |  +--rw normality
      |  |                    |     +--rw unidirectional-delay?
   performance-metric-normality
      |  |                    |     +--rw unidirectional-min-delay?
   performance-metric-normality
      |  |                    |     +--rw unidirectional-max-delay?
   performance-metric-normality
```

```
      |  |                    |     +--rw unidirectional-delay-
   variation?       performance-metric-normality
      |  |                    |     +--rw unidirectional-packet-loss?
   performance-metric-normality
      |  |                    |     +--rw unidirectional-residual-
   bandwidth?    performance-metric-normality
      |  |                    |     +--rw unidirectional-available-
   bandwidth?   performance-metric-normality
      |  |                    |     +--rw unidirectional-utilized-
   bandwidth?    performance-metric-normality
      |  |                    +--rw link-protection-type?
   enumeration
      |  |                    +--rw interface-switching-capabilities*
   [switching-capability]
      |  |                    |  +--rw switching-capability
   ted:switching-capabilities
      |  |                    |  +--rw encoding?
   ted:encoding-type
      |  |                    |  +--rw max-lsp-bandwidth* [priority]
      |  |                    |  |  +--rw priority     uint8
      |  |                    |  |  +--rw bandwidth?   decimal64
      |  |                    |  +--rw packet-switch-capable
      |  |                    |  |  +--rw minimum-lsp-bandwidth?
   decimal64
      |  |                    |  |  +--rw interface-mtu?
   uint16
      |  |                    |  +--rw time-division-multiplex-capable
      |  |                    |     +--rw minimum-lsp-bandwidth?
   decimal64
      |  |                    |     +--rw indication?
   enumeration
      |  |                    +--rw srlg
      |  |                    |  +--rw srlg-values* [srlg-value]
      |  |                    |     +--rw srlg-value    uint32
      |  |                    +--rw alt-information-sources*
   [information-source]
      |  |                       +--rw information-source
   enumeration
      |  |                       +--rw credibility-preference?
   uint16
      |  |                       +--rw link-index?
   uint64
      |  |                       +--rw administrative-group* [sequence]
      |  |                       |  +--rw sequence      uint32
      |  |                       |  +--rw ag-element?   uint32
```

```
      |  |                      +--rw max-link-bandwidth?
   decimal64
      |  |                      +--rw max-resv-link-bandwidth?
   decimal64
      |  |                      +--rw unreserved-bandwidth* [priority]
      |  |                      |  +--rw priority     uint8
      |  |                      |  +--rw bandwidth?   decimal64
      |  |                      +--rw te-default-metric?
   uint32
      |  |                      +--rw performance-metric {te-
   performance-metric}?
      |  |                      |  +--rw measurement
      |  |                      |  |  +--rw unidirectional-delay?
   uint32
      |  |                      |  |  +--rw unidirectional-min-delay?
   uint32
      |  |                      |  |  +--rw unidirectional-max-delay?
   uint32
      |  |                      |  |  +--rw unidirectional-delay-
   variation?       uint32
      |  |                      |  |  +--rw unidirectional-packet-
   loss?           decimal64
      |  |                      |  |  +--rw unidirectional-residual-
   bandwidth?    decimal64
      |  |                      |  |  +--rw unidirectional-available-
   bandwidth?   decimal64
      |  |                      |  |  +--rw unidirectional-utilized-
   bandwidth?    decimal64
      |  |                      |  +--rw normality
      |  |                      |     +--rw unidirectional-delay?
   performance-metric-normality
      |  |                      |     +--rw unidirectional-min-delay?
   performance-metric-normality
      |  |                      |     +--rw unidirectional-max-delay?
   performance-metric-normality
      |  |                      |     +--rw unidirectional-delay-
   variation?       performance-metric-normality
      |  |                      |     +--rw unidirectional-packet-
   loss?           performance-metric-normality
      |  |                      |     +--rw unidirectional-residual-
   bandwidth?    performance-metric-normality
      |  |                      |     +--rw unidirectional-available-
   bandwidth?   performance-metric-normality
      |  |                      |     +--rw unidirectional-utilized-
   bandwidth?    performance-metric-normality
```

```
      |  |                       +--rw link-protection-type?
   enumeration
      |  |                       +--rw interface-switching-
   capabilities* [switching-capability]
      |  |                       |  +--rw switching-capability
   ted:switching-capabilities
      |  |                       |  +--rw encoding?
   ted:encoding-type
      |  |                       |  +--rw max-lsp-bandwidth* [priority]
      |  |                       |  |  +--rw priority     uint8
      |  |                       |  |  +--rw bandwidth?   decimal64
      |  |                       |  +--rw packet-switch-capable
      |  |                       |  |  +--rw minimum-lsp-bandwidth?
   decimal64
      |  |                       |  |  +--rw interface-mtu?
   uint16
      |  |                       |  +--rw time-division-multiplex-
   capable
      |  |                       |     +--rw minimum-lsp-bandwidth?
   decimal64
      |  |                       |     +--rw indication?
   enumeration
      |  |                       +--rw srlg
      |  |                          +--rw srlg-values* [srlg-value]
      |  |                             +--rw srlg-value    uint32
      |  +--rw node* [te-node-uu-id]
      |  |  +--rw te-node-uu-id         yang:uuid
      |  |  +--rw te-node-id?           te-node-id
      |  |  +--rw te-node-attributes
      |  |     +--rw schedules* [schedule-id]
      |  |     |  +--rw schedule-id          uint32
      |  |     |  +--rw start?               yang:date-and-time
      |  |     |  +--rw schedule-duration?   string
      |  |     |  +--rw repeat-interval?     string
      |  |     +--rw name?                  inet:domain-name
      |  |     +--rw signaling-address*     inet:ip-address
      |  |     +--rw flag*                  flag-type
      |  |     +--rw is-abstract?           boolean
      |  |     +--rw underlay-topology?     leafref {te-topology-
   hierarchy}?
      |  |     +--rw te-link* [te-link-id]
      |  |     |  +--rw te-link-id         te-link-id
      |  |     |  +--rw (stack-level)?
      |  |     |     +--:(bundle)
      |  |     |     |  +--rw bundled-links
      |  |     |     |     +--rw bundled-link* [sequence]
```

```
      |  |     |     |        +--rw sequence       uint32
      |  |     |     |        +--rw te-link-ref?   leafref
      |  |     |     +--:(component)
      |  |     |        +--rw component-links
      |  |     |           +--rw component-link* [sequence]
      |  |     |              +--rw sequence              uint32
      |  |     |              +--rw component-link-ref?   leafref
      |  |     +--rw connectivity-matrix* [id]
      |  |     |  +--rw id            uint32
      |  |     |  +--rw from-link
      |  |     |  |  +--rw node-ref?       leafref
      |  |     |  |  +--rw link-end-ref?   leafref
      |  |     |  +--rw to-link
      |  |     |  |  +--rw node-ref?       leafref
      |  |     |  |  +--rw link-end-ref?   leafref
      |  |     |  +--rw is-allowed?   boolean
      |  |     +--rw ted
      |  |        +--rw te-router-id-ipv4?    inet:ipv4-address
      |  |        +--rw te-router-id-ipv6?    inet:ipv6-address
      |  |        +--rw ipv4-local-address* [ipv4-prefix]
      |  |        |  +--rw ipv4-prefix    inet:ipv4-prefix
      |  |        +--rw ipv6-local-address* [ipv6-prefix]
      |  |        |  +--rw ipv6-prefix      inet:ipv6-prefix
      |  |        |  +--rw prefix-option?   uint8
      |  |        +--rw pcc-capabilities?     pcc-capabilities
      |  +--rw link* [source-te-node-id source-te-link-id dest-te-node-
   id dest-te-link-id]
      |     +--rw source-te-node-id     leafref
      |     +--rw source-te-link-id     leafref
      |     +--rw dest-te-node-id       leafref
      |     +--rw dest-te-link-id       leafref
      |     +--rw te-link-attributes
      |        +--rw schedules* [schedule-id]
      |        |  +--rw schedule-id          uint32
      |        |  +--rw start?               yang:date-and-time
      |        |  +--rw schedule-duration?   string
      |        |  +--rw repeat-interval?     string
      |        +--rw name?          string
      |        +--rw flag*          flag-type
      |        +--rw is-abstract?   boolean
      |        +--rw underlay! {te-topology-hierarchy}?
      |        |  +--rw underlay-path
      |        |  |  +--rw topology-id?    leafref
      |        |  |  +--rw path-element* [path-element-id]
      |        |  |     +--rw path-element-id    uint32
      |        |  |     +--rw loose?             boolean
```

```
      |        |  |     +--rw (element-type)?
      |        |  |        +--:(numbered-link)
      |        |  |        |  +--rw link-ip-address?   inet:ip-address
      |        |  |        +--:(unnumbered-link)
      |        |  |        |  +--rw link-node-id?      uint32
      |        |  |        |  +--rw te-link-id?        uint32
      |        |  |        +--:(node)
      |        |  |        |  +--rw te-node-id?        uint32
      |        |  |        +--:(label)
      |        |  |           +--rw label?            uint32
      |        |  +--rw underlay-backup-path
      |        |  |  +--rw topology-id?    leafref
      |        |  |  +--rw path-element* [path-element-id]
      |        |  |     +--rw path-element-id    uint32
      |        |  |     +--rw loose?             boolean
      |        |  |     +--rw (element-type)?
      |        |  |        +--:(numbered-link)
      |        |  |        |  +--rw link-ip-address?   inet:ip-address
      |        |  |        +--:(unnumbered-link)
      |        |  |        |  +--rw link-node-id?      uint32
      |        |  |        |  +--rw te-link-id?        uint32
      |        |  |        +--:(node)
      |        |  |        |  +--rw te-node-id?        uint32
      |        |  |        +--:(label)
      |        |  |           +--rw label?            uint32
      |        |  +--rw underlay-protection-type?   uint16
      |        |  +--rw underlay-trail-src
      |        |  |  +--rw node-ref?       leafref
      |        |  |  +--rw link-end-ref?   leafref
      |        |  +--rw underlay-trail-des
      |        |     +--rw node-ref?       leafref
      |        |     +--rw link-end-ref?   leafref
      |        +--rw ted
      |           +--rw admin-status?                       enumeration
      |           +--rw oper-status?                        enumeration
      |           +--rw area-id?                            binary
      |           +--rw performance-metric-throttle {te-performance-
   metric}?
      |           |  +--rw unidirectional-delay-offset?
   uint32
      |           |  +--rw measure-interval?
   uint32
      |           |  +--rw advertisement-interval?
   uint32
      |           |  +--rw suppression-interval?
   uint32
```

```
      |           |  +--rw threshold-out
      |           |  |  +--rw unidirectional-delay?
   uint32
      |           |  |  +--rw unidirectional-min-delay?
   uint32
      |           |  |  +--rw unidirectional-max-delay?
   uint32
      |           |  |  +--rw unidirectional-delay-variation?
   uint32
      |           |  |  +--rw unidirectional-packet-loss?
   decimal64
      |           |  |  +--rw unidirectional-residual-bandwidth?
   decimal64
      |           |  |  +--rw unidirectional-available-bandwidth?
   decimal64
      |           |  |  +--rw unidirectional-utilized-bandwidth?
   decimal64
      |           |  +--rw threshold-in
      |           |  |  +--rw unidirectional-delay?
   uint32
      |           |  |  +--rw unidirectional-min-delay?
   uint32
      |           |  |  +--rw unidirectional-max-delay?
   uint32
      |           |  |  +--rw unidirectional-delay-variation?
   uint32
      |           |  |  +--rw unidirectional-packet-loss?
   decimal64
      |           |  |  +--rw unidirectional-residual-bandwidth?
   decimal64
      |           |  |  +--rw unidirectional-available-bandwidth?
   decimal64
      |           |  |  +--rw unidirectional-utilized-bandwidth?
   decimal64
      |           |  +--rw threshold-accelerated-advertisement
      |           |     +--rw unidirectional-delay?
   uint32
      |           |     +--rw unidirectional-min-delay?
   uint32
      |           |     +--rw unidirectional-max-delay?
   uint32
      |           |     +--rw unidirectional-delay-variation?
   uint32
      |           |     +--rw unidirectional-packet-loss?
   decimal64
```

```
      |           |     +--rw unidirectional-residual-bandwidth?
   decimal64
      |           |     +--rw unidirectional-available-bandwidth?
   decimal64
      |           |     +--rw unidirectional-utilized-bandwidth?
   decimal64
      |           +--rw information-source?                 enumeration
      |           +--rw credibility-preference?             uint16
      |           +--rw link-index?                         uint64
      |           +--rw administrative-group* [sequence]
      |           |  +--rw sequence      uint32
      |           |  +--rw ag-element?   uint32
      |           +--rw max-link-bandwidth?                 decimal64
      |           +--rw max-resv-link-bandwidth?            decimal64
      |           +--rw unreserved-bandwidth* [priority]
      |           |  +--rw priority     uint8
      |           |  +--rw bandwidth?   decimal64
      |           +--rw te-default-metric?                  uint32
      |           +--rw performance-metric {te-performance-metric}?
      |           |  +--rw measurement
      |           |  |  +--rw unidirectional-delay?
   uint32
      |           |  |  +--rw unidirectional-min-delay?
   uint32
      |           |  |  +--rw unidirectional-max-delay?
   uint32
      |           |  |  +--rw unidirectional-delay-variation?
   uint32
      |           |  |  +--rw unidirectional-packet-loss?
   decimal64
      |           |  |  +--rw unidirectional-residual-bandwidth?
   decimal64
      |           |  |  +--rw unidirectional-available-bandwidth?
   decimal64
      |           |  |  +--rw unidirectional-utilized-bandwidth?
   decimal64
      |           |  +--rw normality
      |           |     +--rw unidirectional-delay?
   performance-metric-normality
      |           |     +--rw unidirectional-min-delay?
   performance-metric-normality
      |           |     +--rw unidirectional-max-delay?
   performance-metric-normality
      |           |     +--rw unidirectional-delay-variation?
   performance-metric-normality
```

```
      |           |     +--rw unidirectional-packet-loss?
   performance-metric-normality
      |           |     +--rw unidirectional-residual-bandwidth?
   performance-metric-normality
      |           |     +--rw unidirectional-available-bandwidth?
   performance-metric-normality
      |           |     +--rw unidirectional-utilized-bandwidth?
   performance-metric-normality
      |           +--rw link-protection-type?               enumeration
      |           +--rw interface-switching-capabilities* [switching-
   capability]
      |           |  +--rw switching-capability
   ted:switching-capabilities
      |           |  +--rw encoding?
   ted:encoding-type
      |           |  +--rw max-lsp-bandwidth* [priority]
      |           |  |  +--rw priority     uint8
      |           |  |  +--rw bandwidth?   decimal64
      |           |  +--rw packet-switch-capable
      |           |  |  +--rw minimum-lsp-bandwidth?   decimal64
      |           |  |  +--rw interface-mtu?           uint16
      |           |  +--rw time-division-multiplex-capable
      |           |     +--rw minimum-lsp-bandwidth?   decimal64
      |           |     +--rw indication?              enumeration
      |           +--rw srlg
      |           |  +--rw srlg-values* [srlg-value]
      |           |     +--rw srlg-value    uint32
      |           +--rw alt-information-sources* [information-source]
      |              +--rw information-source
   enumeration
      |              +--rw credibility-preference?             uint16
      |              +--rw link-index?                         uint64
      |              +--rw administrative-group* [sequence]
      |              |  +--rw sequence      uint32
      |              |  +--rw ag-element?   uint32
      |              +--rw max-link-bandwidth?
   decimal64
      |              +--rw max-resv-link-bandwidth?
   decimal64
      |              +--rw unreserved-bandwidth* [priority]
      |              |  +--rw priority     uint8
      |              |  +--rw bandwidth?   decimal64
      |              +--rw te-default-metric?                  uint32
      |              +--rw performance-metric {te-performance-metric}?
      |              |  +--rw measurement
```

```
      |              |  |  +--rw unidirectional-delay?
   uint32
      |              |  |  +--rw unidirectional-min-delay?
   uint32
      |              |  |  +--rw unidirectional-max-delay?
   uint32
      |              |  |  +--rw unidirectional-delay-variation?
   uint32
      |              |  |  +--rw unidirectional-packet-loss?
   decimal64
      |              |  |  +--rw unidirectional-residual-bandwidth?
   decimal64
      |              |  |  +--rw unidirectional-available-bandwidth?
   decimal64
      |              |  |  +--rw unidirectional-utilized-bandwidth?
   decimal64
      |              |  +--rw normality
      |              |     +--rw unidirectional-delay?
   performance-metric-normality
      |              |     +--rw unidirectional-min-delay?
   performance-metric-normality
      |              |     +--rw unidirectional-max-delay?
   performance-metric-normality
      |              |     +--rw unidirectional-delay-variation?
   performance-metric-normality
      |              |     +--rw unidirectional-packet-loss?
   performance-metric-normality
      |              |     +--rw unidirectional-residual-bandwidth?
   performance-metric-normality
      |              |     +--rw unidirectional-available-bandwidth?
   performance-metric-normality
      |              |     +--rw unidirectional-utilized-bandwidth?
   performance-metric-normality
      |              +--rw link-protection-type?
   enumeration
      |              +--rw interface-switching-capabilities*
   [switching-capability]
      |              |  +--rw switching-capability
   ted:switching-capabilities
      |              |  +--rw encoding?
   ted:encoding-type
      |              |  +--rw max-lsp-bandwidth* [priority]
      |              |  |  +--rw priority     uint8
      |              |  |  +--rw bandwidth?   decimal64
      |              |  +--rw packet-switch-capable
      |              |  |  +--rw minimum-lsp-bandwidth?   decimal64
```

```
      |              |  |  +--rw interface-mtu?           uint16
      |              |  +--rw time-division-multiplex-capable
      |              |     +--rw minimum-lsp-bandwidth?   decimal64
      |              |     +--rw indication?              enumeration
      |              +--rw srlg
      |                 +--rw srlg-values* [srlg-value]
      |                    +--rw srlg-value    uint32
      +--ro te-topologies-state
         +--ro topology* [te-topology-id]
         |  +--ro te-topology-id     te-topology-id
         |  +--ro server-provided?   boolean
         |  +--ro topology-types
         |  |  +--ro te-topology!
         |  +--ro (abstract-type)?
         |     +--:(abstract)
         |     |  +--ro abstract
         |     |     +--ro node* [te-node-id]
         |     |     |  +--ro te-node-uu-id    leafref
         |     |     |  +--ro te-node-id       leafref
         |     |     +--ro link* [source-te-node-id source-te-link-id
   dest-te-node-id dest-te-link-id]
         |     |        +--ro source-te-node-id    leafref
         |     |        +--ro source-te-link-id    leafref
         |     |        +--ro dest-te-node-id      leafref
         |     |        +--ro dest-te-link-id      leafref
         |     +--:(provider)
         |        +--ro provider
         |           +--ro node* [te-node-id]
         |           |  +--ro te-node-id                  te-node-id
         |           |  +--ro te-node-attributes
         |           |  |  +--ro schedules* [schedule-id]
         |           |  |  |  +--ro schedule-id          uint32
         |           |  |  |  +--ro start?               yang:date-and-
   time
         |           |  |  |  +--ro schedule-duration?   string
         |           |  |  |  +--ro repeat-interval?     string
         |           |  |  +--ro name?                  inet:domain-
   name
         |           |  |  +--ro signaling-address*     inet:ip-address
         |           |  |  +--ro flag*                  flag-type
         |           |  |  +--ro is-abstract?           boolean
         |           |  |  +--ro underlay-topology?     leafref {te-
   topology-hierarchy}?
         |           |  |  +--ro te-link* [te-link-id]
         |           |  |  |  +--ro te-link-id         te-link-id
         |           |  |  |  +--ro (stack-level)?
```

```
         |           |  |  |     +--:(bundle)
         |           |  |  |     |  +--ro bundled-links
         |           |  |  |     |     +--ro bundled-link* [sequence]
         |           |  |  |     |        +--ro sequence       uint32
         |           |  |  |     |        +--ro te-link-ref?   leafref
         |           |  |  |     +--:(component)
         |           |  |  |        +--ro component-links
         |           |  |  |           +--ro component-link* [sequence]
         |           |  |  |              +--ro sequence
   uint32
         |           |  |  |              +--ro component-link-ref?
   leafref
         |           |  |  +--ro connectivity-matrix* [id]
         |           |  |  |  +--ro id            uint32
         |           |  |  |  +--ro from-link
         |           |  |  |  |  +--ro node-ref?       leafref
         |           |  |  |  |  +--ro link-end-ref?   leafref
         |           |  |  |  +--ro to-link
         |           |  |  |  |  +--ro node-ref?       leafref
         |           |  |  |  |  +--ro link-end-ref?   leafref
         |           |  |  |  +--ro is-allowed?   boolean
         |           |  |  +--ro ted
         |           |  |     +--ro te-router-id-ipv4?    inet:ipv4-
   address
         |           |  |     +--ro te-router-id-ipv6?    inet:ipv6-
   address
         |           |  |     +--ro ipv4-local-address* [ipv4-prefix]
         |           |  |     |  +--ro ipv4-prefix    inet:ipv4-prefix
         |           |  |     +--ro ipv6-local-address* [ipv6-prefix]
         |           |  |     |  +--ro ipv6-prefix      inet:ipv6-
   prefix
         |           |  |     |  +--ro prefix-option?   uint8
         |           |  |     +--ro pcc-capabilities?     pcc-
   capabilities
         |           |  +--ro te-node-state-attributes
         |           |     +--ro information-source?       enumeration
         |           |     +--ro credibility-preference?   uint16
         |           +--ro link* [source-te-node-id source-te-link-id
   dest-te-node-id dest-te-link-id]
         |              +--ro source-te-node-id           leafref
         |              +--ro source-te-link-id           leafref
         |              +--ro dest-te-node-id             leafref
         |              +--ro dest-te-link-id             leafref
         |              +--ro te-link-attributes
         |              |  +--ro schedules* [schedule-id]
         |              |  |  +--ro schedule-id          uint32
```

```
         |              |  |  +--ro start?               yang:date-and-
   time
         |              |  |  +--ro schedule-duration?   string
         |              |  |  +--ro repeat-interval?     string
         |              |  +--ro name?          string
         |              |  +--ro flag*          flag-type
         |              |  +--ro is-abstract?   boolean
         |              |  +--ro underlay! {te-topology-hierarchy}?
         |              |  |  +--ro underlay-path
         |              |  |  |  +--ro topology-id?    leafref
         |              |  |  |  +--ro path-element* [path-element-id]
         |              |  |  |     +--ro path-element-id    uint32
         |              |  |  |     +--ro loose?             boolean
         |              |  |  |     +--ro (element-type)?
         |              |  |  |        +--:(numbered-link)
         |              |  |  |        |  +--ro link-ip-address?
   inet:ip-address
         |              |  |  |        +--:(unnumbered-link)
         |              |  |  |        |  +--ro link-node-id?
   uint32
         |              |  |  |        |  +--ro te-link-id?
   uint32
         |              |  |  |        +--:(node)
         |              |  |  |        |  +--ro te-node-id?
   uint32
         |              |  |  |        +--:(label)
         |              |  |  |           +--ro label?
   uint32
         |              |  |  +--ro underlay-backup-path
         |              |  |  |  +--ro topology-id?    leafref
         |              |  |  |  +--ro path-element* [path-element-id]
         |              |  |  |     +--ro path-element-id    uint32
         |              |  |  |     +--ro loose?             boolean
         |              |  |  |     +--ro (element-type)?
         |              |  |  |        +--:(numbered-link)
         |              |  |  |        |  +--ro link-ip-address?
   inet:ip-address
         |              |  |  |        +--:(unnumbered-link)
         |              |  |  |        |  +--ro link-node-id?
   uint32
         |              |  |  |        |  +--ro te-link-id?
   uint32
         |              |  |  |        +--:(node)
         |              |  |  |        |  +--ro te-node-id?
   uint32
         |              |  |  |        +--:(label)
```

```
         |              |  |  |          +--ro label?
   uint32
         |              |  |  +--ro underlay-protection-type?   uint16
         |              |  |  +--ro underlay-trail-src
         |              |  |  |  +--ro node-ref?       leafref
         |              |  |  |  +--ro link-end-ref?   leafref
         |              |  |  +--ro underlay-trail-des
         |              |  |  |  +--ro node-ref?       leafref
         |              |  |  |  +--ro link-end-ref?   leafref
         |              |  |  +--ro dynamic?                    boolean
         |              |  |  +--ro committed?                  boolean
         |              |  +--ro ted
         |              |     +--ro admin-status?
   enumeration
         |              |     +--ro oper-status?
   enumeration
         |              |     +--ro area-id?
   binary
         |              |     +--ro performance-metric-throttle {te-
   performance-metric}?
         |              |     |  +--ro unidirectional-delay-offset?
   uint32
         |              |     |  +--ro measure-interval?
   uint32
         |              |     |  +--ro advertisement-interval?
   uint32
         |              |     |  +--ro suppression-interval?
   uint32
         |              |     |  +--ro threshold-out
         |              |     |  |  +--ro unidirectional-delay?
   uint32
         |              |     |  |  +--ro unidirectional-min-delay?
   uint32
         |              |     |  |  +--ro unidirectional-max-delay?
   uint32
         |              |     |  |  +--ro unidirectional-delay-
   variation?       uint32
         |              |     |  |  +--ro unidirectional-packet-loss?
   decimal64
         |              |     |  |  +--ro unidirectional-residual-
   bandwidth?    decimal64
         |              |     |  |  +--ro unidirectional-available-
   bandwidth?   decimal64
         |              |     |  |  +--ro unidirectional-utilized-
   bandwidth?    decimal64
         |              |     |  +--ro threshold-in
```

```
         |              |     |  |  +--ro unidirectional-delay?
   uint32
         |              |     |  |  +--ro unidirectional-min-delay?
   uint32
         |              |     |  |  +--ro unidirectional-max-delay?
   uint32
         |              |     |  |  +--ro unidirectional-delay-
   variation?       uint32
         |              |     |  |  +--ro unidirectional-packet-loss?
   decimal64
         |              |     |  |  +--ro unidirectional-residual-
   bandwidth?    decimal64
         |              |     |  |  +--ro unidirectional-available-
   bandwidth?   decimal64
         |              |     |  |  +--ro unidirectional-utilized-
   bandwidth?    decimal64
         |              |     |  +--ro threshold-accelerated-
   advertisement
         |              |     |     +--ro unidirectional-delay?
   uint32
         |              |     |     +--ro unidirectional-min-delay?
   uint32
         |              |     |     +--ro unidirectional-max-delay?
   uint32
         |              |     |     +--ro unidirectional-delay-
   variation?       uint32
         |              |     |     +--ro unidirectional-packet-loss?
   decimal64
         |              |     |     +--ro unidirectional-residual-
   bandwidth?    decimal64
         |              |     |     +--ro unidirectional-available-
   bandwidth?   decimal64
         |              |     |     +--ro unidirectional-utilized-
   bandwidth?    decimal64
         |              |     +--ro information-source?
   enumeration
         |              |     +--ro credibility-preference?
   uint16
         |              |     +--ro link-index?
   uint64
         |              |     +--ro administrative-group* [sequence]
         |              |     |  +--ro sequence      uint32
         |              |     |  +--ro ag-element?   uint32
         |              |     +--ro max-link-bandwidth?
   decimal64
```

```
         |              |     +--ro max-resv-link-bandwidth?
   decimal64
         |              |     +--ro unreserved-bandwidth* [priority]
         |              |     |  +--ro priority     uint8
         |              |     |  +--ro bandwidth?   decimal64
         |              |     +--ro te-default-metric?
   uint32
         |              |     +--ro performance-metric {te-performance-
   metric}?
         |              |     |  +--ro measurement
         |              |     |  |  +--ro unidirectional-delay?
   uint32
         |              |     |  |  +--ro unidirectional-min-delay?
   uint32
         |              |     |  |  +--ro unidirectional-max-delay?
   uint32
         |              |     |  |  +--ro unidirectional-delay-
   variation?       uint32
         |              |     |  |  +--ro unidirectional-packet-loss?
   decimal64
         |              |     |  |  +--ro unidirectional-residual-
   bandwidth?    decimal64
         |              |     |  |  +--ro unidirectional-available-
   bandwidth?   decimal64
         |              |     |  |  +--ro unidirectional-utilized-
   bandwidth?    decimal64
         |              |     |  +--ro normality
         |              |     |     +--ro unidirectional-delay?
   performance-metric-normality
         |              |     |     +--ro unidirectional-min-delay?
   performance-metric-normality
         |              |     |     +--ro unidirectional-max-delay?
   performance-metric-normality
         |              |     |     +--ro unidirectional-delay-
   variation?       performance-metric-normality
         |              |     |     +--ro unidirectional-packet-loss?
   performance-metric-normality
         |              |     |     +--ro unidirectional-residual-
   bandwidth?    performance-metric-normality
         |              |     |     +--ro unidirectional-available-
   bandwidth?   performance-metric-normality
         |              |     |     +--ro unidirectional-utilized-
   bandwidth?    performance-metric-normality
         |              |     +--ro link-protection-type?
   enumeration
```

```
         |              |     +--ro interface-switching-capabilities*
   [switching-capability]
         |              |     |  +--ro switching-capability
   ted:switching-capabilities
         |              |     |  +--ro encoding?
   ted:encoding-type
         |              |     |  +--ro max-lsp-bandwidth* [priority]
         |              |     |  |  +--ro priority     uint8
         |              |     |  |  +--ro bandwidth?   decimal64
         |              |     |  +--ro packet-switch-capable
         |              |     |  |  +--ro minimum-lsp-bandwidth?
   decimal64
         |              |     |  |  +--ro interface-mtu?
   uint16
         |              |     |  +--ro time-division-multiplex-capable
         |              |     |     +--ro minimum-lsp-bandwidth?
   decimal64
         |              |     |     +--ro indication?
   enumeration
         |              |     +--ro srlg
         |              |     |  +--ro srlg-values* [srlg-value]
         |              |     |     +--ro srlg-value    uint32
         |              |     +--ro alt-information-sources*
   [information-source]
         |              |        +--ro information-source
   enumeration
         |              |        +--ro credibility-preference?
   uint16
         |              |        +--ro link-index?
   uint64
         |              |        +--ro administrative-group* [sequence]
         |              |        |  +--ro sequence      uint32
         |              |        |  +--ro ag-element?   uint32
         |              |        +--ro max-link-bandwidth?
   decimal64
         |              |        +--ro max-resv-link-bandwidth?
   decimal64
         |              |        +--ro unreserved-bandwidth* [priority]
         |              |        |  +--ro priority     uint8
         |              |        |  +--ro bandwidth?   decimal64
         |              |        +--ro te-default-metric?
   uint32
         |              |        +--ro performance-metric {te-
   performance-metric}?
         |              |        |  +--ro measurement
```

```
         |              |        |  |  +--ro unidirectional-delay?
   uint32
         |              |        |  |  +--ro unidirectional-min-delay?
   uint32
         |              |        |  |  +--ro unidirectional-max-delay?
   uint32
         |              |        |  |  +--ro unidirectional-delay-
   variation?       uint32
         |              |        |  |  +--ro unidirectional-packet-
   loss?           decimal64
         |              |        |  |  +--ro unidirectional-residual-
   bandwidth?    decimal64
         |              |        |  |  +--ro unidirectional-available-
   bandwidth?   decimal64
         |              |        |  |  +--ro unidirectional-utilized-
   bandwidth?    decimal64
         |              |        |  +--ro normality
         |              |        |     +--ro unidirectional-delay?
   performance-metric-normality
         |              |        |     +--ro unidirectional-min-delay?
   performance-metric-normality
         |              |        |     +--ro unidirectional-max-delay?
   performance-metric-normality
         |              |        |     +--ro unidirectional-delay-
   variation?       performance-metric-normality
         |              |        |     +--ro unidirectional-packet-
   loss?           performance-metric-normality
         |              |        |     +--ro unidirectional-residual-
   bandwidth?    performance-metric-normality
         |              |        |     +--ro unidirectional-available-
   bandwidth?   performance-metric-normality
         |              |        |     +--ro unidirectional-utilized-
   bandwidth?    performance-metric-normality
         |              |        +--ro link-protection-type?
   enumeration
         |              |        +--ro interface-switching-
   capabilities* [switching-capability]
         |              |        |  +--ro switching-capability
   ted:switching-capabilities
         |              |        |  +--ro encoding?
   ted:encoding-type
         |              |        |  +--ro max-lsp-bandwidth* [priority]
         |              |        |  |  +--ro priority     uint8
         |              |        |  |  +--ro bandwidth?   decimal64
         |              |        |  +--ro packet-switch-capable
```

```
         |              |        |  |  +--ro minimum-lsp-bandwidth?
   decimal64
         |              |        |  |  +--ro interface-mtu?
   uint16
         |              |        |  +--ro time-division-multiplex-
   capable
         |              |        |     +--ro minimum-lsp-bandwidth?
   decimal64
         |              |        |     +--ro indication?
   enumeration
         |              |        +--ro srlg
         |              |           +--ro srlg-values* [srlg-value]
         |              |              +--ro srlg-value    uint32
         |              +--ro te-link-state-attributes
         |                 +--ro information-source?       enumeration
         |                 +--ro credibility-preference?   uint16
         +--ro node* [te-node-uu-id]
         |  +--ro te-node-uu-id               yang:uuid
         |  +--ro te-node-id?                 te-node-id
         |  +--ro te-node-attributes
         |  |  +--ro schedules* [schedule-id]
         |  |  |  +--ro schedule-id          uint32
         |  |  |  +--ro start?               yang:date-and-time
         |  |  |  +--ro schedule-duration?   string
         |  |  |  +--ro repeat-interval?     string
         |  |  +--ro name?                  inet:domain-name
         |  |  +--ro signaling-address*     inet:ip-address
         |  |  +--ro flag*                  flag-type
         |  |  +--ro is-abstract?           boolean
         |  |  +--ro underlay-topology?     leafref {te-topology-
   hierarchy}?
         |  |  +--ro te-link* [te-link-id]
         |  |  |  +--ro te-link-id         te-link-id
         |  |  |  +--ro (stack-level)?
         |  |  |     +--:(bundle)
         |  |  |     |  +--ro bundled-links
         |  |  |     |     +--ro bundled-link* [sequence]
         |  |  |     |        +--ro sequence       uint32
         |  |  |     |        +--ro te-link-ref?   leafref
         |  |  |     +--:(component)
         |  |  |        +--ro component-links
         |  |  |           +--ro component-link* [sequence]
         |  |  |              +--ro sequence              uint32
         |  |  |              +--ro component-link-ref?   leafref
         |  |  +--ro connectivity-matrix* [id]
         |  |  |  +--ro id            uint32
```

```
         |  |  |  +--ro from-link
         |  |  |  |  +--ro node-ref?       leafref
         |  |  |  |  +--ro link-end-ref?   leafref
         |  |  |  +--ro to-link
         |  |  |  |  +--ro node-ref?       leafref
         |  |  |  |  +--ro link-end-ref?   leafref
         |  |  |  +--ro is-allowed?   boolean
         |  |  +--ro ted
         |  |     +--ro te-router-id-ipv4?    inet:ipv4-address
         |  |     +--ro te-router-id-ipv6?    inet:ipv6-address
         |  |     +--ro ipv4-local-address* [ipv4-prefix]
         |  |     |  +--ro ipv4-prefix    inet:ipv4-prefix
         |  |     +--ro ipv6-local-address* [ipv6-prefix]
         |  |     |  +--ro ipv6-prefix      inet:ipv6-prefix
         |  |     |  +--ro prefix-option?   uint8
         |  |     +--ro pcc-capabilities?     pcc-capabilities
         |  +--ro te-node-state-attributes
         |     +--ro information-source?       enumeration
         |     +--ro credibility-preference?   uint16
         +--ro link* [source-te-node-id source-te-link-id dest-te-node-
   id dest-te-link-id]
            +--ro source-te-node-id           leafref
            +--ro source-te-link-id           leafref
            +--ro dest-te-node-id             leafref
            +--ro dest-te-link-id             leafref
            +--ro te-link-attributes
            |  +--ro schedules* [schedule-id]
            |  |  +--ro schedule-id          uint32
            |  |  +--ro start?               yang:date-and-time
            |  |  +--ro schedule-duration?   string
            |  |  +--ro repeat-interval?     string
            |  +--ro name?          string
            |  +--ro flag*          flag-type
            |  +--ro is-abstract?   boolean
            |  +--ro underlay! {te-topology-hierarchy}?
            |  |  +--ro underlay-path
            |  |  |  +--ro topology-id?    leafref
            |  |  |  +--ro path-element* [path-element-id]
            |  |  |     +--ro path-element-id    uint32
            |  |  |     +--ro loose?             boolean
            |  |  |     +--ro (element-type)?
            |  |  |        +--:(numbered-link)
            |  |  |        |  +--ro link-ip-address?   inet:ip-address
            |  |  |        +--:(unnumbered-link)
            |  |  |        |  +--ro link-node-id?      uint32
            |  |  |        |  +--ro te-link-id?        uint32
```

```
            |  |  |        +--:(node)
            |  |  |        |  +--ro te-node-id?        uint32
            |  |  |        +--:(label)
            |  |  |           +--ro label?             uint32
            |  |  +--ro underlay-backup-path
            |  |  |  +--ro topology-id?    leafref
            |  |  |  +--ro path-element* [path-element-id]
            |  |  |     +--ro path-element-id    uint32
            |  |  |     +--ro loose?             boolean
            |  |  |     +--ro (element-type)?
            |  |  |        +--:(numbered-link)
            |  |  |        |  +--ro link-ip-address?   inet:ip-address
            |  |  |        +--:(unnumbered-link)
            |  |  |        |  +--ro link-node-id?      uint32
            |  |  |        |  +--ro te-link-id?        uint32
            |  |  |        +--:(node)
            |  |  |        |  +--ro te-node-id?        uint32
            |  |  |        +--:(label)
            |  |  |           +--ro label?             uint32
            |  |  +--ro underlay-protection-type?   uint16
            |  |  +--ro underlay-trail-src
            |  |  |  +--ro node-ref?       leafref
            |  |  |  +--ro link-end-ref?   leafref
            |  |  +--ro underlay-trail-des
            |  |     +--ro node-ref?       leafref
            |  |     +--ro link-end-ref?   leafref
            |  +--ro ted
            |     +--ro admin-status?                       enumeration
            |     +--ro oper-status?                        enumeration
            |     +--ro area-id?                            binary
            |     +--ro performance-metric-throttle {te-performance-
   metric}?
            |     |  +--ro unidirectional-delay-offset?
   uint32
            |     |  +--ro measure-interval?
   uint32
            |     |  +--ro advertisement-interval?
   uint32
            |     |  +--ro suppression-interval?
   uint32
            |     |  +--ro threshold-out
            |     |  |  +--ro unidirectional-delay?
   uint32
            |     |  |  +--ro unidirectional-min-delay?
   uint32
```

```
             |     |  |  +--ro unidirectional-max-delay?
   uint32
             |     |  |  +--ro unidirectional-delay-variation?
   uint32
             |     |  |  +--ro unidirectional-packet-loss?
   decimal64
             |     |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
             |     |  |  +--ro unidirectional-available-bandwidth?
   decimal64
             |     |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
             |     |  +--ro threshold-in
             |     |  |  +--ro unidirectional-delay?
   uint32
             |     |  |  +--ro unidirectional-min-delay?
   uint32
             |     |  |  +--ro unidirectional-max-delay?
   uint32
             |     |  |  +--ro unidirectional-delay-variation?
   uint32
             |     |  |  +--ro unidirectional-packet-loss?
   decimal64
             |     |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
             |     |  |  +--ro unidirectional-available-bandwidth?
   decimal64
             |     |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
             |     |  +--ro threshold-accelerated-advertisement
             |     |     +--ro unidirectional-delay?
   uint32
             |     |     +--ro unidirectional-min-delay?
   uint32
             |     |     +--ro unidirectional-max-delay?
   uint32
             |     |     +--ro unidirectional-delay-variation?
   uint32
             |     |     +--ro unidirectional-packet-loss?
   decimal64
             |     |     +--ro unidirectional-residual-bandwidth?
   decimal64
             |     |     +--ro unidirectional-available-bandwidth?
   decimal64
             |     |     +--ro unidirectional-utilized-bandwidth?
   decimal64
```

```
            |     +--ro information-source?                 enumeration
            |     +--ro credibility-preference?             uint16
            |     +--ro link-index?                         uint64
            |     +--ro administrative-group* [sequence]
            |     |  +--ro sequence      uint32
            |     |  +--ro ag-element?   uint32
            |     +--ro max-link-bandwidth?                 decimal64
            |     +--ro max-resv-link-bandwidth?            decimal64
            |     +--ro unreserved-bandwidth* [priority]
            |     |  +--ro priority     uint8
            |     |  +--ro bandwidth?   decimal64
            |     +--ro te-default-metric?                  uint32
            |     +--ro performance-metric {te-performance-metric}?
            |     |  +--ro measurement
            |     |  |  +--ro unidirectional-delay?
   uint32
            |     |  |  +--ro unidirectional-min-delay?
   uint32
            |     |  |  +--ro unidirectional-max-delay?
   uint32
            |     |  |  +--ro unidirectional-delay-variation?
   uint32
            |     |  |  +--ro unidirectional-packet-loss?
   decimal64
            |     |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
            |     |  |  +--ro unidirectional-available-bandwidth?
   decimal64
            |     |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
            |     |  +--ro normality
            |     |     +--ro unidirectional-delay?
   performance-metric-normality
            |     |     +--ro unidirectional-min-delay?
   performance-metric-normality
            |     |     +--ro unidirectional-max-delay?
   performance-metric-normality
            |     |     +--ro unidirectional-delay-variation?
   performance-metric-normality
            |     |     +--ro unidirectional-packet-loss?
   performance-metric-normality
            |     |     +--ro unidirectional-residual-bandwidth?
   performance-metric-normality
            |     |     +--ro unidirectional-available-bandwidth?
   performance-metric-normality
```

```
             |     |     +--ro unidirectional-utilized-bandwidth?
   performance-metric-normality
             |     +--ro link-protection-type?               enumeration
             |     +--ro interface-switching-capabilities* [switching-
   capability]
             |     |  +--ro switching-capability
   ted:switching-capabilities
             |     |  +--ro encoding?
   ted:encoding-type
             |     |  +--ro max-lsp-bandwidth* [priority]
             |     |  |  +--ro priority     uint8
             |     |  |  +--ro bandwidth?   decimal64
             |     |  +--ro packet-switch-capable
             |     |  |  +--ro minimum-lsp-bandwidth?   decimal64
             |     |  |  +--ro interface-mtu?           uint16
             |     |  +--ro time-division-multiplex-capable
             |     |     +--ro minimum-lsp-bandwidth?   decimal64
             |     |     +--ro indication?              enumeration
             |     +--ro srlg
             |     |  +--ro srlg-values* [srlg-value]
             |     |     +--ro srlg-value    uint32
             |     +--ro alt-information-sources* [information-source]
             |        +--ro information-source
   enumeration
             |        +--ro credibility-preference?             uint16
             |        +--ro link-index?                         uint64
             |        +--ro administrative-group* [sequence]
             |        |  +--ro sequence      uint32
             |        |  +--ro ag-element?   uint32
             |        +--ro max-link-bandwidth?
   decimal64
             |        +--ro max-resv-link-bandwidth?
   decimal64
             |        +--ro unreserved-bandwidth* [priority]
             |        |  +--ro priority     uint8
             |        |  +--ro bandwidth?   decimal64
             |        +--ro te-default-metric?                  uint32
             |        +--ro performance-metric {te-performance-metric}?
             |        |  +--ro measurement
             |        |  |  +--ro unidirectional-delay?
   uint32
             |        |  |  +--ro unidirectional-min-delay?
   uint32
             |        |  |  +--ro unidirectional-max-delay?
   uint32
```

```
             |        |  |  +--ro unidirectional-delay-variation?
   uint32
             |        |  |  +--ro unidirectional-packet-loss?
   decimal64
             |        |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
             |        |  |  +--ro unidirectional-available-bandwidth?
   decimal64
             |        |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
             |        |  +--ro normality
             |        |     +--ro unidirectional-delay?
   performance-metric-normality
             |        |     +--ro unidirectional-min-delay?
   performance-metric-normality
             |        |     +--ro unidirectional-max-delay?
   performance-metric-normality
             |        |     +--ro unidirectional-delay-variation?
   performance-metric-normality
             |        |     +--ro unidirectional-packet-loss?
   performance-metric-normality
             |        |     +--ro unidirectional-residual-bandwidth?
   performance-metric-normality
             |        |     +--ro unidirectional-available-bandwidth?
   performance-metric-normality
             |        |     +--ro unidirectional-utilized-bandwidth?
   performance-metric-normality
             |        +--ro link-protection-type?
   enumeration
             |        +--ro interface-switching-capabilities*
   [switching-capability]
             |        |  +--ro switching-capability
   ted:switching-capabilities
             |        |  +--ro encoding?
   ted:encoding-type
             |        |  +--ro max-lsp-bandwidth* [priority]
             |        |  |  +--ro priority     uint8
             |        |  |  +--ro bandwidth?   decimal64
             |        |  +--ro packet-switch-capable
             |        |  |  +--ro minimum-lsp-bandwidth?   decimal64
             |        |  |  +--ro interface-mtu?           uint16
             |        |  +--ro time-division-multiplex-capable
             |        |     +--ro minimum-lsp-bandwidth?   decimal64
             |        |     +--ro indication?              enumeration
             |        +--ro srlg
             |           +--ro srlg-values* [srlg-value]
```

```
           |              +--ro srlg-value    uint32
           +--ro te-link-state-attributes
              +--ro information-source?       enumeration
              +--ro credibility-preference?   uint16
   notifications:
      +---n te-node-event
      |  +--ro event-type?           te-topology-event-type
      |  +--ro node-ref?             leafref
      |  +--ro te-topology!
      |  +--ro te-node-attributes
      |     +--ro schedules* [schedule-id]
      |     |  +--ro schedule-id          uint32
      |     |  +--ro start?               yang:date-and-time
      |     |  +--ro schedule-duration?   string
      |     |  +--ro repeat-interval?     string
      |     +--ro name?                  inet:domain-name
      |     +--ro signaling-address*     inet:ip-address
      |     +--ro flag*                  flag-type
      |     +--ro is-abstract?           boolean
      |     +--ro underlay-topology?     leafref {te-topology-
   hierarchy}?
      |     +--ro te-link* [te-link-id]
      |     |  +--ro te-link-id         te-link-id
      |     |  +--ro (stack-level)?
      |     |     +--:(bundle)
      |     |     |  +--ro bundled-links
      |     |     |     +--ro bundled-link* [sequence]
      |     |     |        +--ro sequence       uint32
      |     |     |        +--ro te-link-ref?   leafref
      |     |     +--:(component)
      |     |        +--ro component-links
      |     |           +--ro component-link* [sequence]
      |     |              +--ro sequence              uint32
      |     |              +--ro component-link-ref?   leafref
      |     +--ro connectivity-matrix* [id]
      |     |  +--ro id            uint32
      |     |  +--ro from-link
      |     |  |  +--ro node-ref?       leafref
      |     |  |  +--ro link-end-ref?   leafref
      |     |  +--ro to-link
      |     |  |  +--ro node-ref?       leafref
      |     |  |  +--ro link-end-ref?   leafref
      |     |  +--ro is-allowed?   boolean
      |     +--ro ted
      |        +--ro te-router-id-ipv4?    inet:ipv4-address
      |        +--ro te-router-id-ipv6?    inet:ipv6-address
```

```
      |        +--ro ipv4-local-address* [ipv4-prefix]
      |        |  +--ro ipv4-prefix    inet:ipv4-prefix
      |        +--ro ipv6-local-address* [ipv6-prefix]
      |        |  +--ro ipv6-prefix      inet:ipv6-prefix
      |        |  +--ro prefix-option?   uint8
      |        +--ro pcc-capabilities?     pcc-capabilities
      +---n te-link-event
         +--ro event-type?              te-topology-event-type
         +--ro source-te-node-id-ref?   leafref
         +--ro source-te-link-id-ref?   leafref
         +--ro dest-te-node-id-ref?     leafref
         +--ro dest-te-link-id-ref?     leafref
         +--ro te-topology!
         +--ro te-link-attributes
            +--ro schedules* [schedule-id]
            |  +--ro schedule-id          uint32
            |  +--ro start?               yang:date-and-time
            |  +--ro schedule-duration?   string
            |  +--ro repeat-interval?     string
            +--ro name?          string
            +--ro flag*          flag-type
            +--ro is-abstract?   boolean
            +--ro underlay! {te-topology-hierarchy}?
            |  +--ro underlay-path
            |  |  +--ro topology-id?    leafref
            |  |  +--ro path-element* [path-element-id]
            |  |     +--ro path-element-id    uint32
            |  |     +--ro loose?             boolean
            |  |     +--ro (element-type)?
            |  |        +--:(numbered-link)
            |  |        |  +--ro link-ip-address?   inet:ip-address
            |  |        +--:(unnumbered-link)
            |  |        |  +--ro link-node-id?      uint32
            |  |        |  +--ro te-link-id?        uint32
            |  |        +--:(node)
            |  |        |  +--ro te-node-id?        uint32
            |  |        +--:(label)
            |  |           +--ro label?             uint32
            |  +--ro underlay-backup-path
            |  |  +--ro topology-id?    leafref
            |  |  +--ro path-element* [path-element-id]
            |  |     +--ro path-element-id    uint32
            |  |     +--ro loose?             boolean
            |  |     +--ro (element-type)?
            |  |        +--:(numbered-link)
            |  |        |  +--ro link-ip-address?   inet:ip-address
```

```
           |  |        +--:(unnumbered-link)
           |  |        |  +--ro link-node-id?      uint32
           |  |        |  +--ro te-link-id?        uint32
           |  |        +--:(node)
           |  |        |  +--ro te-node-id?        uint32
           |  |        +--:(label)
           |  |           +--ro label?            uint32
           |  +--ro underlay-protection-type?   uint16
           |  +--ro underlay-trail-src
           |  |  +--ro node-ref?       leafref
           |  |  +--ro link-end-ref?   leafref
           |  +--ro underlay-trail-des
           |  |  +--ro node-ref?       leafref
           |  |  +--ro link-end-ref?   leafref
           |  +--ro dynamic?                    boolean
           |  +--ro committed?                  boolean
           +--ro ted
              +--ro admin-status?                      enumeration
              +--ro oper-status?                       enumeration
              +--ro area-id?                           binary
              +--ro performance-metric-throttle {te-performance-
   metric}?
              |  +--ro unidirectional-delay-offset?           uint32
              |  +--ro measure-interval?                      uint32
              |  +--ro advertisement-interval?                uint32
              |  +--ro suppression-interval?                  uint32
              |  +--ro threshold-out
              |  |  +--ro unidirectional-delay?                uint32
              |  |  +--ro unidirectional-min-delay?            uint32
              |  |  +--ro unidirectional-max-delay?            uint32
              |  |  +--ro unidirectional-delay-variation?      uint32
              |  |  +--ro unidirectional-packet-loss?
   decimal64
              |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
              |  |  +--ro unidirectional-available-bandwidth?
   decimal64
              |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
              |  +--ro threshold-in
              |  |  +--ro unidirectional-delay?                uint32
              |  |  +--ro unidirectional-min-delay?            uint32
              |  |  +--ro unidirectional-max-delay?            uint32
              |  |  +--ro unidirectional-delay-variation?      uint32
              |  |  +--ro unidirectional-packet-loss?
   decimal64
```

```
               |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
               |  |  +--ro unidirectional-available-bandwidth?
   decimal64
               |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
               |  +--ro threshold-accelerated-advertisement
               |     +--ro unidirectional-delay?                uint32
               |     +--ro unidirectional-min-delay?            uint32
               |     +--ro unidirectional-max-delay?            uint32
               |     +--ro unidirectional-delay-variation?      uint32
               |     +--ro unidirectional-packet-loss?
   decimal64
               |     +--ro unidirectional-residual-bandwidth?
   decimal64
               |     +--ro unidirectional-available-bandwidth?
   decimal64
               |     +--ro unidirectional-utilized-bandwidth?
   decimal64
               +--ro information-source?                enumeration
               +--ro credibility-preference?            uint16
               +--ro link-index?                        uint64
               +--ro administrative-group* [sequence]
               |  +--ro sequence      uint32
               |  +--ro ag-element?   uint32
               +--ro max-link-bandwidth?                decimal64
               +--ro max-resv-link-bandwidth?           decimal64
               +--ro unreserved-bandwidth* [priority]
               |  +--ro priority     uint8
               |  +--ro bandwidth?   decimal64
               +--ro te-default-metric?                 uint32
               +--ro performance-metric {te-performance-metric}?
               |  +--ro measurement
               |  |  +--ro unidirectional-delay?                uint32
               |  |  +--ro unidirectional-min-delay?            uint32
               |  |  +--ro unidirectional-max-delay?            uint32
               |  |  +--ro unidirectional-delay-variation?      uint32
               |  |  +--ro unidirectional-packet-loss?
   decimal64
               |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
               |  |  +--ro unidirectional-available-bandwidth?
   decimal64
               |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
               |  +--ro normality
```

```
               |     +--ro unidirectional-delay?
   performance-metric-normality
               |     +--ro unidirectional-min-delay?
   performance-metric-normality
               |     +--ro unidirectional-max-delay?
   performance-metric-normality
               |     +--ro unidirectional-delay-variation?
   performance-metric-normality
               |     +--ro unidirectional-packet-loss?
   performance-metric-normality
               |     +--ro unidirectional-residual-bandwidth?
   performance-metric-normality
               |     +--ro unidirectional-available-bandwidth?
   performance-metric-normality
               |     +--ro unidirectional-utilized-bandwidth?
   performance-metric-normality
               +--ro link-protection-type?               enumeration
               +--ro interface-switching-capabilities* [switching-
   capability]
               |  +--ro switching-capability
   ted:switching-capabilities
               |  +--ro encoding?
   ted:encoding-type
               |  +--ro max-lsp-bandwidth* [priority]
               |  |  +--ro priority     uint8
               |  |  +--ro bandwidth?   decimal64
               |  +--ro packet-switch-capable
               |  |  +--ro minimum-lsp-bandwidth?   decimal64
               |  |  +--ro interface-mtu?           uint16
               |  +--ro time-division-multiplex-capable
               |     +--ro minimum-lsp-bandwidth?   decimal64
               |     +--ro indication?              enumeration
               +--ro srlg
               |  +--ro srlg-values* [srlg-value]
               |     +--ro srlg-value    uint32
               +--ro alt-information-sources* [information-source]
                  +--ro information-source                  enumeration
                  +--ro credibility-preference?             uint16
                  +--ro link-index?                         uint64
                  +--ro administrative-group* [sequence]
                  |  +--ro sequence      uint32
                  |  +--ro ag-element?   uint32
                  +--ro max-link-bandwidth?                 decimal64
                  +--ro max-resv-link-bandwidth?            decimal64
                  +--ro unreserved-bandwidth* [priority]
                  |  +--ro priority     uint8
```

```
                  |  +--ro bandwidth?   decimal64
                  +--ro te-default-metric?                  uint32
                  +--ro performance-metric {te-performance-metric}?
                  |  +--ro measurement
                  |  |  +--ro unidirectional-delay?
   uint32
                  |  |  +--ro unidirectional-min-delay?
   uint32
                  |  |  +--ro unidirectional-max-delay?
   uint32
                  |  |  +--ro unidirectional-delay-variation?
   uint32
                  |  |  +--ro unidirectional-packet-loss?
   decimal64
                  |  |  +--ro unidirectional-residual-bandwidth?
   decimal64
                  |  |  +--ro unidirectional-available-bandwidth?
   decimal64
                  |  |  +--ro unidirectional-utilized-bandwidth?
   decimal64
                  |  +--ro normality
                  |     +--ro unidirectional-delay?
   performance-metric-normality
                  |     +--ro unidirectional-min-delay?
   performance-metric-normality
                  |     +--ro unidirectional-max-delay?
   performance-metric-normality
                  |     +--ro unidirectional-delay-variation?
   performance-metric-normality
                  |     +--ro unidirectional-packet-loss?
   performance-metric-normality
                  |     +--ro unidirectional-residual-bandwidth?
   performance-metric-normality
                  |     +--ro unidirectional-available-bandwidth?
   performance-metric-normality
                  |     +--ro unidirectional-utilized-bandwidth?
   performance-metric-normality
                  +--ro link-protection-type?               enumeration
                  +--ro interface-switching-capabilities* [switching-
   capability]
                  |  +--ro switching-capability
   ted:switching-capabilities
                  |  +--ro encoding?
   ted:encoding-type
                  |  +--ro max-lsp-bandwidth* [priority]
                  |  |  +--ro priority     uint8
```

```
                  |  |  +--ro bandwidth?   decimal64
                  |  +--ro packet-switch-capable
                  |  |  +--ro minimum-lsp-bandwidth?   decimal64
                  |  |  +--ro interface-mtu?           uint16
                  |  +--ro time-division-multiplex-capable
                  |     +--ro minimum-lsp-bandwidth?   decimal64
                  |     +--ro indication?              enumeration
                  +--ro srlg
                     +--ro srlg-values* [srlg-value]
                        +--ro srlg-value    uint32
```

4. TE Topology Yang Module

```
<CODE BEGINS>


module ietf-te-topology {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology";
  // replace with IANA namespace when assigned

  prefix "tet";

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

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

  import ted {
    prefix "ted";
  }

  import ietf-interfaces {
    prefix "if";
  }

  organization "TBD";
  contact "TBD";
  description "TE topology model";

  revision "2015-03-02" {
    description "Initial revision";
    reference "TBD";
```

```
     }

     /*
      * Features
      */

     feature te-topology-hierarchy {
       description
         "This feature indicates that the system allows underlay
          and/or overlay TE topology hierarchy.";
     }

     /*
      * Typedefs
      */

     typedef te-topology-id {
       type string {
         pattern '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
       }
       description
         "An identifier for a topology.";
     }

     typedef te-node-id {
       type inet:ip-address;
       description
         "An identifier for a node in a topology.
          The identifier is represented as an IPv4 or IPv6 address.
          The identifier SHOULD be chosen such that the same node in a
          real network topology will always be identified through the
          same identifier, even if the model is instantiated in
          separate datastores. An implementation MAY choose to capture
          semantics in the identifier, for example to indicate the type
          of node and/or the type of topology that the node is a part
          of.";
     }

     typedef te-link-id {
       type union {
         type uint32;          // Unnumbered
         type inet:ip-address; // IPv4 or IPv6 address
       }
       description
         "An identifier for a TE link on a node.
          The identifier may be opaque.
```

```
          The identifier SHOULD be chosen such that the same TP in a
          real network topology will always be identified through the
          same identifier, even if the model is instantiated in
          separate datastores. An implementation MAY choose to capture
          semantics in the identifier, for example to indicate the type
          of TP and/or the type of node and topology that the TP is a
          part of.";
     }


     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

     /*
      * Identities
      */

     identity flag-identity {
       description "Base type for flags";
     }

     identity undefined-flag {
       base "flag-identity";
       description "Undefined flag";
     }
```

```
typedef flag-type {
  type identityref {
    base "flag-identity";
  }
  description "Type for flags";
}

/*
 * Groupings
 */

grouping topo-ref {
  description
    "Grouping for an absolute reference to a topology instance.";
  leaf topo-ref {
    type leafref {
      path "/tet:te-topologies/tet:topology/tet:te-topology-id";
    }
    description
      "An absolute reference to a topology instance.";
  }
}

grouping link-ref {
  description
    "Grouping for an absolute reference to a link instance.";
  leaf source-te-node-id-ref {
    type leafref {
      path "/tet:te-topologies/tet:link/tet:source-te-node-id";
    }
    description
      "An absolute reference to a link instance.";
  }
  leaf source-te-link-id-ref {
    type leafref {
      path "/tet:te-topologies/tet:link/tet:source-te-link-id";
    }
    description
      "An absolute reference to a link instance.";
  }
  leaf dest-te-node-id-ref {
    type leafref {
      path "/tet:te-topologies/tet:link/tet:dest-te-node-id";
    }
    description
```

```
          "An absolute reference to a link instance.";
      }
      leaf dest-te-link-id-ref {
        type leafref {
          path "/tet:te-topologies/tet:link/tet:dest-te-link-id";
        }
        description
          "An absolute reference to a link instance.";
      }
    }

    grouping node-ref {
      description
        "Grouping for an absolute reference to a node instance.";
      leaf node-ref {
        type leafref {
          path "/tet:te-topologies/tet:node/tet:te-node-uu-id";
        }
        description
          "An absolute reference to a node instance.";
      }
    }

    grouping link-end-ref {
      description
        "Grouping for an absolute reference to a TE link end, which is
        the local representation of a TE link on a node.";
      uses node-ref;
      leaf link-end-ref {
        type leafref {
          path "/tet:te-topologies"
            +"/tet:node[tet:te-node-uu-id = current()/../node-ref]"
            +"/tet:te-node-attributes/tet:te-link/tet:te-link-id";
        }
        description
          "Grouping for an absolute reference to a TE link end.";
      }
    }

    grouping te-topology-type {
      description
        "Identifies the TE topology type.";
      container te-topology {
        presence "indicates TE topology";
        description
          "Its presence identifies the TE topology type.";
```

```
      }
    }

    grouping te-path-element {
      description
        "A group of attributes defining an element in a TE path
        such as TE node, TE link, TE aotomic resource or label.";
      leaf loose {
        type boolean;
        description "true if the element is loose.";
      }
      choice element-type {
        description "Attributes for various element types.";
        case numbered-link {
          leaf link-ip-address {
            type inet:ip-address;
            description "IPv4 or IPv6 address.";
          }
        }
        case unnumbered-link {
          leaf link-node-id {
            type uint32;
            description
              "Node ID of the node where the link end point resides.";
          }
          leaf te-link-id {
            type uint32;
            description "Identifies the link end point.";
          }
        }
        case node {
          leaf te-node-id {
            type uint32;
            description "Identifies the node.";
          }
        }
        case label {
          leaf label {
            type uint32;
            description "Identifies atomic TE resource or label.";
          }
        }
      }
    } // te-path-element

    grouping config-schedule-attributes {
```

```
      description
        "A list of schedules defining when a particular
         configuration takes effect.";
      list schedules {
        key "schedule-id";
        description "A list of schedule elements.";

        leaf schedule-id {
          type uint32;
          description "Identifies the schedule element.";
        }
        leaf start {
          type yang:date-and-time;
          description "Start time.";
        }
        leaf schedule-duration {
          type string {
            pattern
              'P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?(\d+M)?(\d+S)?';
          }
          description "Schedule duration in ISO 8601 format.";
        }
        leaf repeat-interval {
          type string {
            pattern
              'R\d*/P(\d+Y)?(\d+M)?(\d+W)?(\d+D)?T(\d+H)?(\d+M)?'
              + '(\d+S)?';
          }
          description "Repeat interval in ISO 8601 format.";
        }
      }
    }

    grouping information-source-attributes {
      description
        "The attributes identifying source that has provided the
        related information, and the source credibility.";
      leaf information-source {
        type enumeration {
          enum "unknown" {
            description "The source is unknown";
          }
          enum "locally-configured" {
            description "Configured TE link";
          }
          enum "ospfv2" {
```

```
            description "OSPFv2";
          }
          enum "ospfv3" {
            description "OSPFv3";
          }
          enum "isis" {
            description "ISIS";
          }
          enum "other" {
            description "Other source";
          }
        }
        description
          "Indicates the source of the information.";
      }
      leaf credibility-preference {
        type uint16;
        description
          "The preference value to calculate the traffic
          engineering database credibility value used for
          tie-break selection between different
          information-source values.
          Higher value is more preferable.";
      }
    }

    grouping te-node-attributes {
      description "Node attributes in a TE topology.";
      container te-node-attributes {
        description "Node attributes in a TE topology.";
        uses config-schedule-attributes;
        leaf name {
          type inet:domain-name;
          description "Node name.";
        }
        leaf-list signaling-address {
          type inet:ip-address;
          description "Node signaling address.";
        }
        leaf-list flag {
          type flag-type;
          description "Node operational flags.";
        }
        leaf is-abstract {
          type boolean;
          description
```

```
            "true if the node is abstract, false when the node is
            actual.";
        }
        leaf underlay-topology {
          if-feature te-topology-hierarchy;
          type leafref {
            path "/tet:te-topologies/tet:topology/tet:te-topology-id";
          }
          description
            "When an abstract node encapsulates a topology,
             this reference points to said topology.";
        }
        list te-link {
          key "te-link-id";
          description
            "The local representation of a TE link, which
              interconnect TE nodes.";
          leaf te-link-id {
            type te-link-id;
            description
              "TE link identifier.";
          }
          choice stack-level {
            description
              "The TE link can be partitioned into bundled
                links, or component links.";
            case bundle {
              container bundled-links {
                description
                  "A set of bundled links";
                list bundled-link {
                  key "sequence";
                  description
                    "Specify a bundled interfface that is
                      further partitioned.";
                  leaf sequence {
                    type uint32;
                    description
                      "Identify the sequence in the bundle.";
                  }
                  leaf te-link-ref {
                    type leafref {
                      path "../../../te-link-id";
                      require-instance "true";
                    }
                    description
```

```
                    "Reference to TE link on this node.";
                }
              }
            }
          }
          case component {
            container component-links {
              description
                "A set of component links";
              list component-link {
                key "sequence";
                description
                  "Specify a component interfface that is
                   sufficient to unambiguously identify the
                    appropriate resources";

               leaf sequence {
                  type uint32;
                  description
                    "Identify the sequence in the bundle.";
                }
                leaf component-link-ref {
                  type leafref {
                    path "/if:interfaces/if:interface/if:name";
                    require-instance "false";
                  }
                  description
                    "Reference to component link on this
                      node.";
                }
              }
            }
          }
        }
      }
      list connectivity-matrix {
        key "id";
        description
          "Represents node's switching limitations, i.e. limitations
           in interconnecting network TE links across the node.";
        leaf id {
          type uint32;
          description "Identifies the connectivity-matrix entry.";
        }
        container from-link {
          uses tet:link-end-ref;
```

```
            description
              "Reference to source NTP.";
          }
          container to-link {
            uses tet:link-end-ref;
            description
              "Reference to destination NTP.";
          }
          leaf is-allowed {
            type boolean;
            description
              "true  - switching is allowed,
               false - switching is disallowed.";
          }
        }
        container ted {
          description "Includes TE node attributes.";
          uses ted:ted-node-attributes;
        }
      }
    } // te-node-attributes

    grouping te-node-state-attributes {
      description "Node state attributes in a TE topology.";
      container te-node-state-attributes {
        description "Node state attributes in a TE topology.";
        uses information-source-attributes;
      }
    } // te-node-state-attributes

    grouping te-link-underlay-attributes {
      description "Attributes for  te-link underlay.";
      container underlay-path {
        description
          "The service path on the underlay topology that
           supports this link.";
        leaf topology-id {
          type leafref {
            path "/tet:te-topologies/tet:topology/tet:te-topology-id";
            require-instance false;
          }
          description
            "Identifies the topology where the path belongs.";
        }
        list path-element {
          key "path-element-id";
```

```
          description
            "A list of path elements describing the service path";
          leaf path-element-id {
            type uint32;
            description "To identify the element in a path.";
          }
          uses te-path-element;
        }
      } // underlay-path
      container underlay-backup-path {
        description
          "The backup service path on the underlay topology that
           supports this link.";
        leaf topology-id {
          type leafref {
            path "/tet:te-topologies/tet:topology/tet:te-topology-id";
            require-instance false;
          }
          description
            "Identifies the topology where the path belongs.";
        }
        list path-element {
          key "path-element-id";
          description
            "A list of path elements describing the backup service
            path";
          leaf path-element-id {
            type uint32;
            description "To identify the element in a path.";
          }
          uses te-path-element;
        }
      } // underlay-backup-path
      leaf underlay-protection-type {
        type uint16;
        description
          "Underlay protection type desired for this link";
      }
      container underlay-trail-src {
        uses tet:link-end-ref;
        description
          "Source TE link of the underlay trail.";
      }
      container underlay-trail-des {
        uses tet:link-end-ref;
        description
```

```
          "Destination TE link of the underlay trail.";
      }
    } // te-link-underlay-attributes

    grouping te-link-state-underlay-attributes {
      description "State attributes for te-link underlay.";
      leaf dynamic {
        type boolean;
        description
          "true if the underlay is dynamically created.";
      }
      leaf committed {
        type boolean;
        description
          "true if the underlay is committed.";
      }
    } // te-link-state-underlay-attributes

    grouping te-link-attributes {
      description
        "Link attributes in a TE topology.";
      container te-link-attributes {
        description "Link attributes in a TE topology.";
        uses config-schedule-attributes;
        leaf name {
          type string;
          description "Link Name";
        }
        leaf-list flag {
          type flag-type;
          description "Link flags";
        }
        leaf is-abstract {
          type boolean;
          description "true if the link is abstract.";
        }
        container underlay {
          if-feature te-topology-hierarchy;
          presence
            "Indicates the underlay exists for this link.";
          description "State of the underlay of this link.";

          uses te-link-underlay-attributes;
        } // underlay
        container ted {
          description "Includes TE link attributes.";
```

```
          uses ted:ted-link-attributes;
        }
      }
    } // te-link-attributes

    grouping te-link-state-attributes {
      description
        "Link state attributes in a TE topology.";
      container te-link-state-attributes {
        description "Link state attributes in a TE topology.";
        uses information-source-attributes;
      }
    } // te-link-state-attributes

    /*
     * Configuration data nodes
     */

    container te-topologies {
      description
        "This container acts as the top-level data element of
         configuration data.";
      list topology {
        key "te-topology-id";
        description
          "This is the model of an abstract topology. A topology
          contains nodes and links. Each topology MUST be identified
          by a unique te-topology-id for reason that a network could
          contain many topologies.";
        leaf te-topology-id {
          type 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.";
        }
        container topology-types {
          description
            "This container is used to identify the type, or types (as
            a topology can support several types simultaneously), of
            the topology.
            Topology types are the subject of several integrity
            constraints that an implementing server can validate in
            order to maintain integrity of the datastore.
            Topology types are indicated through separate data nodes;
            the set of topology types is expected to increase over
```

```
              time.
              To add support for a new topology, an augmenting module
              needs to augment this container with a new empty optional
              container to indicate the new topology type.
              The use of a container allows to indicate a
              subcategorization of topology types.
              The container SHALL NOT be augmented with any data nodes
              that serve a purpose other than identifying a particular
              topology type.";
            uses te-topology-type; // Defines the TE topology type.
          }
          choice abstract-type {
            description
              "Different structures for abstract or provider
              topologies.";
            case abstract {
              container abstract {
                description
                  "The structure for an abstract topology.";
                list node {
                  key "te-node-id";
                  leaf te-node-uu-id {
                    type leafref {
                      path "../../../../node/te-node-uu-id";
                    }
                    mandatory true;
                    description
                      "The reference to the te-node.";
                  }
                  leaf te-node-id {
                    type leafref {
                      path "../../../../node[te-node-uu-id = "
                        + "current()/../te-node-uu-id]/te-node-id";
                    }
                    description
                      "The identifier of a node in the topology.
                      A node is specific to a topology to which it
                      belongs.";
                  }
                  description
                    "The list of network nodes defined for the
                    topology.";
                }
                list link {
                  key "source-te-node-id source-te-link-id "
                    + "dest-te-node-id dest-te-link-id";
```

```
                leaf source-te-node-id {
                  type leafref {
                    path "../../../../link/source-te-node-id";
                  }
                  description
                    "Source node identifier.";
                }
                leaf source-te-link-id {
                  type leafref {
                    path "../../../../link/source-te-link-id";
                  }
                  description
                    "Source TE link identifier.";
                }
                leaf dest-te-node-id {
                  type leafref {
                     path "../../../../link/dest-te-node-id";
                  }
                  description
                    "Destination node identifier.";
                }
                leaf dest-te-link-id {
                  type leafref {
                    path "../../../../link/dest-te-link-id";
                  }
                  description
                    "Destination TE link identifier.";
                }
                description
                  "TE links in this topology.";
              } // link
            } // abstract
          }
          case provider {
            container provider {
              description
                "The structure for a provider topology.";
              list node {
                key "te-node-id";
                leaf te-node-id {
                  type te-node-id;
                  description
                    "The identifier of a node in the topology.
                    A node is specific to a topology to which it
                    belongs.";
                 }
```

```
                  description
                    "The list of network nodes defined for the
                    topology.";
                  uses te-node-attributes;
                }
                list link {
                  key "source-te-node-id source-te-link-id "
                    + "dest-te-node-id dest-te-link-id";
                  leaf source-te-node-id {
                    type leafref {
                      path "../../node/te-node-id";
                    }
                    mandatory true;
                    description
                      "Source node identifier, must be in same
                      topology.";
                  }
                  leaf source-te-link-id {
                    type leafref {
                      path "../../node[te-node-id = "
                        + "current()/../source-te-node-id]/"
                        + "te-node-attributes/te-link/te-link-id";
                    }
                    mandatory true;
                    description
                      "Source TE link identifier, must be in same
                      topology.";
                  }
                  leaf dest-te-node-id {
                    type leafref {
                      path "../../node/te-node-id";
                    }
                    mandatory true;
                    description
                      "Destination node identifier, must be in the same
                      topology.";
                  }
                  leaf dest-te-link-id {
                    type leafref {
                      path "../../node[te-node-id = "
                        + "current()/../dest-te-node-id]/"
                        + "te-node-attributes/te-link/te-link-id";
                    }
                    mandatory true;
                    description
                      "Destination TE link identifier, must be in same
```

```
                    topology.";
                }
                description
                  "TE link is a logical construct that represents a
                  way to group/map information about certain physical
                  resources (and their properties) that interconnect
                  TE nodes.
                  A Network Link connects a by Local (Source) node and
                  a Remote (Destination) Network Nodes via a set of
                  the nodes' TE links.
                  As it is possible to have several links between the
                  same source and destination nodes, and as a link
                  could potentially be re-homed, to ensure that we
                  would always know to distinguish between
                  links, every link is identified by a dedicated link
                  identifier.
                  Note that a link models a point-to-point link, not a
                  multipoint link.";
                uses te-link-attributes;
              } // link
            } // basic
          }
        } // choice abstract-type
      } // topology

      list node {
        key "te-node-uu-id";
        leaf te-node-uu-id {
          type yang:uuid;
          description
            "The universally unique identifier of a node.";
        }
        leaf te-node-id {
          type te-node-id;
          description
            "The identifier of a node in the topology.
            A node is specific to a topology to which it belongs.";
         }
         description
          "The list of network nodes defined for the topology.";
        uses te-node-attributes;
      } // node

      list link {
        key "source-te-node-id source-te-link-id "
          + "dest-te-node-id dest-te-link-id";
```

```
         leaf source-te-node-id {
           type leafref {
             path "../../node/te-node-uu-id";
           }
           mandatory true;
           description
             "Source node identifier, must be in same topology.";
         }
         leaf source-te-link-id {
           type leafref {
             path "../../node[te-node-uu-id = "
               + "current()/../source-te-node-id]/"
               + "te-node-attributes/te-link/te-link-id";
           }
           mandatory true;
           description
             "Source TE link identifier, must be in same
             topology.";
         }
         leaf dest-te-node-id {
            type leafref {
             path "../../node/te-node-uu-id";
           }
           mandatory true;
           description
                 "Destination node identifier, must be in the same
                 topology.";
         }
         leaf dest-te-link-id {
           type leafref {
             path "../../node[te-node-uu-id = "
               + "current()/../dest-te-node-id]/"
               + "te-node-attributes/te-link/te-link-id";
           }
           mandatory true;
           description
             "Destination TE link identifier, must be in same
             topology.";
         }
         description
           "TE link is a logical construct that represents a way to
           group/map information about certain physical resources
           (and their properties) that interconnect TE nodes.
           A Network Link connects a by Local (Source) node and
           a Remote (Destination) Network Nodes via a set of the
           nodes' TE links.
```

```
          As it is possible to have several links between the same
          source and destination nodes, and as a link could
          potentially be re-homed, to
          ensure that we would always know to distinguish between
          links, every link is identified by a dedicated link
          identifier.
          Note that a link models a point-to-point link, not a
          multipoint link.";
        uses te-link-attributes;
      } // link
    } // te-topologies

    /*
     * Operational state data nodes
     */

    container te-topologies-state {
      config "false";
      description
        "This container acts as the top-level state data element of
         operational data.";
      list topology {
        key "te-topology-id";
        description
          "This is the model of an abstract topology. A topology
          contains nodes and links. Each topology MUST be identified
          by a unique te-topology-id for reason that a network could
          contain many topologies.";
        leaf te-topology-id {
          type 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.";
        }
        leaf server-provided {
          type boolean;
          config false;
          description
            "Indicates whether the topology is client-configurable
             or whether it is provided by the server.  This leaf is
             populated by the server implementing the model.
             It is set to false for topologies that are created by a
             client; it is set to true otherwise.  If it is set to
             true, any attempt to edit the topology MUST be
```

```
            rejected.";
        }
        container topology-types {
          description
            "This container is used to identify the type, or types (as
            a topology can support several types simultaneously), of
            the topology.
            Topology types are the subject of several integrity
            constraints that an implementing server can validate in
            order to maintain integrity of the datastore.
            Topology types are indicated through separate data nodes;
            the set of topology types is expected to increase over
            time.
            To add support for a new topology, an augmenting module
            needs to augment this container with a new empty optional
            container to indicate the new topology type.
            The use of a container allows to indicate a
            subcategorization of topology types.
            The container SHALL NOT be augmented with any data nodes
            that serve a purpose other than identifying a particular
            topology type.";
          uses te-topology-type; // Defines the TE topology type.
        }
        choice abstract-type {
          description
            "Different structures for abstract or provider
            topologies.";
          case abstract {
            container abstract {
              description
                "The structure for an abstract topology.";
              list node {
                key "te-node-id";
                leaf te-node-uu-id {
                  type leafref {
                    path "../../../../node/te-node-uu-id";
                  }
                  mandatory true;
                  description
                    "The reference to the te-node.";
                }
                leaf te-node-id {
                  type leafref {
                    path "../../../../node[te-node-uu-id = "
                      + "current()/../te-node-uu-id]/te-node-id";
                  }
```

```
                      description
                        "The identifier of a node in the topology.
                        A node is specific to a topology to which it
                        belongs.";
                    }
                    description
                      "The list of network nodes defined for the
                      topology.";
                  }
                  list link {
                    key "source-te-node-id source-te-link-id "
                      + "dest-te-node-id dest-te-link-id";
                    leaf source-te-node-id {
                      type leafref {
                        path "../../../../link/source-te-node-id";
                      }
                      description
                        "Source node identifier.";
                    }
                    leaf source-te-link-id {
                      type leafref {
                        path "../../../../link/source-te-link-id";
                      }
                      description
                        "Source TE link identifier.";
                    }
                    leaf dest-te-node-id {
                      type leafref {
                        path "../../../../link/dest-te-node-id";
                      }
                      description
                        "Destination node identifier.";
                    }
                    leaf dest-te-link-id {
                      type leafref {
                        path "../../../../link/dest-te-link-id";
                      }
                      description
                        "Destination TE link identifier.";
                    }
                    description
                      "TE links in this topology.";
                  } // link
                } // abstract
              }
              case provider {
```

```
container provider {
  description
    "The structure for a provider topology.";
  list node {
    key "te-node-id";
    leaf te-node-id {
      type te-node-id;
      description
        "The identifier of a node in the topology.
        A node is specific to a topology to which it
        belongs.";
    }
    description
      "The list of network nodes defined for the
      topology.";
    uses te-node-attributes;
    uses te-node-state-attributes;
  }
  list link {
    key "source-te-node-id source-te-link-id "
      + "dest-te-node-id dest-te-link-id";
    leaf source-te-node-id {
      type leafref {
        path "../../node/te-node-id";
      }
      mandatory true;
      description
        "Source node identifier, must be in same
        topology.";
    }
    leaf source-te-link-id {
      type leafref {
        path "../../node[te-node-id = "
          + "current()/../source-te-node-id]/"
          + "te-node-attributes/te-link/te-link-id";
      }
      mandatory true;
      description
        "Source TE link identifier, must be in same
        topology.";
    }
    leaf dest-te-node-id {
      type leafref {
        path "../../node/te-node-id";
      }
      mandatory true;
```

```
                  description
                    "Destination node identifier, must be in the same
                    topology.";
                }
                leaf dest-te-link-id {
                  type leafref {
                    path "../../node[te-node-id = "
                      + "current()/../dest-te-node-id]/"
                      + "te-node-attributes/te-link/te-link-id";
                  }
                  mandatory true;
                  description
                    "Destination TE link identifier, must be in same
                    topology.";
                }
                description
                  "TE link is a logical construct that represents a
                  way to group/map information about certain physical
                  resources (and their properties) that interconnect
                  TE nodes. A Network Link connects a by Local
                  (Source) node and a Remote (Destination) Network
                  Nodes via a set of the nodes' TE links.
                  As it is possible to have several links between the
                  same source and destination nodes, and as a link
                  could potentially be re-homed, to ensure that we
                  would always know to distinguish between
                  links, every link is identified by a dedicated link
                  identifier.
                  Note that a link models a point-to-point link, not a
                  multipoint link.";
                uses te-link-attributes;
                uses te-link-state-attributes;
              } // link
            } // basic
          }
        } // choice abstract-type
      }

      list node {
        key "te-node-uu-id";
        leaf te-node-uu-id {
          type yang:uuid;
          description
            "The universally unique identifier of a node.";
        }
        leaf te-node-id {
```

```
          type te-node-id;
          description
            "The identifier of a node in the topology.
            A node is specific to a topology to which it belongs.";
        }
        description
          "The list of network nodes defined for the topology.";
        uses te-node-attributes;
        uses te-node-state-attributes;
      } // node

      list link {
        key "source-te-node-id source-te-link-id "
          + "dest-te-node-id dest-te-link-id";
        leaf source-te-node-id {
          type leafref {
            path "../../node/te-node-uu-id";
          }
          mandatory true;
          description
            "Source node identifier, must be in same topology.";
        }
        leaf source-te-link-id {
          type leafref {
            path "../../node[te-node-uu-id = "
              + "current()/../source-te-node-id]/"
              + "te-node-attributes/te-link/te-link-id";
          }
          mandatory true;
          description
            "Source TE link identifier, must be in same
            topology.";
        }
        leaf dest-te-node-id {
          type leafref {
            path "../../node/te-node-uu-id";
          }
          mandatory true;
          description
            "Destination node identifier, must be in the same
            topology.";
        }
        leaf dest-te-link-id {
          type leafref {
            path "../../node[te-node-uu-id = "
              + "current()/../dest-te-node-id]/"
```

```
              + "te-node-attributes/te-link/te-link-id";
          }
          mandatory true;
          description
            "Destination TE link identifier, must be in same
            topology.";
        }
        description
          "TE link is a logical construct that represents a way to
          group/map information about certain physical resources
          (and their properties) that interconnect TE nodes.
          A Network Link connects a by Local (Source) node and
          a Remote (Destination) Network Nodes via a set of the
          nodes' TE links.
          As it is possible to have several links between the same
          source and destination nodes, and as a link could
          potentially be re-homed, to
          ensure that we would always know to distinguish between
          links, every link is identified by a dedicated link
          identifier.
          Note that a link models a point-to-point link, not a
          multipoint link.";
        uses te-link-attributes;
        uses te-link-state-attributes;
      } // link
    } // te-topologies

    augment "/te-topologies-state/topology/abstract-type/provider/"
      + "provider/link/te-link-attributes/underlay" {
      description "Add state attributes to te-link underlay.";
      uses te-link-state-underlay-attributes;
    }

    /*
     * Notifications
     */

    notification te-node-event {
      description "Notification event for TE node";
      leaf event-type {
        type te-topology-event-type;
        description "Event type";
      }
      uses node-ref;
      uses te-topology-type;
      uses tet:te-node-attributes;
```

```
    }

    notification te-link-event {
      description "Notification event for TE link";
      leaf event-type {
        type te-topology-event-type;
        description "Event type";
      }
      uses link-ref;
      uses te-topology-type;
      uses tet:te-link-attributes;
    }

    augment "/te-link-event/te-link-attributes/underlay" {
      description "Add state attributes to te-link underlay.";
      uses te-link-state-underlay-attributes;
    }
  }
  <CODE ENDS>
```

5. Security Considerations

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

6. 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-topology
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-topology
namespace: urn:ietf:params:xml:ns:yang:ietf-te-topology
prefix: tet

7. Normative References

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

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

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

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

[YANG-TED] Liu, X., "A Yang module for TED", <draft-liu-yang-ted>.

[YANG-NET-TOPO] Clemm, A., "A Data Model for Network Topologies", <draft-clemm-i2rs-yang-network-topo>

8. Acknowledgments

The authors would like to thank Lou Berger, Sue Hares, Mazen Khaddam and Zafar Ali for participating in design discussions and providing valuable insights.

Authors' Addresses

Xufeng Liu
Ericsson
Email: xufeng.liu@ericsson.com

Igor Bryskin
ADVA Optical Networking
Email: ibryskin@advaoptical.com

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

Tarek Saad
Cisco Systems Inc

Email: tsaad@cisco.com

Himanshu Shah
Ciena
Email: hshah@ciena.com

Oscar Gonzalez De Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com