TEAS Working Group
Internet Draft
Intended status: Standards Track

Xufeng Liu
Jabil
Igor Bryskin
Huawei Technologies
Vishnu Pavan Beeram
Juniper Networks
Tarek Saad
Cisco Systems Inc
Himanshu Shah
Ciena
Oscar Gonzalez De Dios
Telefonica

Expires: September 13, 2017 March 13, 2017

YANG Data Model for TE Topologies
draft-ietf-teas-yang-te-topo-08

Status of this Memo

Copyright Notice

Abstract

This document defines a YANG data model for representing, retrieving and manipulating TE Topologies. The model serves as a base model that other technology specific TE Topology models can augment.

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

The Traffic Engineering Database (TED) is an essential component of Traffic Engineered (TE) systems that are based on MPLS-TE [RFC2702] and GMPLS [RFC3945]. The TED is a collection of all TE information about all TE nodes and TE links in the network. The TE Topology is a schematic arrangement of TE nodes and TE links present in a given TED. There could be one or more TE Topologies present in a given Traffic Engineered system. The TE Topology is the topology on which path computational algorithms are run to compute Traffic Engineered Paths (TE Paths).

This document defines a YANG [RFC6020] 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. Terminology

TED: The Traffic Engineering Database is a collection of all TE information about all TE nodes and TE links in a given network.

TE-Topology: The TE Topology is a schematic arrangement of TE nodes and TE links in a given TED. It forms the basis for a graph suitable for TE path computations.

Native TE Topology: Native TE Topology is a topology that is native to a given provider network. Native TE topology could be discovered via various routing protocols and/or subscribe/publish techniques. This is the topology on which path computational algorithms are run to compute TE Paths.

Customized TE Topology: Customized TE Topology is a custom topology that is produced by a provider for a given Client. This topology typically augments the Client's Native TE Topology. Path computational algorithms aren't typically run on the Customized TE Topology; they are run on the Client's augmented Native TE Topology.

1.2. Tree Structure - Legend

A simplified graphical representation of the data model is presented in Section 6. 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.
```

1.3. Prefixes in Data Node Names

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

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

2. Characterizing TE Topologies

The data model proposed by this document takes the following characteristics of TE Topologies into account:

- TE Topology is an abstract control-plane representation of the data-plane topology. Hence attributes specific to the data-plane must make their way into the corresponding TE Topology modeling. The TE Topology comprises of dynamic auto-discovered data (data that may change frequently - example: unreserved bandwidth available on data-plane links) as well as fairly static data (data that rarely changes- examples: layer network identification, switching and adaptation capabilities and limitations, fate sharing, administrative colors) associated with data-plane nodes and links. It is possible for a single TE Topology to encompass TE information at multiple switching layers.

- TE Topologies are protocol independent. Information about topological elements may be learnt via link-state protocols, but the topology can exist without being dependent on any particular protocol.

- TE Topology may not be congruent to the routing topology (topology constructed based on routing adjacencies) in a given TE System. There isn't always a one-to-one association between a TE-link and a routing adjacency. For example, the presence of a TE link between a pair of nodes doesn't necessarily imply the existence of a routing-adjacency between these nodes.

- Each TE Topological element has an information source associated with it. In some scenarios, there could be more than one information source associated with each topological element.

- TE Topologies can be hierarchical. Each node and link of a given TE Topology can be associated with respective underlay topology. This means that each node and link of a given TE Topology can be associated with an independent stack of supporting TE Topologies.

- TE Topologies can be customized. TE topologies of a given network presented by the network provider to its client could be customized on per-client request basis. This customization could be performed by provider, by client or by provider/client negotiation. The relationship between a customized topology (as presented to the client) and provider's native topology (as known in its entirety to the provider itself) could be captured as hierarchical (overlay-underlay), but otherwise the two topologies are decoupled from each other.

3. Modeling Abstractions and Transformations

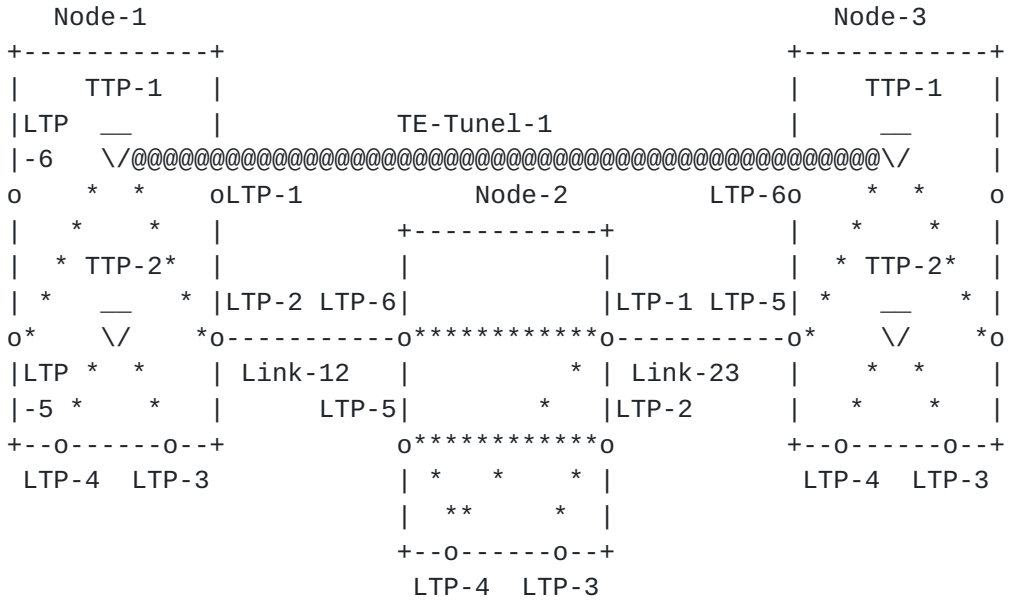


Figure 1: TE Topology Modeling Abstractions

3.1. TE Topology

TE topology is a traffic engineering representation of one or more layers of network topologies. TE topology is comprised of TE nodes (TE graph vertices) interconnected via TE links (TE graph edges). A TE topology is mapped to a TE graph.

3.2. TE Node

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

Multi-layer TE nodes providing switching functions at multiple network layers are an example where a physical node can be decomposed into multiple logical TE nodes (fractions of a node). Some of these (logical) TE nodes may reside in the client layer TE topology while the remaining TE nodes belong to the server layer TE topology.

In Figure 1, Node-1, Node-2, and Node-3 are TE nodes.

3.3. TE Link

TE link is an element of a TE topology (presented as an edge on TE graph, arrows indicate one or both directions of the TE link). TE link represents one or several (physical) links or a fraction of a link. TE link belongs to and is fully defined in exactly one TE topology. TE link is assigned with the TE topology scope unique ID. TE link attributes include parameters related to the data plane aspects of the associated link(s) (e.g. unreserved bandwidth, resource maps/pools, etc.), as well as the configuration data (such as remote node/link IDs, SRLGs, administrative colors, etc.). TE link is connected to TE node, terminating the TE link via exactly one TE link termination point (LTP).

In Figure 1, Link-12 and Link-23 are TE links.

3.4. Transitional TE Link for Multi-Layer Topologies

Networks are typically composed of multiple network layers where one or multiple signals in the client layer network can be multiplexed and encapsulated into a server layer signal. The server layer signal can be carried in the server layer network across multiple nodes until the server layer signal is terminated and the client layer signals reappear in the node that terminates the server layer signal. Examples of multi-layer networks are: IP over MPLS over Ethernet, low order ODUk signals multiplexed into a high order ODUl (l>k) carried over an OCh signal (optical transport network).

TE links as defined in 3.3. can be used to represent links within a network layer. In case of a multi-layer network, TE nodes and TE links only allow representation of each network layer as a separate TE topology Each of these single layer TE topologies would be isolated from their client and their server layer TE topology, if present (the highest and the lowest network layer in the hierarchy only have a single adjacent layer below or above, respectively). Multiplexing of client layer signals and encapsulating them into a server layer signal requires a function that is provided inside a node (typically realized in hardware). This function is also called layer transition.

One of the key requirements for path computation is to be able to calculate a path between two endpoints across a multi-layer network based on the TE topology representing this multi-layer network. This means that an additional TE construct is needed that represents

potential layer transitions in the multi-layer TE-topology that connects the TE-topologies representing each separate network layer. The so-called transitional TE link is such a construct and it represents the layer transition function residing inside a node that is decomposed into multiple logical nodes that are represented as TE nodes. Hence, a transitional TE link connects a client layer node with a server layer node. A TE link as defined in 3.3. has LTPs of exactly the same kind on each link end whereas the transitional TE link has client layer LTPs on the client side of the transitional link and in most cases a single server layer LTP on the server side. It should be noted that transitional links are a helper construct in the multi-layer TE topology and they only exist as long as they are not in use (as they represent potential connectivity). When the server layer trail has been established between the server layer LTP of two transitional links in the server layer network, the resulting client layer link in the data plane will be represented as a normal TE link in the client layer topology. The transitional TE links will re-appear when the server layer trail has been torn down.

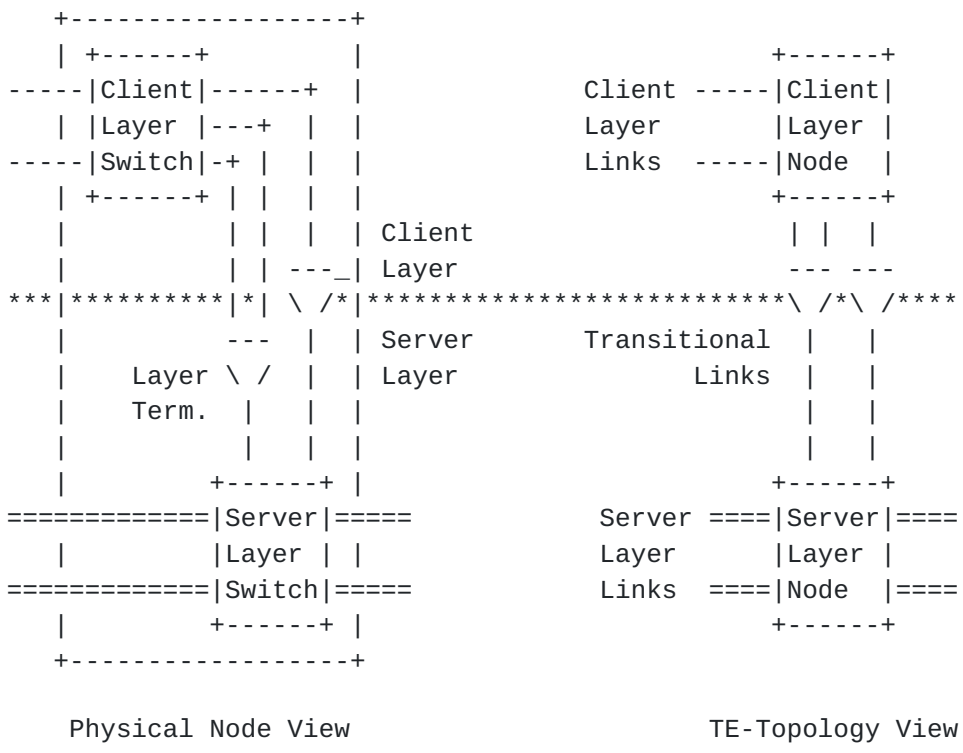


Figure 2: Modeling a Multi-Layer Node (Dual-Layer Example)

3.5. TE Link Termination Point (LTP)

TE link termination point (LTP) is a conceptual point of connection of a TE node to one of the TE links, terminated by the TE node. Cardinality between an LTP and the associated TE link is 1:0..1.

In Figure 1, Node-2 has six LTPs: LTP-1 to LTP-6.

3.6. TE Tunnel Termination Point (TTP)

TE tunnel termination point (TTP) is an element of TE topology representing one or several of potential transport service termination points (i.e. service client adaptation points such as WDM/OCh transponder). TTP is associated with (hosted by) exactly one TE node. TTP is assigned with the TE node scope unique ID. Depending on the TE node's internal constraints, a given TTP hosted by the TE node could be accessed via one, several or all TE links terminated by the TE node.

In Figure 1, Node-1 has two TTPs: TTP-1 and TTP-2.

3.7. TE Node Connectivity Matrix

TE node connectivity matrix is a TE node's attribute describing the TE node's switching limitations in a form of valid switching combinations of the TE node's LTPs (see below). From the point of view of a potential TE path arriving at the TE node at a given inbound LTP, the node's connectivity matrix describes valid (permissible) outbound LTPs for the TE path to leave the TE node from.

In Figure 1, the connectivity matrix on Node-2 is:
{<LTP-6, LTP-1>, <LTP-5, LTP-2>, <LTP-5, LTP-4>, <LTP-4, LTP-1>, <LTP-3, LTP-2>}

3.8. TTP Local Link Connectivity List (LLCL)

TTP Local Link Connectivity List (LLCL) is a List of TE links terminated by the TTP hosting TE node (i.e. list of the TE link LTPs), which the TTP could be connected to. From the point of view of a potential TE path LLCL provides a list of valid TE links the TE path needs to start/stop on for the connection, taking the TE path, to be successfully terminated on the TTP in question.

In Figure 1, the LLCL on Node-1 is:
{<TTP-1, LTP-5>, <TTP-1, LTP-2>, <TTP-2, LTP-3>, <TTP-2, LTP4>}

3.9. TE Path

TE path is an ordered list of TE links and/or TE nodes on the TE topology graph, inter-connecting a pair of TTPs to be taken by a potential connection. TE paths, for example, could be a product of successful path computation performed for a given transport service.

In Figure 1, the TE Path for TE-Tunnel-1 is:
{Node-1:TTP-1, Link-12, Node-2, Link-23, Node-3:TTP1}

3.10. TE Inter-Layer Lock

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

TE inter-layer lock are decorated with the same inter-layer lock ID attribute.

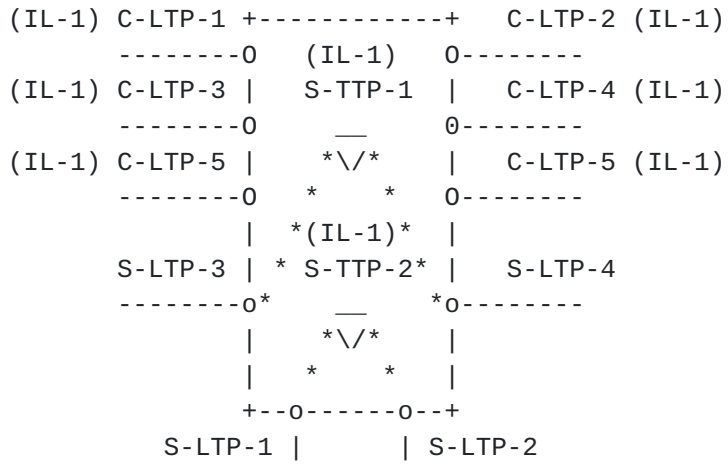


Figure 3: TE Inter-Layer Lock ID Associations

On the picture above a TE inter-layer lock with IL_1 ID associates 6 client layer LTPs (C-LTP-1 - C-LTP-6) with two server layer TTPs (S-TTP-1 and S-TTP-2). They all have the same attribute - TE inter-layer lock ID: IL-1, which is the only thing that indicates the association. A given LTP may have 0, 1 or more inter-layer lock IDs. In the latter case this means that the data arriving at the LTP may be adopted onto any of TTPs associated with all specified inter-layer locks. For example, C-LTP-1 could have two inter-layer lock IDs - IL-1 and IL-2. This would mean that C-LTP-1 for adaptation purposes could use not just TTPs associated with inter-layer lock IL-1 (i.e. S-TTP-1 and S-TTP-2 on the picture), but any of TTPs associated with inter-layer lock IL-2 as well. Likewise, a given TTP may have one or more inter-layer lock IDs, meaning that it can offer the adaptation service to any of client layer LTPs with inter-layer lock ID matching one of its own. Additionally, each TTP has an attribute - Unreserved Adaptation Bandwidth, which announces its remaining adaptation resources sharable between all potential client LTPs.

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

3.11. Underlay TE topology

Underlay TE topology is a TE topology that serves as a base for constructing of overlay TE topologies

3.12. Overlay TE topology

Overlay TE topology is a TE topology constructed based on one or more underlay TE topologies. Each TE node of the overlay TE topology represents an arbitrary segment of an underlay TE topology; each TE link of the overlay TE topology represents an arbitrary TE path in one of the underlay TE topologies. The overlay TE topology and the supporting underlay TE topologies may represent distinct layer networks (e.g. OTN/ODUk and WDM/OCh respectively) or the same layer network.

3.13. Abstract TE topology

Abstract TE topology is a topology that contains abstract topological elements (nodes, links, tunnel termination points). Abstract TE topology is an overlay TE topology created by a topology provider and customized for a topology provider's client based on one or more of the provider's native TE topologies (underlay TE topologies), the provider's policies and the client's preferences. For example, a first level topology provider (such as Domain Controller) can create an abstract TE topology for its client (e.g. Multi-Domain Service Coordinator) based on the provider's one or more native TE topologies, local policies/profiles and the client's TE topology configuration requests

Figure 4 shows an example of abstract TE topology.

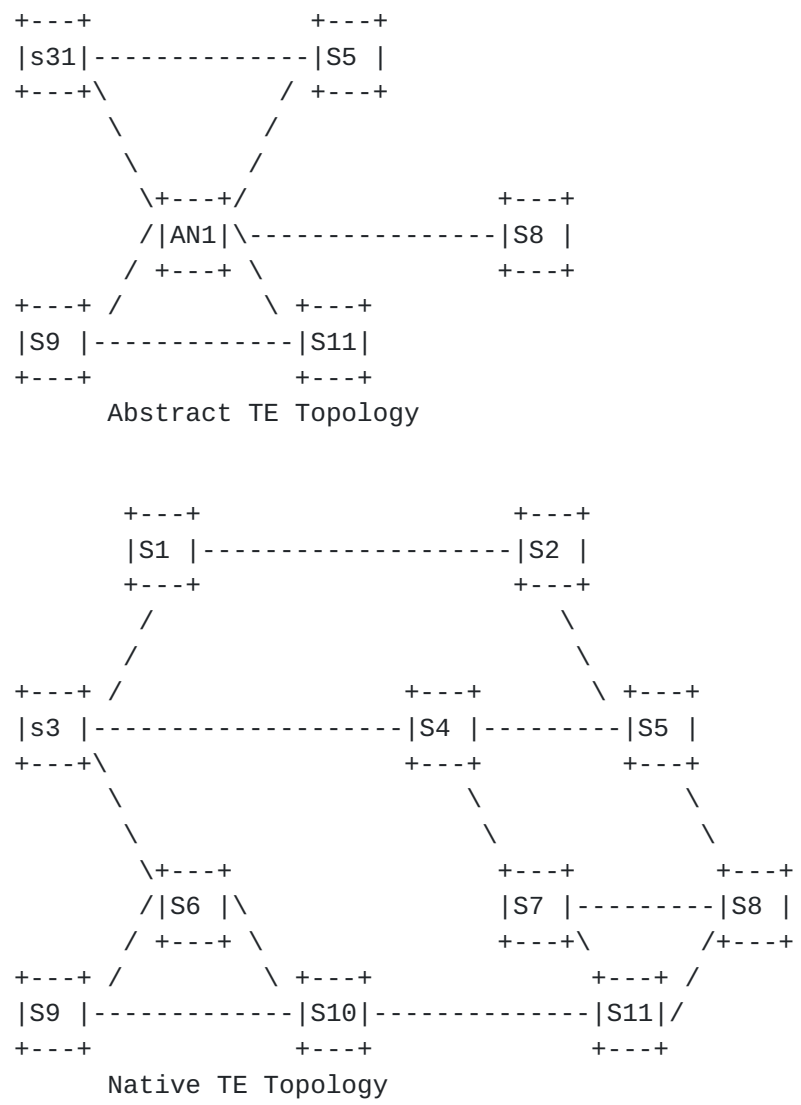


Figure 4: Abstract TE Topology

4. Model Applicability

4.1. Native TE Topologies

The model discussed in this draft can be used to represent and retrieve native TE topologies on a given TE system.

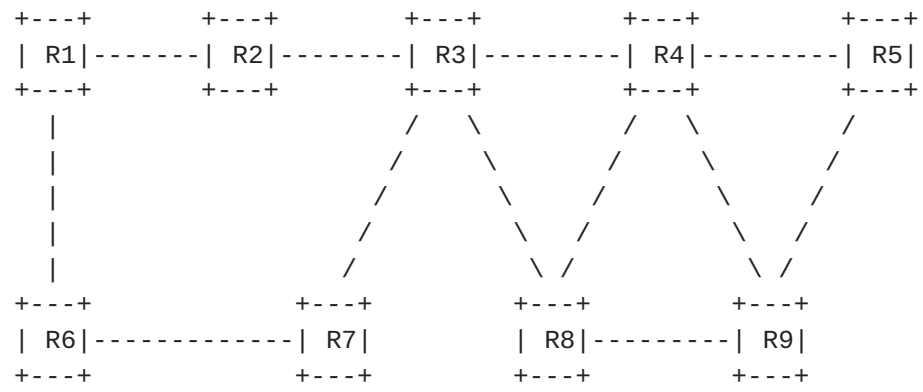


Figure 5a: Example Network Topology

Consider the network topology depicted in Figure 5a (R1 .. R9 are nodes representing routers). An implementation MAY choose to construct a native TE Topology using all nodes and links present in the given TED as depicted in Figure 5b. The data model proposed in this document can be used to retrieve/represent this TE topology.

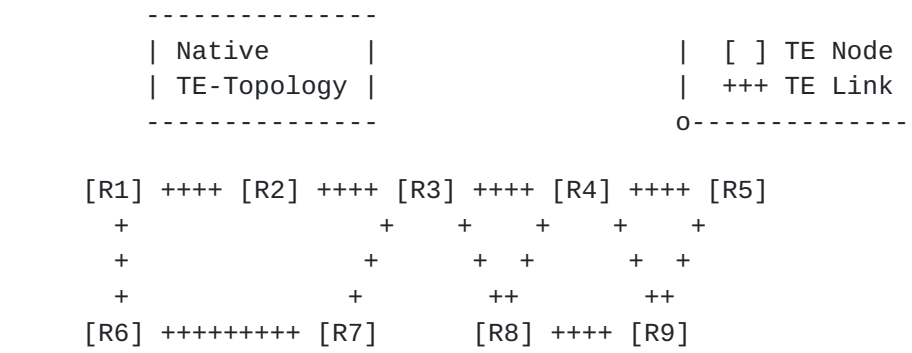


Figure 5b: Native TE Topology as seen on Node R3

Consider the case of the topology being split in a way that some nodes participate in OSPF-TE while others participate in ISIS-TE (Figure 6a). An implementation MAY choose to construct separate TE Topologies based on the information source. The native TE Topologies constructed using only nodes and links that were learnt via a specific information source are depicted in Figure 6b. The data model proposed in this document can be used to retrieve/represent these TE topologies.

Similarly, the data model can be used to represent/retrieve a TE Topology that is constructed using only nodes and links that belong to a particular technology layer. The data model is flexible enough to retrieve and represent many such native TE Topologies.

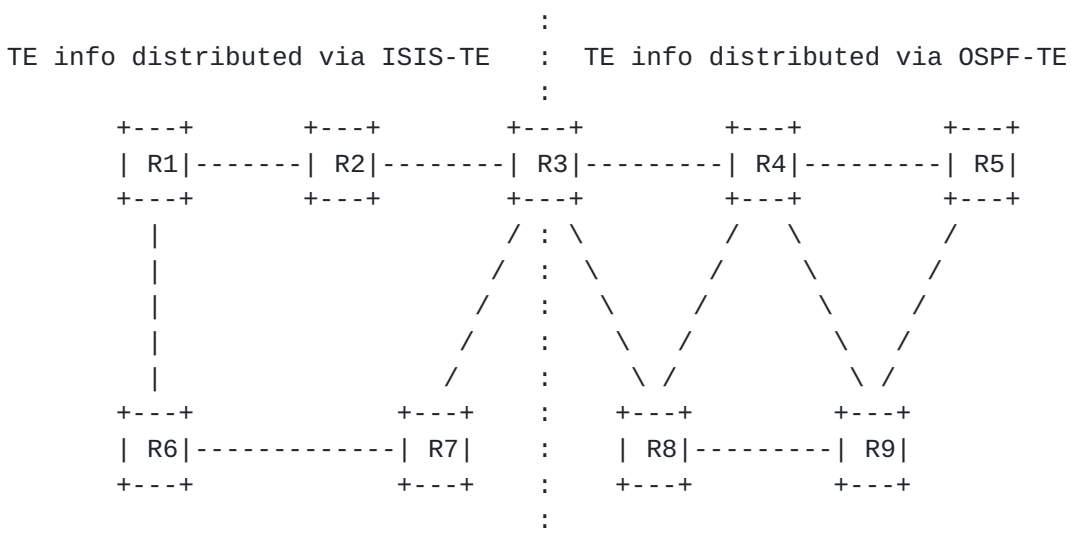


Figure 6a: Example Network Topology

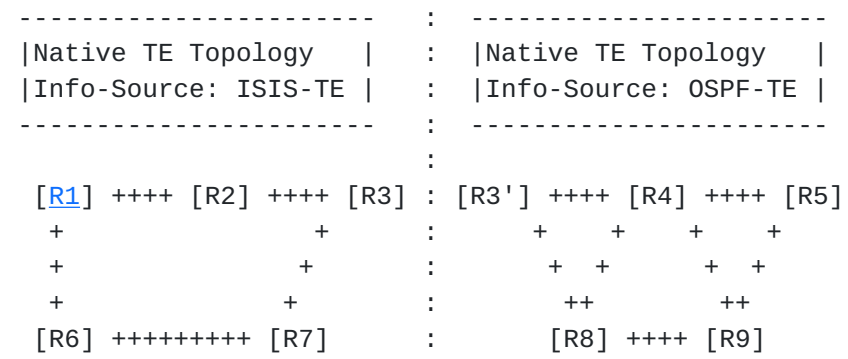


Figure 6b: Native TE Topologies as seen on Node R3

4.2. Customized TE Topologies

Customized TE topology is a topology that was modified by the provider to honor a particular client's requirements or preferences. The model discussed in this draft can be used to represent, retrieve and manipulate customized TE Topologies. The model allows the provider to present the network in abstract TE Terms on a per client basis. These customized topologies contain sufficient information for the path computing client to select paths according to its policies.

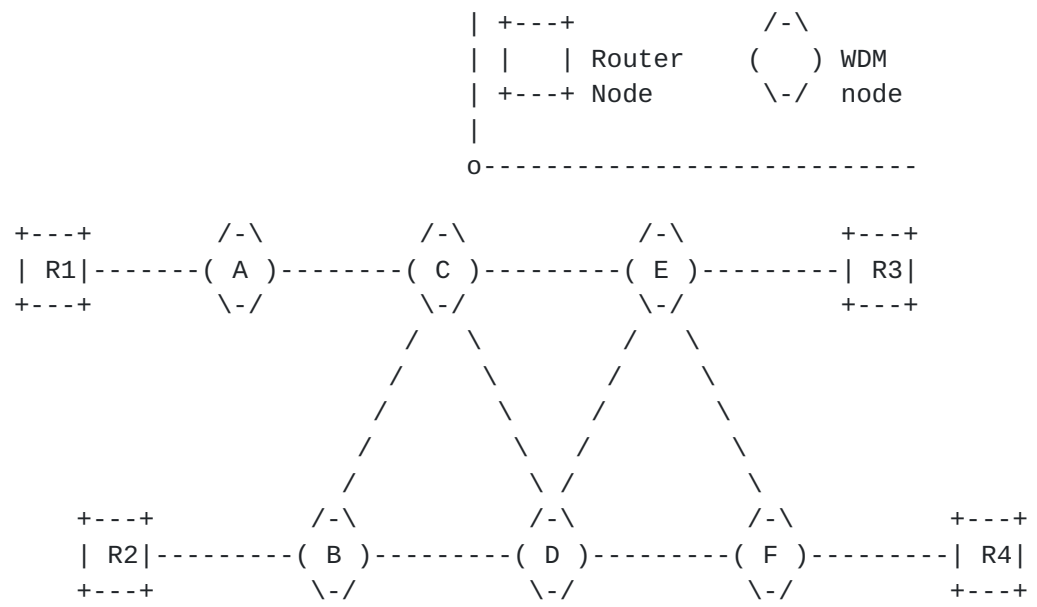


Figure 7: Example packet optical topology

Consider the network topology depicted in Figure 7. This is a typical packet optical transport deployment scenario where the WDM layer network domain serves as a Server Network Domain providing transport connectivity to the packet layer network Domain (Client Network Domain). Nodes R1, R2, R3 and R4 are IP routers that are connected to an Optical WDM transport network. A, B, C, D, E and F are WDM nodes that constitute the Server Network Domain.

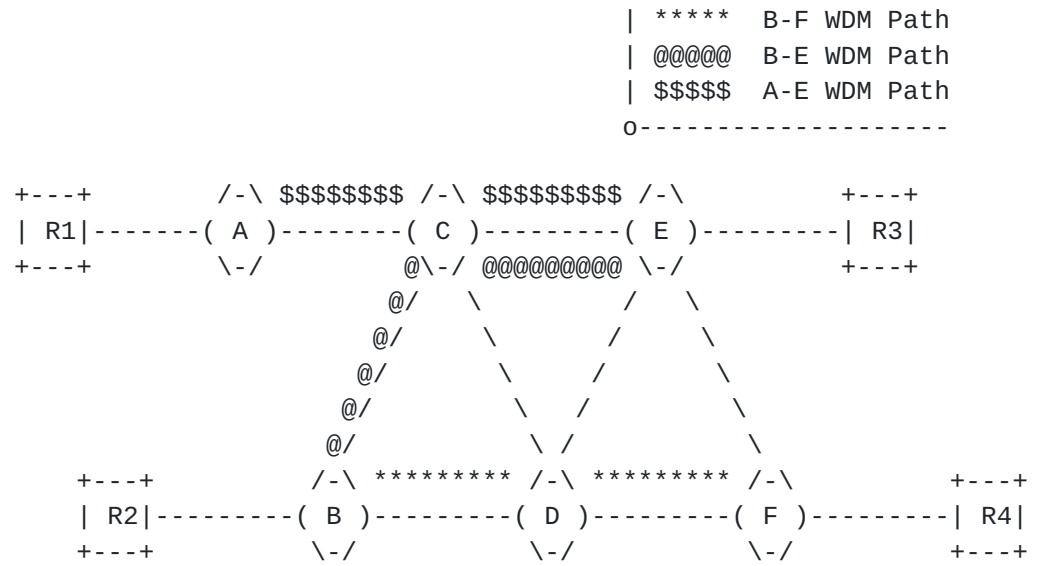


Figure 8a: Paths within the provider domain

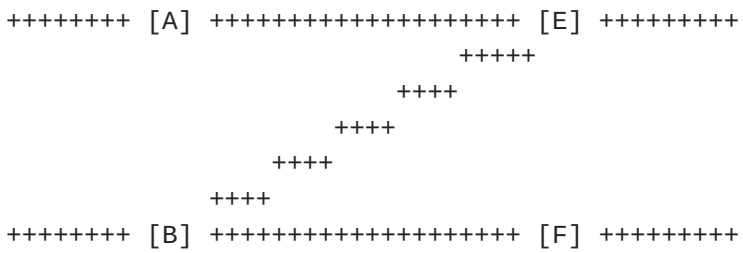


Figure 8b: Customized TE Topology provided to the Client

The goal here is to augment the Client TE Topology with a customized TE Topology provided by the WDM network. Given the availability of the paths A-E, B-F and B-E (Figure 8a), a customized TE Topology as depicted in Figure 8b is provided to the Client. This customized TE Topology is merged with the Client's Native TE Topology and the resulting topology is depicted in Figure 8c.

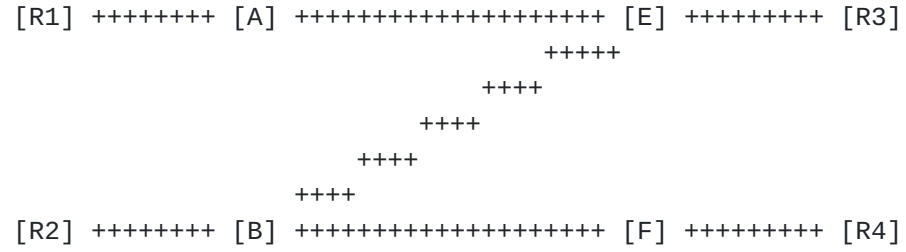


Figure 8c: Customized TE Topology merged with the Client's Native TE Topology

The data model proposed in this document can be used to retrieve/represent/manipulate the customized TE Topology depicted in Figure 8b.

A customized TE topology is not necessarily an abstract TE topology. he provider may produce, for example, an abstract TE topology of certain type (e.g. single-abstract-node-with-connectivit-matrix topology, a border_nodes_connected_via_mesh_of_abstract_links topology, etc.) and expose it to all/some clients in expectation that the clients will use it without customization.
On the other hand, a client may request a customized version of the provider's native TE topology (e.g. by requesting removal of TE links which belong to certain layers, are too slow, not protected and/or

have a certain affinity). Note that the resulting TE topology will not be abstract (because it will not contain abstract elements), but customized (modified upon client's instructions).

The client ID field in the TE topology identifier (Section 5.4. indicates which client the TE topology is customized for. Although a authorized client MAY receive a TE topology with the client ID field matching some other client, the client can customize only TE topologies with the client ID field either 0 or matching the ID of the client in question. If the client starts reconfiguration of a topology its client ID will be automatically set in the topology ID field for all future configurations and updates wrt. the topology in question.

The provider MAY tell the client that a given TE topology cannot be re-negotiated, by setting its own (provider's) ID in the client ID field of the topology ID.

4.3. Merging TE Topologies Provided by Multiple Providers

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

In particular, the process of merging TE topologies includes:

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

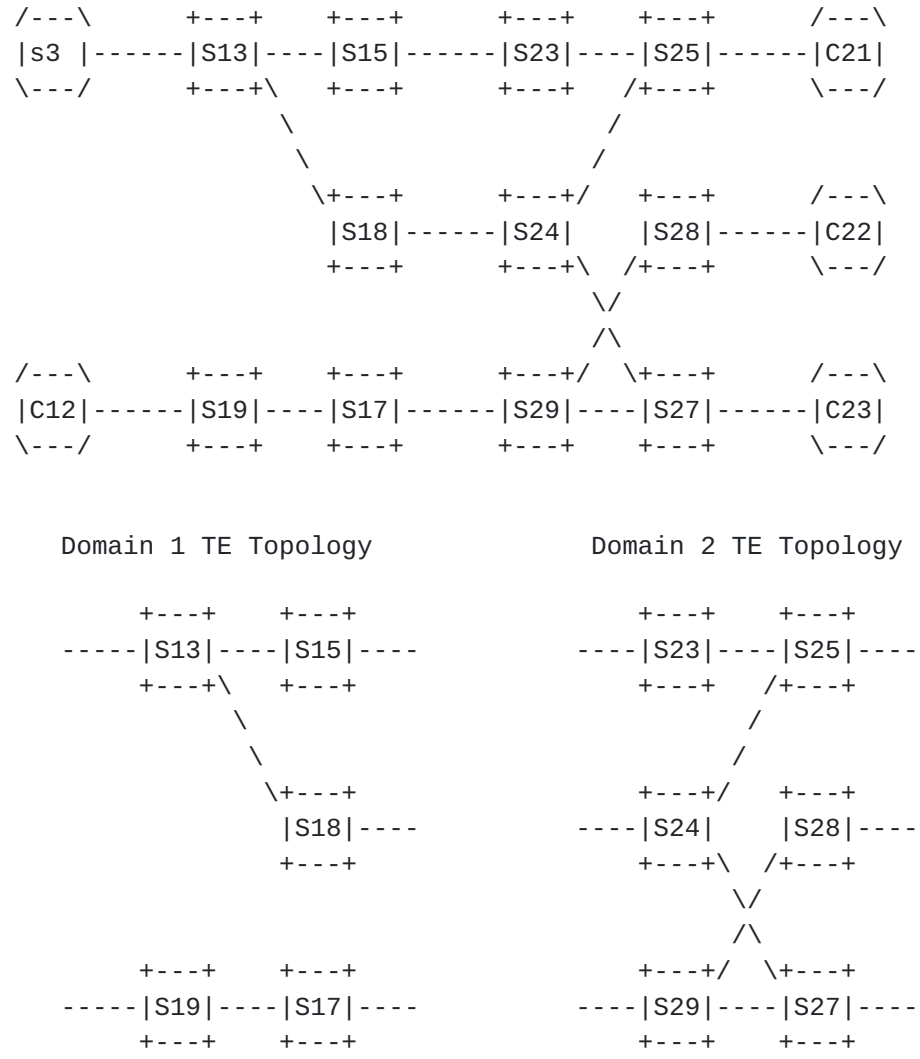


Figure 9: Merging Domain TE Topologies

Figure 9 illustrates the process of merging, by the client, of TE topologies provided by the client's providers. In the Figure, each of the two providers caters to the client (abstract or native) TE topology, describing the network domain under the respective provider's control. The client, by consulting the attributes of the inter-domain TE links - such as inter-domain plug IDs or remote TE node/link IDs (as defined by the TE Topology model) - is able to determine that:

a) the two domains are adjacent and are inter-connected via three inter-domain TE links, and;

b) each domain is connected to a separate customer site, connecting the left domain in the Figure to customer devices C-11 and C-12, and the right domain to customer devices C-21, C-22 and C-23.

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

As mentioned, one way to inter-connect the open-ended inter-domain TE links of neighboring domains is to mandate the providers to specify remote nodeID/linkID attribute in the provided inter-domain TE links. This, however, may prove to be not flexible. For example, the providers may not know the respective remote nodeIDs/ linkIDs. More importantly, this option does not allow for the client to mix-n-match multiple (more than one) topologies catered by the same providers (see below). Another, more flexible, option to resolve the open-ended inter-domain TE links is by decorating them with the inter-domain plug ID attribute. Inter-domain plug ID is a network-wide unique number that identifies on the network a connectivity supporting a given inter-domain TE link. Instead of specifying remote node ID/link ID, an inter-domain TE link may provide a non-zero inert-domain plug ID. It is expected that two neighboring domain TE topologies (provided by separate providers) will have each at least one open-ended inter-domain TE link with an inter-domain plug ID matching to one provided by its neighbor. For example, the inter-domain TE link originating from node S5 of the Domain 1 TE topology (Figure 1) and the inter-domain TE link coming from node S3 of Domain2 TE topology may specify matching inter-domain plug ID (e.g. 175344) This allows for the client to identify adjacent nodes in the separate neighboring TE topologies and resolve the inter-domain TE links connecting them regardless of their respective nodeIDs/linkIDs (which, as mentioned, could be allocated from independent name spaces). Inter-domain plug IDs may be assigned and managed by a central network authority. Alternatively, inter-domain plug IDs could be dynamically auto-discovered (e.g. via LMP protocol).

Furthermore, the client renames the TE nodes, links and SRLGs offered in the abstract TE topologies by assigning to them IDs allocated from a separate name space managed by the client. Such renaming is necessary, because the two abstract TE topologies may have their own name spaces, generally speaking, independent one from another; hence, ID overlaps/clashes are possible. For example, both TE topologies have TE nodes named S7, which, after renaming, appear in the merged TE topology as S17 and S27, respectively.

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

4.4. Dealing with Multiple Abstract TE Topologies Provided by the Same Provider

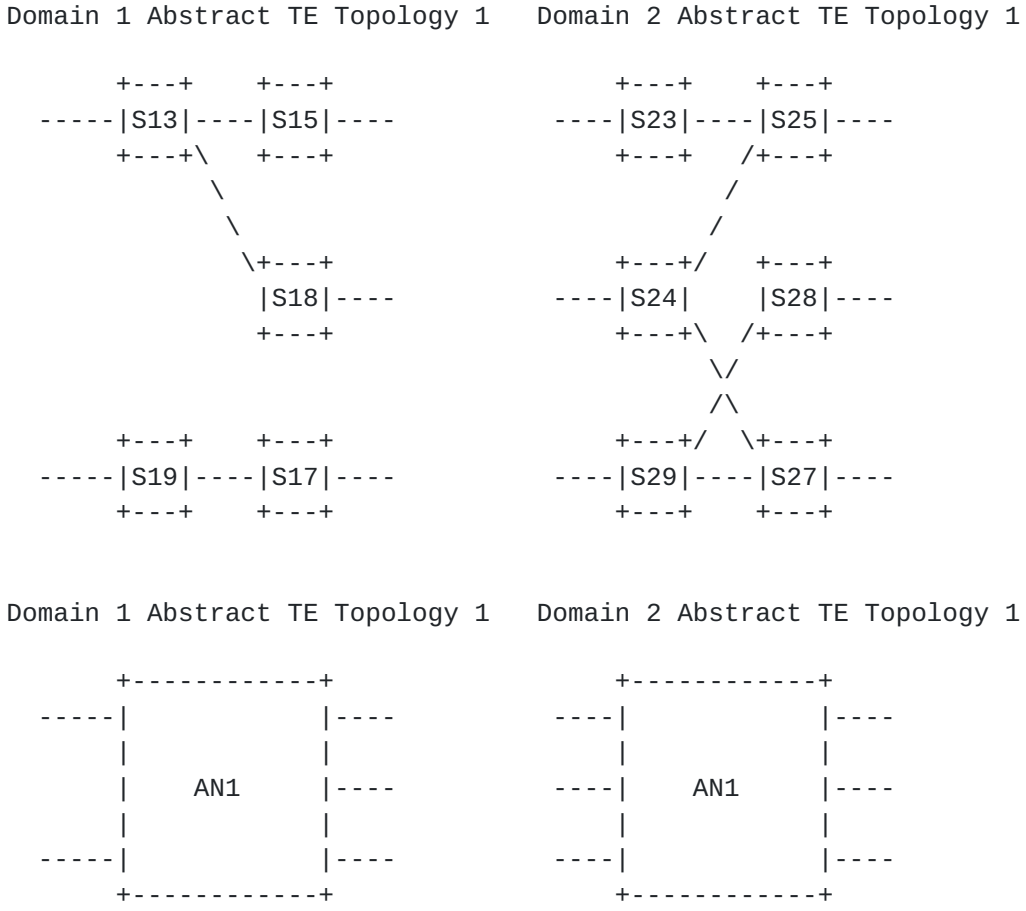


Figure 10: Merging Domain TE Topologies

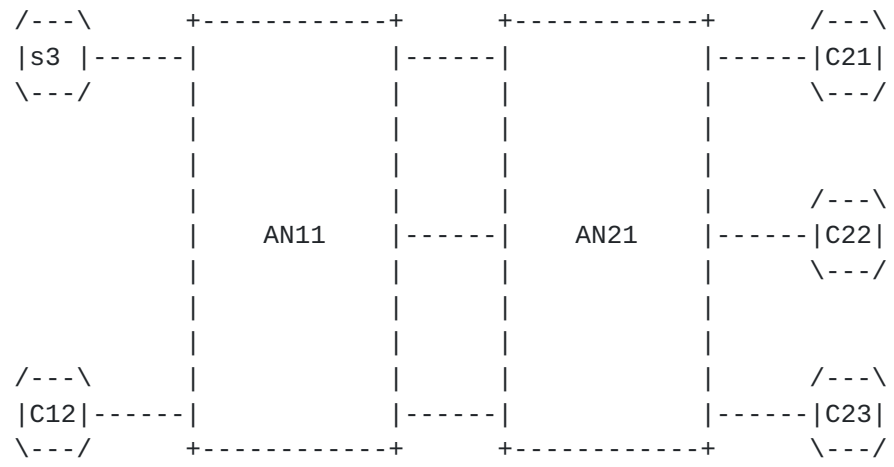
Based on local configuration, templates and/or policies pushed by the client, a given provider may expose more than one abstract TE topology to the client. For example, one abstract TE topology could be optimized based on a lowest-cost criterion, while another one could be based on best possible delay metrics, while yet another one could be based on maximum bandwidth availability for the client services. Furthermore, the client may request all or some providers to expose additional abstract TE topologies, possibly of a different type and/or optimized differently, as compared to already-provided TE

topologies. In any case, the client should be prepared for a provider to offer to the client more than one abstract TE topology.

It should be up to the client (based on the client's local configuration and/or policies conveyed to the client by the client's clients) to decide how to mix-and-match multiple abstract TE topologies provided by each or some of the providers, as well as how to merge them into the client's native TE topologies. The client also decides how many such merged TE topologies it needs to produce and maintain. For example, in addition to the merged TE topology depicted in the upper part of Figure 1, the client may merge the abstract TE topologies received from the two providers, as shown in Figure 10, into the client's additional native TE topologies, as shown in Figure 11.

Note that allowing for the client mix-n-matching of multiple TE topologies assumes that inter-domain plug IDs (rather than remote nodeID/linkID) option is used for identifying neighboring domains and inter-domain TE link resolution.

Client's Merged TE Topology 2



Client's Merged TE Topology 3

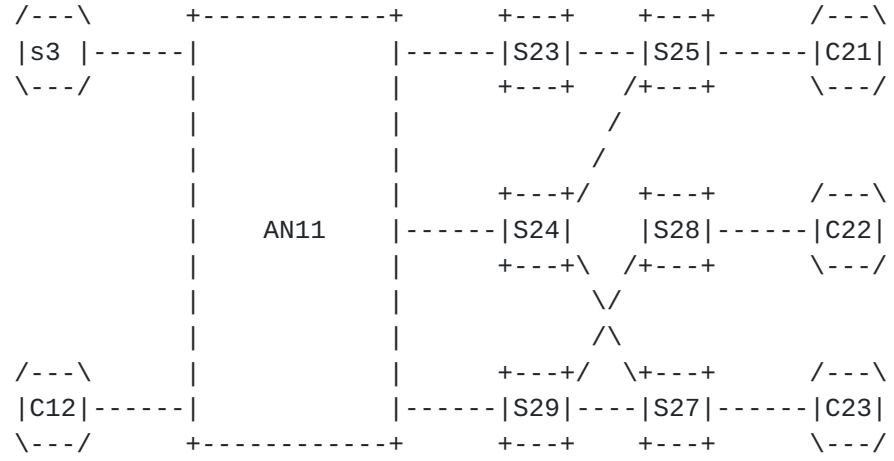


Figure 11: Multiple Native (Merged) Client's TE Topologies

It is important to note that each of the three native (merged) TE topologies could be used by the client for computing TE paths for any of the multi-domain services. The choice as to which topology to use for a given service depends on the service parameters/requirements and the topology's style, optimization criteria and the level of details.

5. Modeling Considerations

5.1. Generic network topology building blocks

The generic network topology building blocks are discussed in [YANG-NET-TOPO]. The TE Topology model proposed in this document augments and uses the ietf-network-topology module defined in [YANG-NET-TOPO].

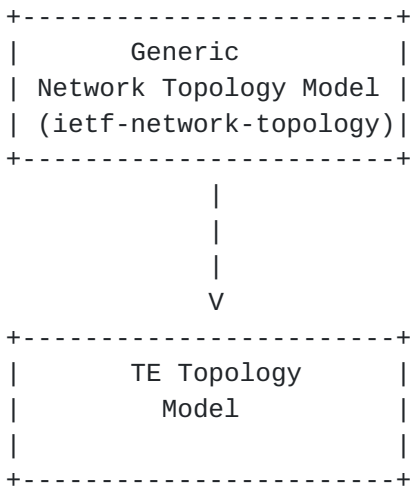


Figure 12: Augmenting the Generic Network Topology Model

5.2. Technology agnostic TE Topology 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.

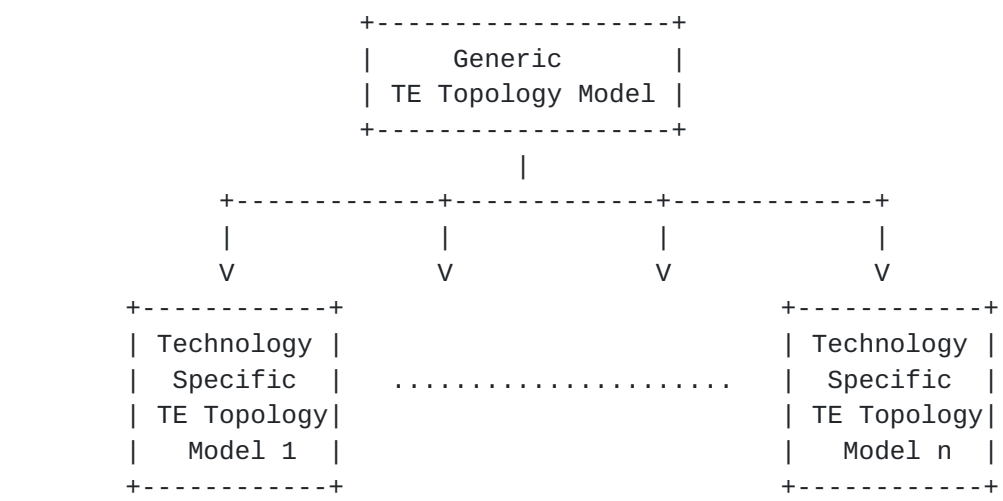


Figure 13: Augmenting the Technology agnostic TE Topology model

5.3. Model Structure

The high-level model structure proposed by this document is as shown below:

```
module: ietf-te-topology
augment /nw:networks/nw:network/nw:network-types:
   +--rw te-topology!

augment /nw:networks:
   +--rw te!
      +--rw templates
         +--rw node-template* [name] {template}?
         |  .............
         +--rw link-template* [name] {template}?
            .............

augment /nw:networks/nw:network:
   +--rw provider-id?      te-types:te-global-id
   +--rw client-id?        te-types:te-global-id
   +--rw te-topology-id?   te-types:te-topology-id
   +--rw te!
      +--rw config
      |  .............
      +--ro state
         .............

augment /nw:networks/nw:network/nw:node:
   +--rw te-node-id?   te-types:te-node-id
```

```
      +--rw te!
         +--rw config
         |  ............
         +--ro state
         |  ............
         +--rw tunnel-termination-point* [tunnel-tp-id]
            +--rw tunnel-tp-id    binary
            +--rw config
            |  ............
            +--ro state

   augment /nw:networks/nw:network/nt:link:
      +--rw te!
         +--rw config
         |  ..........
         +--ro state
            ..........

   augment /nw:networks/nw:network/nw:node/nt:termination-point:
      +--rw te-tp-id?   te-types:te-tp-id
      +--rw te!
         +--rw config
         |  ............
         +--ro state
            ............
```

5.4. Topology Identifiers

The TE-Topology is uniquely identified by a key that has 3 constituents - te-topology-id, provider-id and client-id. The combination of provider-id and te-topology-id uniquely identifies a native TE Topology on a given provider. The client-id is used only when Customized TE Topologies come into play; a value of "0" is used as the client-id for native TE Topologies.

```
   augment /nw:networks/nw:network:
      +--rw te!
         +--rw provider-id       te-global-id
         +--rw client-id         te-global-id
         +--rw te-topology-id    te-topology-id
```

5.5. Generic TE Link Attributes

The model covers the definitions for generic TE Link attributes - bandwidth, admin groups, SRLGs, switching capabilities, TE metric extensions etc.

```
     +--rw te-link-attributes
        ......................
        +--rw admin-status?                    te-admin-status
        |  ......................
        +--rw link-index?                      uint64
        +--rw administrative-group?            te-types:admin-groups
        +--rw link-protection-type?            enumeration
        +--rw max-link-bandwidth?              te-bandwidth
        +--rw max-resv-link-bandwidth?         te-bandwidth
        +--rw unreserved-bandwidth* [priority]
        |  ......................
        +--rw te-default-metric?               uint32
        |  ......................
        +--rw te-srlgs
        +--rw te-nsrlgs {nsrlg}?       ......................
```

5.6. Generic TE Node Attributes

The model covers the definitions for generic TE Node attributes.

The definition of a generic connectivity matrix is shown below:

```
     +--rw te-node-attributes
        ............
        +--rw connectivity-matrices
        ............
        |  +--rw connectivity-matrix* [id]
        |  |  +--rw id            uint32
        |  |  +--rw from
        |  |  |  +--rw tp-ref?        leafref
        |  |  +--rw to
        |  |  |  +--rw tp-ref?        leafref
        |  |  +--rw is-allowed?   boolean
        |  |  +--rw label-restriction* [inclusive-exclusive label-start]
        ............
        |  |  +--rw underlay! {te-topology-hierarchy}?
        ............
        |  |  +--rw max-link-bandwidth?        te-bandwidth
        |  |  +--rw max-resv-link-bandwidth?   te-bandwidth
        |  |  +--rw unreserved-bandwidth* [priority]
        ............
        |  |  +--rw te-default-metric?         uint32
        |  |  +--rw te-delay-metric?           uint32
        |  |  +--rw te-srlgs
```

```
      |  |  +--rw te-nsrlgs {nsrlg}?     ...........The definition of
   a TTP Local Link Connectivity List is shown below:

     +--rw tunnel-termination-point* [tunnel-tp-id]
        +--rw tunnel-tp-id    binary
        +--rw config
        |  +--rw switching-capability?      identityref
        |  +--rw encoding?                  identityref
        |  +--rw inter-layer-lock-id?       uint32
        |  +--rw protection-type?           identityref
        |  +--rw local-link-connectivities
        ...........
        |     +--rw local-link-connectivity* [link-tp-ref]
        |        +--rw link-tp-ref               leafref
        |        +--rw label-restriction* [inclusive-exclusive label-
   start]
        ...........
        |        +--rw max-lsp-bandwidth* [priority]
        ...........
        |        +--rw max-link-bandwidth?        te-bandwidth
        |        +--rw max-resv-link-bandwidth?   te-bandwidth
        |        +--rw unreserved-bandwidth* [priority]
        ...........
        |        +--rw te-default-metric?         uint32
        |        +--rw te-delay-metric?           uint32
        |        +--rw te-srlgs
        |        +--rw te-nsrlgs {nsrlg}?
        ...........
        +--ro state
        |  +--ro switching-capability?      identityref
        |  +--ro encoding?                  identityref
        |  +--ro inter-layer-lock-id?       uint32
        |  +--ro protection-type?           identityref
        |  +--ro local-link-connectivities
        ...........
        |     +--ro local-link-connectivity* [link-tp-ref]
        |     |  +--ro link-tp-ref               leafref
        |     |  +--ro label-restriction* [inclusive-exclusive label-
   start]
        ...........
        |     |  +--ro max-lsp-bandwidth* [priority]
        ...........
        |     |  +--ro max-link-bandwidth?        te-bandwidth
        |     |  +--ro max-resv-link-bandwidth?   te-bandwidth
        |     |  +--ro unreserved-bandwidth* [priority]
        ...........
```

```
      |     |  +--ro te-default-metric?        uint32
      |     |  +--ro te-delay-metric?          uint32
      |     |  +--ro te-srlgs
      |     |  +--ro te-nsrlgs {nsrlg}?
      .....
      +--rw supporting-tunnel-termination-point* [node-ref tunnel-tp-
ref]
        +--rw node-ref         union
        +--rw tunnel-tp-ref    union
```

The attributes directly under container connectivity-matrices are the default attributes for all connectivity-matrix entries when the per entry corresponding attribute is not specified. When a per entry attribute is specified, it overrides the cooresponding attribute directly under the container connectivity-matrices. The same rule applies to the attributes directly under container local-link-connectivities.

Each TTP (Tunnel Termination Point) MAY be supported by one or more supporting TTPs. If the TE node hosting the TTP in question refers to a supporting TE node, then the supporting TTPs are hosted by the supporting TE node. If the TE node refers to an underlay TE topology, the supporting TTPs are hosted by one or more specified TE nodes of the underlay TE topology.

5.7. TED Information Sources

The model allows each TE topological element to have multiple TE information sources (OSPF-TE, ISIS-TE, BGP-LS, User-Configured, System-Processed, Other). Each information source is associated with a credibility preference to indicate precedence. In scenarios where a customized TE Topology is merged into a Client's native TE Topology, the merged topological elements would point to the corresponding customized TE Topology as its information source.

```
augment /nw:networks/nw:network/nw:node:
   +--rw te!
      ...........
      +--ro state
         ........
      |  +--ro information-source?         te-info-source
      |  +--ro information-source-state
      |  |  +--ro credibility-preference?    uint16
      |  |  +--ro logical-network-element?   string
      |  |  +--ro network-instance?          string
      |  |  +--ro topology
```

```
         |  |     +--ro network-ref?       leafref
         |  |     +--ro node-ref?          leafref
         |  +--ro information-source-entry* [information-source]
               .............

   augment /nw:networks/nw:network/nt:link:
      +--rw te!
         ...........
         +--ro state
            .........
         |  +--ro information-source?         te-info-source
         |  +--ro information-source-state
         |  |  +--ro credibility-preference?    uint16
         |  |  +--ro logical-network-element?   string
         |  |  +--ro network-instance?          string
         |  |  +--ro topology
         |  |     +--ro network-ref?       leafref
         |  |     +--ro link-ref?          leafref
         |  +--ro information-source-entry* [information-source]
               .............
```

5.8. 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.

This relationship is captured via the "underlay-topology" field for the node and via the "underlay" field for the link. The use of these fields is optional and this functionality is tagged as a "feature" ("te-topology-hierarchy").

```
   augment /nw:networks/nw:network/nw:node:
      +--rw te!
         +--rw te-node-id    te-node-id
         +--rw config
         |  +--rw te-node-template*     leafref {template}?
         |  +--rw te-node-attributes
         |     ....................
         |     +--rw underlay-topology {te-topology-hierarchy}?
         |        +--rw network-ref?       leafref
```

```
   augment /nw:networks/nw:network/nt:link:
      +--rw te!
         +--rw config
         |  .........
         |  +--rw te-link-attributes
         |     ....................
         |     +--rw underlay! {te-topology-hierarchy}?
         |     |  +--rw primary-path
         |     |  |  +--rw network-ref?       leafref
         |     |  |  +--rw path-element* [path-element-id]
         |     |  |     ................
         |     |  +--rw backup-path* [index]
         |     |  |  +--rw index              uint32
         |     |  |  +--rw network-ref?       leafref
         |     |  |  +--rw path-element* [path-element-id]
         |     |  |     ................
         |     |  +--rw underlay-protection-type?   uint16
         |     |  +--rw underlay-tunnel-src
         |     |  |  ...........

         |     |  +--rw underlay-tunnel-des
         |     |     ...........
```

5.9. Templates

The data model provides the users with the ability to define templates and apply them to link and node configurations. The use of "template" configuration is optional and this functionality is tagged as a "feature" ("template").

```
     +--rw topology* [provider-id client-id te-topology-id]
     |  ...........
     |  +--rw node* [te-node-id]
     |  |  +--rw te-node-template?     leafref {template}?
     |  |     ..........
     |  +--rw link* [source-te-node-id source-te-link-id dest-te-node-id
   dest-te-link-id]
     |     +--rw te-link-template?     leafref {template}?
     |        ..........

   augment /nw:networks:
      +--rw te!
         +--rw templates
            +--rw node-template* [name] {template}?
```

```
          |  +--rw name                        te-types:te-template-
name
          |  +--rw priority?                   uint16
          |  +--rw reference-change-policy?    enumeration
          |  +--rw te-node-attributes
             ..........
          +--rw link-template* [name] {template}?
             +--rw name                        te-types:te-template-
name
             +--rw priority?                   uint16
             +--rw reference-change-policy?    enumeration
             +--rw te-link-attributes
             ..........
```

Multiple templates can be specified to a configuration element. When two or more templates specify values for the same configuration field, the value from the template with the highest priority is used. The reference-change-policy specifies the action that needs to be taken when the template changes on a configuration element that has a reference to this template. The choices of action include taking no action, rejecting the change to the template and applying the change to the corresponding configuration. [Editor's Note: The notion of "templates" has wider applicability. It is possible for this to be discussed in a separate document.]

5.10. Scheduling Parameters

The model allows time scheduling parameters to be specified for each topological element or for the topology as a whole. These parameters allow the provider to present different topological views to the client at different time slots. The use of "scheduling parameters" is optional.

The YANG data model for configuration scheduling is defined in [YANG-SCHEDULE], which allows specifying configuration schedules without altering this data model.

5.11. Notifications

Notifications are a key component of any topology data model.

[YANG-PUSH] and [RFC5277bis] define a subscription and push mechanism for YANG datastores. This mechanism currently allows the user to:

- Subscribe notifications on a per client basis

- Specify subtree filters or xpath filters so that only interested contents will be sent.
- Specify either periodic or on-demand notifications.

6. Tree Structure

```
module: ietf-te-topology
augment /nw:networks/nw:network/nw:network-types:
   +--rw te-topology!
augment /nw:networks:
   +--rw te!
      +--rw templates
         +--rw node-template* [name] {template}?
         |  +--rw name                       te-types:te-template-
name
         |  +--rw priority?                  uint16
         |  +--rw reference-change-policy?   enumeration
         |  +--rw te-node-attributes
         |     +--rw admin-status?        te-types:te-admin-status
         |     +--rw domain-id?           uint32
         |     +--rw is-abstract?         empty
         |     +--rw name?                inet:domain-name
         |     +--rw signaling-address*   inet:ip-address
         |     +--rw underlay-topology {te-topology-hierarchy}?
         |        +--rw network-ref?   leafref
         +--rw link-template* [name] {template}?
            +--rw name                       te-types:te-template-
name
            +--rw priority?                  uint16
            +--rw reference-change-policy?   enumeration
            +--rw te-link-attributes
               +--rw access-type?                      te-types:te-
link-access-type
               +--rw external-domain
               |  +--rw network-ref?            leafref
               |  +--rw remote-te-node-id?      te-types:te-node-id
               |  +--rw remote-te-link-tp-id?   te-types:te-tp-id
               |  +--rw plug-id?                uint32
               +--rw is-abstract?                      empty
               +--rw name?                             string
               +--rw underlay! {te-topology-hierarchy}?
               |  +--rw primary-path
               |  |  +--rw network-ref?    leafref
               |  |  +--rw path-element* [path-element-id]
               |  |     +--rw path-element-id    uint32
               |  |     +--rw index?             uint32
```

```
                  |  |     +--rw (type)?
                  |  |        +--:(ip-address)
                  |  |        |  +--rw ip-address-hop
                  |  |        |     +--rw address?    inet:ip-address
                  |  |        |     +--rw hop-type?   te-hop-type
                  |  |        +--:(as-number)
                  |  |        |  +--rw as-number-hop
                  |  |        |     +--rw as-number?   binary
                  |  |        |     +--rw hop-type?    te-hop-type
                  |  |        +--:(unnumbered-link)
                  |  |        |  +--rw unnumbered-hop
                  |  |        |     +--rw router-id?      inet:ip-
   address
                  |  |        |     +--rw interface-id?   uint32
                  |  |        |     +--rw hop-type?       te-hop-type
                  |  |        +--:(label)
                  |  |        |  +--rw label-hop
                  |  |        |     +--rw value?   rt-types:generalized-
   label
                  |  |        +--:(sid)
                  |  |           +--rw sid-hop
                  |  |              +--rw sid?   rt-types:generalized-
   label
                  |  +--rw backup-path* [index]
                  |  |  +--rw index           uint32
                  |  |  +--rw network-ref?    leafref
                  |  |  +--rw path-element* [path-element-id]
                  |  |     +--rw path-element-id    uint32
                  |  |     +--rw index?             uint32
                  |  |     +--rw (type)?
                  |  |        +--:(ip-address)
                  |  |        |  +--rw ip-address-hop
                  |  |        |     +--rw address?    inet:ip-address
                  |  |        |     +--rw hop-type?   te-hop-type
                  |  |        +--:(as-number)
                  |  |        |  +--rw as-number-hop
                  |  |        |     +--rw as-number?   binary
                  |  |        |     +--rw hop-type?    te-hop-type
                  |  |        +--:(unnumbered-link)
                  |  |        |  +--rw unnumbered-hop
                  |  |        |     +--rw router-id?      inet:ip-
   address
                  |  |        |     +--rw interface-id?   uint32
                  |  |        |     +--rw hop-type?       te-hop-type
                  |  |        +--:(label)
                  |  |        |  +--rw label-hop
```

```
                |  |        |     +--rw value?   rt-types:generalized-
   label
                |  |        +--:(sid)
                |  |           +--rw sid-hop
                |  |              +--rw sid?   rt-types:generalized-
   label
                |  +--rw protection-type?   uint16
                |  +--rw tunnels
                |     +--rw sharing?   boolean
                |     +--rw tunnel* [tunnel-name]
                |        +--rw tunnel-name    string
                |        +--rw sharing?       boolean
                +--rw admin-status?                      te-types:te-
   admin-status
                +--rw link-index?                        uint64
                +--rw administrative-group?              te-
   types:admin-groups
                +--rw interface-switching-capability* [switching-
   capability encoding]
                |  +--rw switching-capability    identityref
                |  +--rw encoding                identityref
                |  +--rw max-lsp-bandwidth* [priority]
                |     +--rw priority     uint8
                |     +--rw bandwidth?   te-bandwidth
                +--rw link-protection-type?              enumeration
                +--rw max-link-bandwidth?                te-bandwidth
                +--rw max-resv-link-bandwidth?           te-bandwidth
                +--rw unreserved-bandwidth* [priority]
                |  +--rw priority     uint8
                |  +--rw bandwidth?   te-bandwidth
                +--rw te-default-metric?                 uint32
                +--rw te-delay-metric?                   uint32
                +--rw te-srlgs
                |  +--rw value*   te-types:srlg
                +--rw te-nsrlgs {nsrlg}?
                   +--rw id*   uint32
   augment /nw:networks/nw:network:
      +--rw provider-id?      te-types:te-global-id
      +--rw client-id?        te-types:te-global-id
      +--rw te-topology-id?   te-types:te-topology-id
      +--rw te!
         +--rw config
         |  +--rw preference?               uint8
         |  +--rw optimization-criterion?   identityref
         |  +--rw nsrlg* [id] {nsrlg}?
         |     +--rw id              uint32
```

```
         |     +--rw disjointness?   te-path-disjointness
         +--ro state
            +--ro preference?               uint8
            +--ro optimization-criterion?   identityref
            +--ro nsrlg* [id] {nsrlg}?
            |  +--ro id              uint32
            |  +--ro disjointness?   te-path-disjointness
            +--ro geolocation
               +--ro altitude?    int64
               +--ro latitude?    geographic-coordinate-degree
               +--ro longitude?   geographic-coordinate-degree
   augment /nw:networks/nw:network/nw:node:
      +--rw te-node-id?   te-types:te-node-id
      +--rw te!
         +--rw config
         |  +--rw te-node-template*     leafref {template}?
         |  +--rw te-node-attributes
         |     +--rw admin-status?            te-types:te-admin-status
         |     +--rw connectivity-matrices
         |     |  +--rw number-of-entries?         uint16
         |     |  +--rw is-allowed?                boolean
         |     |  +--rw label-restriction* [inclusive-exclusive label-
   start]
         |     |  |  +--rw inclusive-exclusive    enumeration
         |     |  |  +--rw label-start            rt-types:generalized-
   label
         |     |  |  +--rw label-end?             rt-types:generalized-
   label
         |     |  |  +--rw range-bitmap?          binary
         |     |  +--rw underlay! {te-topology-hierarchy}?
         |     |  |  +--rw primary-path
         |     |  |  |  +--rw network-ref?    leafref
         |     |  |  |  +--rw path-element* [path-element-id]
         |     |  |  |     +--rw path-element-id    uint32
         |     |  |  |     +--rw index?             uint32
         |     |  |  |     +--rw (type)?
         |     |  |  |        +--:(ip-address)
         |     |  |  |        |  +--rw ip-address-hop
         |     |  |  |        |     +--rw address?    inet:ip-address
         |     |  |  |        |     +--rw hop-type?   te-hop-type
         |     |  |  |        +--:(as-number)
         |     |  |  |        |  +--rw as-number-hop
         |     |  |  |        |     +--rw as-number?   binary
         |     |  |  |        |     +--rw hop-type?    te-hop-type
         |     |  |  |        +--:(unnumbered-link)
         |     |  |  |        |  +--rw unnumbered-hop
```

```
      |     |  |  |        |     +--rw router-id?      inet:ip-
   address
      |     |  |  |        |     +--rw interface-id?   uint32
      |     |  |  |        |     +--rw hop-type?       te-hop-type
      |     |  |  |        +--:(label)
      |     |  |  |        |  +--rw label-hop
      |     |  |  |        |     +--rw value?   rt-types:generalized-
   label
      |     |  |  |        +--:(sid)
      |     |  |  |           +--rw sid-hop
      |     |  |  |              +--rw sid?   rt-types:generalized-
   label
      |     |  |  +--rw backup-path* [index]
      |     |  |  |  +--rw index           uint32
      |     |  |  |  +--rw network-ref?    leafref
      |     |  |  |  +--rw path-element* [path-element-id]
      |     |  |  |     +--rw path-element-id    uint32
      |     |  |  |     +--rw index?             uint32
      |     |  |  |     +--rw (type)?
      |     |  |  |        +--:(ip-address)
      |     |  |  |        |  +--rw ip-address-hop
      |     |  |  |        |     +--rw address?    inet:ip-address
      |     |  |  |        |     +--rw hop-type?   te-hop-type
      |     |  |  |        +--:(as-number)
      |     |  |  |        |  +--rw as-number-hop
      |     |  |  |        |     +--rw as-number?   binary
      |     |  |  |        |     +--rw hop-type?    te-hop-type
      |     |  |  |        +--:(unnumbered-link)
      |     |  |  |        |  +--rw unnumbered-hop
      |     |  |  |        |     +--rw router-id?      inet:ip-
   address
      |     |  |  |        |     +--rw interface-id?   uint32
      |     |  |  |        |     +--rw hop-type?       te-hop-type
      |     |  |  |        +--:(label)
      |     |  |  |        |  +--rw label-hop
      |     |  |  |        |     +--rw value?   rt-types:generalized-
   label
      |     |  |  |        +--:(sid)
      |     |  |  |           +--rw sid-hop
      |     |  |  |              +--rw sid?   rt-types:generalized-
   label
      |     |  |  +--rw protection-type?   uint16
      |     |  |  +--rw tunnels
      |     |  |     +--rw sharing?   boolean
      |     |  |     +--rw tunnel* [tunnel-name]
      |     |  |        +--rw tunnel-name    string
```

```
        |     |  |        +--rw sharing?       boolean
        |     |  +--rw max-lsp-bandwidth* [priority]
        |     |  |  +--rw priority     uint8
        |     |  |  +--rw bandwidth?   te-bandwidth
        |     |  +--rw max-link-bandwidth?        te-bandwidth
        |     |  +--rw max-resv-link-bandwidth?   te-bandwidth
        |     |  +--rw unreserved-bandwidth* [priority]
        |     |  |  +--rw priority     uint8
        |     |  |  +--rw bandwidth?   te-bandwidth
        |     |  +--rw te-default-metric?         uint32
        |     |  +--rw te-delay-metric?           uint32
        |     |  +--rw te-srlgs
        |     |  |  +--rw value*   te-types:srlg
        |     |  +--rw te-nsrlgs {nsrlg}?
        |     |  |  +--rw id*   uint32
        |     |  +--rw connectivity-matrix* [id]
        |     |     +--rw id                          uint32
        |     |     +--rw from
        |     |     |  +--rw tp-ref?   leafref
        |     |     +--rw to
        |     |     |  +--rw tp-ref?   leafref
        |     |     +--rw is-allowed?                 boolean
        |     |     +--rw label-restriction* [inclusive-exclusive
   label-start]
        |     |     |  +--rw inclusive-exclusive    enumeration
        |     |     |  +--rw label-start            rt-
   types:generalized-label
        |     |     |  +--rw label-end?             rt-
   types:generalized-label
        |     |     |  +--rw range-bitmap?          binary
        |     |     +--rw underlay! {te-topology-hierarchy}?
        |     |     |  +--rw primary-path
        |     |     |  |  +--rw network-ref?    leafref
        |     |     |  |  +--rw path-element* [path-element-id]
        |     |     |  |     +--rw path-element-id    uint32
        |     |     |  |     +--rw index?             uint32
        |     |     |  |     +--rw (type)?
        |     |     |  |        +--:(ip-address)
        |     |     |  |        |  +--rw ip-address-hop
        |     |     |  |        |     +--rw address?    inet:ip-address
        |     |     |  |        |     +--rw hop-type?   te-hop-type
        |     |     |  |        +--:(as-number)
        |     |     |  |        |  +--rw as-number-hop
        |     |     |  |        |     +--rw as-number?   binary
        |     |     |  |        |     +--rw hop-type?    te-hop-type
        |     |     |  |        +--:(unnumbered-link)
```

```
        |     |     |  |        |  +--rw unnumbered-hop
        |     |     |  |        |     +--rw router-id?      inet:ip-
   address
        |     |     |  |        |     +--rw interface-id?   uint32
        |     |     |  |        |     +--rw hop-type?       te-hop-type
        |     |     |  |        +--:(label)
        |     |     |  |        |  +--rw label-hop
        |     |     |  |        |     +--rw value?   rt-
   types:generalized-label
        |     |     |  |        +--:(sid)
        |     |     |  |           +--rw sid-hop
        |     |     |  |              +--rw sid?   rt-
   types:generalized-label
        |     |     |  +--rw backup-path* [index]
        |     |     |  |  +--rw index           uint32
        |     |     |  |  +--rw network-ref?    leafref
        |     |     |  |  +--rw path-element* [path-element-id]
        |     |     |  |     +--rw path-element-id    uint32
        |     |     |  |     +--rw index?             uint32
        |     |     |  |     +--rw (type)?
        |     |     |  |        +--:(ip-address)
        |     |     |  |        |  +--rw ip-address-hop
        |     |     |  |        |     +--rw address?    inet:ip-address
        |     |     |  |        |     +--rw hop-type?   te-hop-type
        |     |     |  |        +--:(as-number)
        |     |     |  |        |  +--rw as-number-hop
        |     |     |  |        |     +--rw as-number?   binary
        |     |     |  |        |     +--rw hop-type?    te-hop-type
        |     |     |  |        +--:(unnumbered-link)
        |     |     |  |        |  +--rw unnumbered-hop
        |     |     |  |        |     +--rw router-id?      inet:ip-
   address
        |     |     |  |        |     +--rw interface-id?   uint32
        |     |     |  |        |     +--rw hop-type?       te-hop-type
        |     |     |  |        +--:(label)
        |     |     |  |        |  +--rw label-hop
        |     |     |  |        |     +--rw value?   rt-
   types:generalized-label
        |     |     |  |        +--:(sid)
        |     |     |  |           +--rw sid-hop
        |     |     |  |              +--rw sid?   rt-
   types:generalized-label
        |     |     |  +--rw protection-type?   uint16
        |     |     |  +--rw tunnels
        |     |     |     +--rw sharing?   boolean
        |     |     |     +--rw tunnel* [tunnel-name]
```

```
         |     |     |        +--rw tunnel-name    string
         |     |     |        +--rw sharing?       boolean
         |     |     +--rw max-lsp-bandwidth* [priority]
         |     |     |  +--rw priority     uint8
         |     |     |  +--rw bandwidth?   te-bandwidth
         |     |     +--rw max-link-bandwidth?        te-bandwidth
         |     |     +--rw max-resv-link-bandwidth?   te-bandwidth
         |     |     +--rw unreserved-bandwidth* [priority]
         |     |     |  +--rw priority     uint8
         |     |     |  +--rw bandwidth?   te-bandwidth
         |     |     +--rw te-default-metric?         uint32
         |     |     +--rw te-delay-metric?           uint32
         |     |     +--rw te-srlgs
         |     |     |  +--rw value*   te-types:srlg
         |     |     +--rw te-nsrlgs {nsrlg}?
         |     |        +--rw id*   uint32
         |     +--rw domain-id?               uint32
         |     +--rw is-abstract?             empty
         |     +--rw name?                    inet:domain-name
         |     +--rw signaling-address*       inet:ip-address
         |     +--rw underlay-topology {te-topology-hierarchy}?
         |        +--rw network-ref?   leafref
         +--ro state
         |  +--ro te-node-template*           leafref {template}?
         |  +--ro te-node-attributes
         |  |  +--ro admin-status?            te-types:te-admin-status
         |  |  +--ro connectivity-matrices
         |  |  |  +--ro number-of-entries?         uint16
         |  |  |  +--ro is-allowed?                boolean
         |  |  |  +--ro label-restriction* [inclusive-exclusive label-
   start]
         |  |  |  |  +--ro inclusive-exclusive    enumeration
         |  |  |  |  +--ro label-start            rt-types:generalized-
   label
         |  |  |  |  +--ro label-end?             rt-types:generalized-
   label
         |  |  |  |  +--ro range-bitmap?          binary
         |  |  |  +--ro underlay! {te-topology-hierarchy}?
         |  |  |  |  +--ro primary-path
         |  |  |  |  |  +--ro network-ref?    leafref
         |  |  |  |  |  +--ro path-element* [path-element-id]
         |  |  |  |  |     +--ro path-element-id    uint32
         |  |  |  |  |     +--ro index?             uint32
         |  |  |  |  |     +--ro (type)?
         |  |  |  |  |        +--:(ip-address)
         |  |  |  |  |        |  +--ro ip-address-hop
```

```
          |  |  |  |  |        |     +--ro address?    inet:ip-address
          |  |  |  |  |        |     +--ro hop-type?   te-hop-type
          |  |  |  |  |        +--:(as-number)
          |  |  |  |  |        |  +--ro as-number-hop
          |  |  |  |  |        |     +--ro as-number?   binary
          |  |  |  |  |        |     +--ro hop-type?    te-hop-type
          |  |  |  |  |        +--:(unnumbered-link)
          |  |  |  |  |        |  +--ro unnumbered-hop
          |  |  |  |  |        |     +--ro router-id?      inet:ip-
    address
          |  |  |  |  |        |     +--ro interface-id?   uint32
          |  |  |  |  |        |     +--ro hop-type?       te-hop-type
          |  |  |  |  |        +--:(label)
          |  |  |  |  |        |  +--ro label-hop
          |  |  |  |  |        |     +--ro value?   rt-types:generalized-
    label
          |  |  |  |  |        +--:(sid)
          |  |  |  |  |           +--ro sid-hop
          |  |  |  |  |              +--ro sid?   rt-types:generalized-
    label
          |  |  |  |  +--ro backup-path* [index]
          |  |  |  |  |  +--ro index           uint32
          |  |  |  |  |  +--ro network-ref?    leafref
          |  |  |  |  |  +--ro path-element* [path-element-id]
          |  |  |  |  |     +--ro path-element-id    uint32
          |  |  |  |  |     +--ro index?             uint32
          |  |  |  |  |     +--ro (type)?
          |  |  |  |  |        +--:(ip-address)
          |  |  |  |  |        |  +--ro ip-address-hop
          |  |  |  |  |        |     +--ro address?    inet:ip-address
          |  |  |  |  |        |     +--ro hop-type?   te-hop-type
          |  |  |  |  |        +--:(as-number)
          |  |  |  |  |        |  +--ro as-number-hop
          |  |  |  |  |        |     +--ro as-number?   binary
          |  |  |  |  |        |     +--ro hop-type?    te-hop-type
          |  |  |  |  |        +--:(unnumbered-link)
          |  |  |  |  |        |  +--ro unnumbered-hop
          |  |  |  |  |        |     +--ro router-id?      inet:ip-
    address
          |  |  |  |  |        |     +--ro interface-id?   uint32
          |  |  |  |  |        |     +--ro hop-type?       te-hop-type
          |  |  |  |  |        +--:(label)
          |  |  |  |  |        |  +--ro label-hop
          |  |  |  |  |        |     +--ro value?   rt-types:generalized-
    label
          |  |  |  |  |        +--:(sid)
```

```
        |  |  |  |  |          +--ro sid-hop
        |  |  |  |  |             +--ro sid?   rt-types:generalized-
   label
        |  |  |  |  +--ro protection-type?   uint16
        |  |  |  |  +--ro tunnels
        |  |  |  |     +--ro sharing?   boolean
        |  |  |  |     +--ro tunnel* [tunnel-name]
        |  |  |  |        +--ro tunnel-name    string
        |  |  |  |        +--ro sharing?       boolean
        |  |  |  +--ro max-lsp-bandwidth* [priority]
        |  |  |  |  +--ro priority     uint8
        |  |  |  |  +--ro bandwidth?   te-bandwidth
        |  |  |  +--ro max-link-bandwidth?        te-bandwidth
        |  |  |  +--ro max-resv-link-bandwidth?   te-bandwidth
        |  |  |  +--ro unreserved-bandwidth* [priority]
        |  |  |  |  +--ro priority     uint8
        |  |  |  |  +--ro bandwidth?   te-bandwidth
        |  |  |  +--ro te-default-metric?         uint32
        |  |  |  +--ro te-delay-metric?           uint32
        |  |  |  +--ro te-srlgs
        |  |  |  |  +--ro value*   te-types:srlg
        |  |  |  +--ro te-nsrlgs {nsrlg}?
        |  |  |  |  +--ro id*   uint32
        |  |  |  +--ro connectivity-matrix* [id]
        |  |  |     +--ro id                          uint32
        |  |  |     +--ro from
        |  |  |     |  +--ro tp-ref?   leafref
        |  |  |     +--ro to
        |  |  |     |  +--ro tp-ref?   leafref
        |  |  |     +--ro is-allowed?                boolean
        |  |  |     +--ro label-restriction* [inclusive-exclusive
   label-start]
        |  |  |     |  +--ro inclusive-exclusive    enumeration
        |  |  |     |  +--ro label-start            rt-
   types:generalized-label
        |  |  |     |  +--ro label-end?             rt-
   types:generalized-label
        |  |  |     |  +--ro range-bitmap?          binary
        |  |  |     +--ro underlay! {te-topology-hierarchy}?
        |  |  |     |  +--ro primary-path
        |  |  |     |  |  +--ro network-ref?    leafref
        |  |  |     |  |  +--ro path-element* [path-element-id]
        |  |  |     |  |     +--ro path-element-id    uint32
        |  |  |     |  |     +--ro index?             uint32
        |  |  |     |  |     +--ro (type)?
        |  |  |     |  |        +--:(ip-address)
```

```
        |  |  |     |  |        |  +--ro ip-address-hop
        |  |  |     |  |        |     +--ro address?    inet:ip-address
        |  |  |     |  |        |     +--ro hop-type?   te-hop-type
        |  |  |     |  |        +--:(as-number)
        |  |  |     |  |        |  +--ro as-number-hop
        |  |  |     |  |        |     +--ro as-number?   binary
        |  |  |     |  |        |     +--ro hop-type?    te-hop-type
        |  |  |     |  |        +--:(unnumbered-link)
        |  |  |     |  |        |  +--ro unnumbered-hop
        |  |  |     |  |        |     +--ro router-id?      inet:ip-
   address
        |  |  |     |  |        |     +--ro interface-id?   uint32
        |  |  |     |  |        |     +--ro hop-type?       te-hop-type
        |  |  |     |  |        +--:(label)
        |  |  |     |  |        |  +--ro label-hop
        |  |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
        |  |  |     |  |        +--:(sid)
        |  |  |     |  |           +--ro sid-hop
        |  |  |     |  |              +--ro sid?   rt-
   types:generalized-label
        |  |  |     |  +--ro backup-path* [index]
        |  |  |     |  |  +--ro index           uint32
        |  |  |     |  |  +--ro network-ref?    leafref
        |  |  |     |  |  +--ro path-element* [path-element-id]
        |  |  |     |  |     +--ro path-element-id    uint32
        |  |  |     |  |     +--ro index?             uint32
        |  |  |     |  |     +--ro (type)?
        |  |  |     |  |        +--:(ip-address)
        |  |  |     |  |        |  +--ro ip-address-hop
        |  |  |     |  |        |     +--ro address?    inet:ip-address
        |  |  |     |  |        |     +--ro hop-type?   te-hop-type
        |  |  |     |  |        +--:(as-number)
        |  |  |     |  |        |  +--ro as-number-hop
        |  |  |     |  |        |     +--ro as-number?   binary
        |  |  |     |  |        |     +--ro hop-type?    te-hop-type
        |  |  |     |  |        +--:(unnumbered-link)
        |  |  |     |  |        |  +--ro unnumbered-hop
        |  |  |     |  |        |     +--ro router-id?      inet:ip-
   address
        |  |  |     |  |        |     +--ro interface-id?   uint32
        |  |  |     |  |        |     +--ro hop-type?       te-hop-type
        |  |  |     |  |        +--:(label)
        |  |  |     |  |        |  +--ro label-hop
        |  |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
```

```
         |  |  |     |  |       +--:(sid)
         |  |  |     |  |          +--ro sid-hop
         |  |  |     |  |             +--ro sid?   rt-
   types:generalized-label
         |  |  |     |  +--ro protection-type?   uint16
         |  |  |     |  +--ro tunnels
         |  |  |     |     +--ro sharing?   boolean
         |  |  |     |     +--ro tunnel* [tunnel-name]
         |  |  |     |        +--ro tunnel-name    string
         |  |  |     |        +--ro sharing?       boolean
         |  |  |     +--ro max-lsp-bandwidth* [priority]
         |  |  |     |  +--ro priority     uint8
         |  |  |     |  +--ro bandwidth?   te-bandwidth
         |  |  |     +--ro max-link-bandwidth?        te-bandwidth
         |  |  |     +--ro max-resv-link-bandwidth?   te-bandwidth
         |  |  |     +--ro unreserved-bandwidth* [priority]
         |  |  |     |  +--ro priority     uint8
         |  |  |     |  +--ro bandwidth?   te-bandwidth
         |  |  |     +--ro te-default-metric?         uint32
         |  |  |     +--ro te-delay-metric?           uint32
         |  |  |     +--ro te-srlgs
         |  |  |     |  +--ro value*   te-types:srlg
         |  |  |     +--ro te-nsrlgs {nsrlg}?
         |  |  |        +--ro id*   uint32
         |  |  +--ro domain-id?              uint32
         |  |  +--ro is-abstract?            empty
         |  |  +--ro name?                   inet:domain-name
         |  |  +--ro signaling-address*      inet:ip-address
         |  |  +--ro underlay-topology {te-topology-hierarchy}?
         |  |     +--ro network-ref?   leafref
         |  +--ro oper-status?               te-types:te-oper-status
         |  +--ro geolocation
         |  |  +--ro altitude?    int64
         |  |  +--ro latitude?    geographic-coordinate-degree
         |  |  +--ro longitude?   geographic-coordinate-degree
         |  +--ro is-multi-access-dr?         empty
         |  +--ro information-source?         te-info-source
         |  +--ro information-source-state
         |  |  +--ro credibility-preference?    uint16
         |  |  +--ro logical-network-element?   string
         |  |  +--ro network-instance?          string
         |  |  +--ro topology
         |  |     +--ro network-ref?   leafref
         |  |     +--ro node-ref?      leafref
         |  +--ro information-source-entry* [information-source]
         |     +--ro information-source          te-info-source
```

```
        |     +--ro information-source-state
        |     |  +--ro credibility-preference?    uint16
        |     |  +--ro logical-network-element?   string
        |     |  +--ro network-instance?          string
        |     |  +--ro topology
        |     |     +--ro network-ref?   leafref
        |     |     +--ro node-ref?      leafref
        |     +--ro connectivity-matrices
        |     |  +--ro number-of-entries?         uint16
        |     |  +--ro is-allowed?                boolean
        |     |  +--ro label-restriction* [inclusive-exclusive label-
   start]
        |     |  |  +--ro inclusive-exclusive    enumeration
        |     |  |  +--ro label-start            rt-types:generalized-
   label
        |     |  |  +--ro label-end?             rt-types:generalized-
   label
        |     |  |  +--ro range-bitmap?          binary
        |     |  +--ro underlay! {te-topology-hierarchy}?
        |     |  |  +--ro primary-path
        |     |  |  |  +--ro network-ref?    leafref
        |     |  |  |  +--ro path-element* [path-element-id]
        |     |  |  |     +--ro path-element-id    uint32
        |     |  |  |     +--ro index?             uint32
        |     |  |  |     +--ro (type)?
        |     |  |  |        +--:(ip-address)
        |     |  |  |        |  +--ro ip-address-hop
        |     |  |  |        |     +--ro address?    inet:ip-address
        |     |  |  |        |     +--ro hop-type?   te-hop-type
        |     |  |  |        +--:(as-number)
        |     |  |  |        |  +--ro as-number-hop
        |     |  |  |        |     +--ro as-number?   binary
        |     |  |  |        |     +--ro hop-type?    te-hop-type
        |     |  |  |        +--:(unnumbered-link)
        |     |  |  |        |  +--ro unnumbered-hop
        |     |  |  |        |     +--ro router-id?      inet:ip-
   address
        |     |  |  |        |     +--ro interface-id?   uint32
        |     |  |  |        |     +--ro hop-type?       te-hop-type
        |     |  |  |        +--:(label)
        |     |  |  |        |  +--ro label-hop
        |     |  |  |        |     +--ro value?   rt-types:generalized-
   label
        |     |  |  |        +--:(sid)
        |     |  |  |           +--ro sid-hop
```

```
        |     |  |  |             +--ro sid?   rt-types:generalized-
   label
        |     |  |  +--ro backup-path* [index]
        |     |  |  |  +--ro index           uint32
        |     |  |  |  +--ro network-ref?    leafref
        |     |  |  |  +--ro path-element* [path-element-id]
        |     |  |  |     +--ro path-element-id    uint32
        |     |  |  |     +--ro index?             uint32
        |     |  |  |     +--ro (type)?
        |     |  |  |        +--:(ip-address)
        |     |  |  |        |  +--ro ip-address-hop
        |     |  |  |        |     +--ro address?    inet:ip-address
        |     |  |  |        |     +--ro hop-type?   te-hop-type
        |     |  |  |        +--:(as-number)
        |     |  |  |        |  +--ro as-number-hop
        |     |  |  |        |     +--ro as-number?   binary
        |     |  |  |        |     +--ro hop-type?    te-hop-type
        |     |  |  |        +--:(unnumbered-link)
        |     |  |  |        |  +--ro unnumbered-hop
        |     |  |  |        |     +--ro router-id?      inet:ip-
   address
        |     |  |  |        |     +--ro interface-id?   uint32
        |     |  |  |        |     +--ro hop-type?       te-hop-type
        |     |  |  |        +--:(label)
        |     |  |  |        |  +--ro label-hop
        |     |  |  |        |     +--ro value?   rt-types:generalized-
   label
        |     |  |  |        +--:(sid)
        |     |  |  |           +--ro sid-hop
        |     |  |  |              +--ro sid?   rt-types:generalized-
   label
        |     |  |  +--ro protection-type?   uint16
        |     |  |  +--ro tunnels
        |     |  |     +--ro sharing?   boolean
        |     |  |     +--ro tunnel* [tunnel-name]
        |     |  |        +--ro tunnel-name    string
        |     |  |        +--ro sharing?       boolean
        |     |  +--ro max-lsp-bandwidth* [priority]
        |     |  |  +--ro priority     uint8
        |     |  |  +--ro bandwidth?   te-bandwidth
        |     |  +--ro max-link-bandwidth?        te-bandwidth
        |     |  +--ro max-resv-link-bandwidth?   te-bandwidth
        |     |  +--ro unreserved-bandwidth* [priority]
        |     |  |  +--ro priority     uint8
        |     |  |  +--ro bandwidth?   te-bandwidth
        |     |  +--ro te-default-metric?         uint32
```

```
        |     |  +--ro te-delay-metric?          uint32
        |     |  +--ro te-srlgs
        |     |  |  +--ro value*   te-types:srlg
        |     |  +--ro te-nsrlgs {nsrlg}?
        |     |  |  +--ro id*   uint32
        |     |  +--ro connectivity-matrix* [id]
        |     |     +--ro id                         uint32
        |     |     +--ro from
        |     |     |  +--ro tp-ref?   leafref
        |     |     +--ro to
        |     |     |  +--ro tp-ref?   leafref
        |     |     +--ro is-allowed?                boolean
        |     |     +--ro label-restriction* [inclusive-exclusive
   label-start]
        |     |     |  +--ro inclusive-exclusive    enumeration
        |     |     |  +--ro label-start            rt-
   types:generalized-label
        |     |     |  +--ro label-end?             rt-
   types:generalized-label
        |     |     |  +--ro range-bitmap?          binary
        |     |     +--ro underlay! {te-topology-hierarchy}?
        |     |     |  +--ro primary-path
        |     |     |  |  +--ro network-ref?    leafref
        |     |     |  |  +--ro path-element* [path-element-id]
        |     |     |  |     +--ro path-element-id    uint32
        |     |     |  |     +--ro index?             uint32
        |     |     |  |     +--ro (type)?
        |     |     |  |        +--:(ip-address)
        |     |     |  |        |  +--ro ip-address-hop
        |     |     |  |        |     +--ro address?    inet:ip-address
        |     |     |  |        |     +--ro hop-type?   te-hop-type
        |     |     |  |        +--:(as-number)
        |     |     |  |        |  +--ro as-number-hop
        |     |     |  |        |     +--ro as-number?   binary
        |     |     |  |        |     +--ro hop-type?    te-hop-type
        |     |     |  |        +--:(unnumbered-link)
        |     |     |  |        |  +--ro unnumbered-hop
        |     |     |  |        |     +--ro router-id?      inet:ip-
   address
        |     |     |  |        |     +--ro interface-id?   uint32
        |     |     |  |        |     +--ro hop-type?       te-hop-type
        |     |     |  |        +--:(label)
        |     |     |  |        |  +--ro label-hop
        |     |     |  |        |     +--ro value?   rt-
   types:generalized-label
        |     |     |  |        +--:(sid)
```

```
        |     |     |  |           +--ro sid-hop
        |     |     |  |              +--ro sid?   rt-
   types:generalized-label
        |     |     |  +--ro backup-path* [index]
        |     |     |  |  +--ro index           uint32
        |     |     |  |  +--ro network-ref?    leafref
        |     |     |  |  +--ro path-element* [path-element-id]
        |     |     |  |     +--ro path-element-id    uint32
        |     |     |  |     +--ro index?             uint32
        |     |     |  |     +--ro (type)?
        |     |     |  |        +--:(ip-address)
        |     |     |  |        |  +--ro ip-address-hop
        |     |     |  |        |     +--ro address?    inet:ip-address
        |     |     |  |        |     +--ro hop-type?   te-hop-type
        |     |     |  |        +--:(as-number)
        |     |     |  |        |  +--ro as-number-hop
        |     |     |  |        |     +--ro as-number?   binary
        |     |     |  |        |     +--ro hop-type?    te-hop-type
        |     |     |  |        +--:(unnumbered-link)
        |     |     |  |        |  +--ro unnumbered-hop
        |     |     |  |        |     +--ro router-id?      inet:ip-
   address
        |     |     |  |        |     +--ro interface-id?   uint32
        |     |     |  |        |     +--ro hop-type?       te-hop-type
        |     |     |  |        +--:(label)
        |     |     |  |        |  +--ro label-hop
        |     |     |  |        |     +--ro value?   rt-
   types:generalized-label
        |     |     |  |        +--:(sid)
        |     |     |  |           +--ro sid-hop
        |     |     |  |              +--ro sid?   rt-
   types:generalized-label
        |     |     |  +--ro protection-type?   uint16
        |     |     |  +--ro tunnels
        |     |     |     +--ro sharing?   boolean
        |     |     |     +--ro tunnel* [tunnel-name]
        |     |     |        +--ro tunnel-name    string
        |     |     |        +--ro sharing?       boolean
        |     |     +--ro max-lsp-bandwidth* [priority]
        |     |     |  +--ro priority     uint8
        |     |     |  +--ro bandwidth?   te-bandwidth
        |     |     +--ro max-link-bandwidth?        te-bandwidth
        |     |     +--ro max-resv-link-bandwidth?   te-bandwidth
        |     |     +--ro unreserved-bandwidth* [priority]
        |     |     |  +--ro priority     uint8
        |     |     |  +--ro bandwidth?   te-bandwidth
```

```
         |     |     +--ro te-default-metric?         uint32
         |     |     +--ro te-delay-metric?           uint32
         |     |     +--ro te-srlgs
         |     |     |  +--ro value*   te-types:srlg
         |     |     +--ro te-nsrlgs {nsrlg}?
         |     |        +--ro id*   uint32
         |     +--ro domain-id?                 uint32
         |     +--ro is-abstract?               empty
         |     +--ro name?                      inet:domain-name
         |     +--ro signaling-address*         inet:ip-address
         |     +--ro underlay-topology {te-topology-hierarchy}?
         |        +--ro network-ref?   leafref
        +--ro statistics
         |  +--ro discontinuity-time           yang:date-and-time
         |  +--ro node
         |  |  +--ro disables?             yang:counter32
         |  |  +--ro enables?              yang:counter32
         |  |  +--ro maintenance-sets?     yang:counter32
         |  |  +--ro maintenance-clears?   yang:counter32
         |  |  +--ro modifies?             yang:counter32
         |  +--ro connectivity-matrix-entry
         |     +--ro creates?    yang:counter32
         |     +--ro deletes?    yang:counter32
         |     +--ro disables?   yang:counter32
         |     +--ro enables?    yang:counter32
         |     +--ro modifies?   yang:counter32
        +--rw tunnel-termination-point* [tunnel-tp-id]
           +--rw tunnel-tp-id                          binary
           +--rw config
           |  +--rw switching-capability?        identityref
           |  +--rw encoding?                    identityref
           |  +--rw inter-layer-lock-id?         uint32
           |  +--rw protection-type?             identityref
           |  +--rw client-layer-adaptation
           |  |  +--rw switching-capability* [switching-capability
   encoding]
           |  |     +--rw switching-capability    identityref
           |  |     +--rw encoding                identityref
           |  |     +--rw bandwidth?              te-bandwidth
           |  +--rw local-link-connectivities
           |     +--rw number-of-entries?         uint16
           |     +--rw is-allowed?                boolean
           |     +--rw label-restriction* [inclusive-exclusive label-
   start]
           |     |  +--rw inclusive-exclusive    enumeration
```

```
          |     |  +--rw label-start           rt-types:generalized-
   label
          |     |  +--rw label-end?            rt-types:generalized-
   label
          |     |  +--rw range-bitmap?         binary
          |     +--rw underlay! {te-topology-hierarchy}?
          |     |  +--rw primary-path
          |     |  |  +--rw network-ref?    leafref
          |     |  |  +--rw path-element* [path-element-id]
          |     |  |     +--rw path-element-id    uint32
          |     |  |     +--rw index?             uint32
          |     |  |     +--rw (type)?
          |     |  |        +--:(ip-address)
          |     |  |        |  +--rw ip-address-hop
          |     |  |        |     +--rw address?    inet:ip-address
          |     |  |        |     +--rw hop-type?   te-hop-type
          |     |  |        +--:(as-number)
          |     |  |        |  +--rw as-number-hop
          |     |  |        |     +--rw as-number?   binary
          |     |  |        |     +--rw hop-type?    te-hop-type
          |     |  |        +--:(unnumbered-link)
          |     |  |        |  +--rw unnumbered-hop
          |     |  |        |     +--rw router-id?      inet:ip-
   address
          |     |  |        |     +--rw interface-id?   uint32
          |     |  |        |     +--rw hop-type?       te-hop-type
          |     |  |        +--:(label)
          |     |  |        |  +--rw label-hop
          |     |  |        |     +--rw value?   rt-types:generalized-
   label
          |     |  |        +--:(sid)
          |     |  |           +--rw sid-hop
          |     |  |              +--rw sid?   rt-types:generalized-
   label
          |     |  +--rw backup-path* [index]
          |     |  |  +--rw index           uint32
          |     |  |  +--rw network-ref?    leafref
          |     |  |  +--rw path-element* [path-element-id]
          |     |  |     +--rw path-element-id    uint32
          |     |  |     +--rw index?             uint32
          |     |  |     +--rw (type)?
          |     |  |        +--:(ip-address)
          |     |  |        |  +--rw ip-address-hop
          |     |  |        |     +--rw address?    inet:ip-address
          |     |  |        |     +--rw hop-type?   te-hop-type
          |     |  |        +--:(as-number)
```

```
          |     |  |        |  +--rw as-number-hop
          |     |  |        |     +--rw as-number?   binary
          |     |  |        |     +--rw hop-type?    te-hop-type
          |     |  |        +--:(unnumbered-link)
          |     |  |        |  +--rw unnumbered-hop
          |     |  |        |     +--rw router-id?      inet:ip-
   address
          |     |  |        |     +--rw interface-id?   uint32
          |     |  |        |     +--rw hop-type?       te-hop-type
          |     |  |        +--:(label)
          |     |  |        |  +--rw label-hop
          |     |  |        |     +--rw value?   rt-types:generalized-
   label
          |     |  |        +--:(sid)
          |     |  |           +--rw sid-hop
          |     |  |              +--rw sid?   rt-types:generalized-
   label
          |     |  +--rw protection-type?   uint16
          |     |  +--rw tunnels
          |     |     +--rw sharing?   boolean
          |     |     +--rw tunnel* [tunnel-name]
          |     |        +--rw tunnel-name    string
          |     |        +--rw sharing?       boolean
          |     +--rw max-lsp-bandwidth* [priority]
          |     |  +--rw priority     uint8
          |     |  +--rw bandwidth?   te-bandwidth
          |     +--rw max-link-bandwidth?        te-bandwidth
          |     +--rw max-resv-link-bandwidth?   te-bandwidth
          |     +--rw unreserved-bandwidth* [priority]
          |     |  +--rw priority     uint8
          |     |  +--rw bandwidth?   te-bandwidth
          |     +--rw te-default-metric?         uint32
          |     +--rw te-delay-metric?           uint32
          |     +--rw te-srlgs
          |     |  +--rw value*   te-types:srlg
          |     +--rw te-nsrlgs {nsrlg}?
          |     |  +--rw id*   uint32
          |     +--rw local-link-connectivity* [link-tp-ref]
          |        +--rw link-tp-ref                leafref
          |        +--rw is-allowed?                boolean
          |        +--rw label-restriction* [inclusive-exclusive
   label-start]
          |        |  +--rw inclusive-exclusive    enumeration
          |        |  +--rw label-start            rt-
   types:generalized-label
```

```
          |        |  +--rw label-end?            rt-
   types:generalized-label
          |        |  +--rw range-bitmap?         binary
          |        +--rw underlay! {te-topology-hierarchy}?
          |        |  +--rw primary-path
          |        |  |  +--rw network-ref?    leafref
          |        |  |  +--rw path-element* [path-element-id]
          |        |  |     +--rw path-element-id    uint32
          |        |  |     +--rw index?             uint32
          |        |  |     +--rw (type)?
          |        |  |        +--:(ip-address)
          |        |  |        |  +--rw ip-address-hop
          |        |  |        |     +--rw address?    inet:ip-address
          |        |  |        |     +--rw hop-type?   te-hop-type
          |        |  |        +--:(as-number)
          |        |  |        |  +--rw as-number-hop
          |        |  |        |     +--rw as-number?   binary
          |        |  |        |     +--rw hop-type?    te-hop-type
          |        |  |        +--:(unnumbered-link)
          |        |  |        |  +--rw unnumbered-hop
          |        |  |        |     +--rw router-id?      inet:ip-
   address
          |        |  |        |     +--rw interface-id?   uint32
          |        |  |        |     +--rw hop-type?       te-hop-type
          |        |  |        +--:(label)
          |        |  |        |  +--rw label-hop
          |        |  |        |     +--rw value?   rt-
   types:generalized-label
          |        |  |        +--:(sid)
          |        |  |           +--rw sid-hop
          |        |  |              +--rw sid?   rt-
   types:generalized-label
          |        |  +--rw backup-path* [index]
          |        |  |  +--rw index           uint32
          |        |  |  +--rw network-ref?    leafref
          |        |  |  +--rw path-element* [path-element-id]
          |        |  |     +--rw path-element-id    uint32
          |        |  |     +--rw index?             uint32
          |        |  |     +--rw (type)?
          |        |  |        +--:(ip-address)
          |        |  |        |  +--rw ip-address-hop
          |        |  |        |     +--rw address?    inet:ip-address
          |        |  |        |     +--rw hop-type?   te-hop-type
          |        |  |        +--:(as-number)
          |        |  |        |  +--rw as-number-hop
          |        |  |        |     +--rw as-number?   binary
```

```
           |        |  |        |     +--rw hop-type?    te-hop-type
           |        |  |        +--:(unnumbered-link)
           |        |  |        |  +--rw unnumbered-hop
           |        |  |        |     +--rw router-id?      inet:ip-
   address
           |        |  |        |     +--rw interface-id?   uint32
           |        |  |        |     +--rw hop-type?       te-hop-type
           |        |  |        +--:(label)
           |        |  |        |  +--rw label-hop
           |        |  |        |     +--rw value?   rt-
   types:generalized-label
           |        |  |        +--:(sid)
           |        |  |           +--rw sid-hop
           |        |  |              +--rw sid?   rt-
   types:generalized-label
           |        |  +--rw protection-type?   uint16
           |        |  +--rw tunnels
           |        |     +--rw sharing?   boolean
           |        |     +--rw tunnel* [tunnel-name]
           |        |        +--rw tunnel-name    string
           |        |        +--rw sharing?       boolean
           |        +--rw max-lsp-bandwidth* [priority]
           |        |  +--rw priority     uint8
           |        |  +--rw bandwidth?   te-bandwidth
           |        +--rw max-link-bandwidth?        te-bandwidth
           |        +--rw max-resv-link-bandwidth?   te-bandwidth
           |        +--rw unreserved-bandwidth* [priority]
           |        |  +--rw priority     uint8
           |        |  +--rw bandwidth?   te-bandwidth
           |        +--rw te-default-metric?         uint32
           |        +--rw te-delay-metric?           uint32
           |        +--rw te-srlgs
           |        |  +--rw value*   te-types:srlg
           |        +--rw te-nsrlgs {nsrlg}?
           |           +--rw id*   uint32
           +--ro state
           |  +--ro switching-capability?        identityref
           |  +--ro encoding?                    identityref
           |  +--ro inter-layer-lock-id?         uint32
           |  +--ro protection-type?             identityref
           |  +--ro client-layer-adaptation
           |  |  +--ro switching-capability* [switching-capability
   encoding]
           |  |     +--ro switching-capability    identityref
           |  |     +--ro encoding                identityref
           |  |     +--ro bandwidth?              te-bandwidth
```

```
          |  +--ro local-link-connectivities
          |  |  +--ro number-of-entries?         uint16
          |  |  +--ro is-allowed?                boolean
          |  |  +--ro label-restriction* [inclusive-exclusive label-
   start]
          |  |  |  +--ro inclusive-exclusive    enumeration
          |  |  |  +--ro label-start            rt-types:generalized-
   label
          |  |  |  +--ro label-end?             rt-types:generalized-
   label
          |  |  |  +--ro range-bitmap?          binary
          |  |  +--ro underlay! {te-topology-hierarchy}?
          |  |  |  +--ro primary-path
          |  |  |  |  +--ro network-ref?    leafref
          |  |  |  |  +--ro path-element* [path-element-id]
          |  |  |  |     +--ro path-element-id    uint32
          |  |  |  |     +--ro index?             uint32
          |  |  |  |     +--ro (type)?
          |  |  |  |        +--:(ip-address)
          |  |  |  |        |  +--ro ip-address-hop
          |  |  |  |        |     +--ro address?    inet:ip-address
          |  |  |  |        |     +--ro hop-type?   te-hop-type
          |  |  |  |        +--:(as-number)
          |  |  |  |        |  +--ro as-number-hop
          |  |  |  |        |     +--ro as-number?   binary
          |  |  |  |        |     +--ro hop-type?    te-hop-type
          |  |  |  |        +--:(unnumbered-link)
          |  |  |  |        |  +--ro unnumbered-hop
          |  |  |  |        |     +--ro router-id?      inet:ip-
   address
          |  |  |  |        |     +--ro interface-id?   uint32
          |  |  |  |        |     +--ro hop-type?       te-hop-type
          |  |  |  |        +--:(label)
          |  |  |  |        |  +--ro label-hop
          |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
          |  |  |  |        +--:(sid)
          |  |  |  |           +--ro sid-hop
          |  |  |  |              +--ro sid?   rt-types:generalized-
   label
          |  |  |  +--ro backup-path* [index]
          |  |  |  |  +--ro index           uint32
          |  |  |  |  +--ro network-ref?    leafref
          |  |  |  |  +--ro path-element* [path-element-id]
          |  |  |  |     +--ro path-element-id    uint32
          |  |  |  |     +--ro index?             uint32
```

```
            |  |  |  |     +--ro (type)?
            |  |  |  |        +--:(ip-address)
            |  |  |  |        |  +--ro ip-address-hop
            |  |  |  |        |     +--ro address?    inet:ip-address
            |  |  |  |        |     +--ro hop-type?   te-hop-type
            |  |  |  |        +--:(as-number)
            |  |  |  |        |  +--ro as-number-hop
            |  |  |  |        |     +--ro as-number?   binary
            |  |  |  |        |     +--ro hop-type?    te-hop-type
            |  |  |  |        +--:(unnumbered-link)
            |  |  |  |        |  +--ro unnumbered-hop
            |  |  |  |        |     +--ro router-id?      inet:ip-
   address
            |  |  |  |        |     +--ro interface-id?   uint32
            |  |  |  |        |     +--ro hop-type?       te-hop-type
            |  |  |  |        +--:(label)
            |  |  |  |        |  +--ro label-hop
            |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
            |  |  |  |        +--:(sid)
            |  |  |  |           +--ro sid-hop
            |  |  |  |              +--ro sid?   rt-types:generalized-
   label
            |  |  |  +--ro protection-type?   uint16
            |  |  |  +--ro tunnels
            |  |  |     +--ro sharing?   boolean
            |  |  |     +--ro tunnel* [tunnel-name]
            |  |  |        +--ro tunnel-name    string
            |  |  |        +--ro sharing?       boolean
            |  |  +--ro max-lsp-bandwidth* [priority]
            |  |  |  +--ro priority     uint8
            |  |  |  +--ro bandwidth?   te-bandwidth
            |  |  +--ro max-link-bandwidth?        te-bandwidth
            |  |  +--ro max-resv-link-bandwidth?   te-bandwidth
            |  |  +--ro unreserved-bandwidth* [priority]
            |  |  |  +--ro priority     uint8
            |  |  |  +--ro bandwidth?   te-bandwidth
            |  |  +--ro te-default-metric?         uint32
            |  |  +--ro te-delay-metric?           uint32
            |  |  +--ro te-srlgs
            |  |  |  +--ro value*   te-types:srlg
            |  |  +--ro te-nsrlgs {nsrlg}?
            |  |  |  +--ro id*   uint32
            |  |  +--ro local-link-connectivity* [link-tp-ref]
            |  |     +--ro link-tp-ref                leafref
            |  |     +--ro is-allowed?                boolean
```

```
          |  |     +--ro label-restriction* [inclusive-exclusive
   label-start]
          |  |     |  +--ro inclusive-exclusive    enumeration
          |  |     |  +--ro label-start            rt-
   types:generalized-label
          |  |     |  +--ro label-end?             rt-
   types:generalized-label
          |  |     |  +--ro range-bitmap?          binary
          |  |     +--ro underlay! {te-topology-hierarchy}?
          |  |     |  +--ro primary-path
          |  |     |  |  +--ro network-ref?    leafref
          |  |     |  |  +--ro path-element* [path-element-id]
          |  |     |  |     +--ro path-element-id    uint32
          |  |     |  |     +--ro index?             uint32
          |  |     |  |     +--ro (type)?
          |  |     |  |        +--:(ip-address)
          |  |     |  |        |  +--ro ip-address-hop
          |  |     |  |        |     +--ro address?    inet:ip-address
          |  |     |  |        |     +--ro hop-type?   te-hop-type
          |  |     |  |        +--:(as-number)
          |  |     |  |        |  +--ro as-number-hop
          |  |     |  |        |     +--ro as-number?   binary
          |  |     |  |        |     +--ro hop-type?    te-hop-type
          |  |     |  |        +--:(unnumbered-link)
          |  |     |  |        |  +--ro unnumbered-hop
          |  |     |  |        |     +--ro router-id?      inet:ip-
   address
          |  |     |  |        |     +--ro interface-id?   uint32
          |  |     |  |        |     +--ro hop-type?       te-hop-type
          |  |     |  |        +--:(label)
          |  |     |  |        |  +--ro label-hop
          |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
          |  |     |  |        +--:(sid)
          |  |     |  |           +--ro sid-hop
          |  |     |  |              +--ro sid?   rt-
   types:generalized-label
          |  |     |  +--ro backup-path* [index]
          |  |     |  |  +--ro index           uint32
          |  |     |  |  +--ro network-ref?    leafref
          |  |     |  |  +--ro path-element* [path-element-id]
          |  |     |  |     +--ro path-element-id    uint32
          |  |     |  |     +--ro index?             uint32
          |  |     |  |     +--ro (type)?
          |  |     |  |        +--:(ip-address)
          |  |     |  |        |  +--ro ip-address-hop
```

```
          |  |     |  |        |     +--ro address?    inet:ip-address
          |  |     |  |        |     +--ro hop-type?   te-hop-type
          |  |     |  |        +--:(as-number)
          |  |     |  |        |  +--ro as-number-hop
          |  |     |  |        |     +--ro as-number?   binary
          |  |     |  |        |     +--ro hop-type?    te-hop-type
          |  |     |  |        +--:(unnumbered-link)
          |  |     |  |        |  +--ro unnumbered-hop
          |  |     |  |        |     +--ro router-id?      inet:ip-
   address
          |  |     |  |        |     +--ro interface-id?   uint32
          |  |     |  |        |     +--ro hop-type?       te-hop-type
          |  |     |  |        +--:(label)
          |  |     |  |        |  +--ro label-hop
          |  |     |  |        |     +--ro value?   rt-
   types:generalized-label
          |  |     |  |        +--:(sid)
          |  |     |  |           +--ro sid-hop
          |  |     |  |              +--ro sid?   rt-
   types:generalized-label
          |  |     |  +--ro protection-type?   uint16
          |  |     |  +--ro tunnels
          |  |     |     +--ro sharing?   boolean
          |  |     |     +--ro tunnel* [tunnel-name]
          |  |     |        +--ro tunnel-name    string
          |  |     |        +--ro sharing?       boolean
          |  |     +--ro max-lsp-bandwidth* [priority]
          |  |     |  +--ro priority     uint8
          |  |     |  +--ro bandwidth?   te-bandwidth
          |  |     +--ro max-link-bandwidth?        te-bandwidth
          |  |     +--ro max-resv-link-bandwidth?   te-bandwidth
          |  |     +--ro unreserved-bandwidth* [priority]
          |  |     |  +--ro priority     uint8
          |  |     |  +--ro bandwidth?   te-bandwidth
          |  |     +--ro te-default-metric?         uint32
          |  |     +--ro te-delay-metric?           uint32
          |  |     +--ro te-srlgs
          |  |     |  +--ro value*   te-types:srlg
          |  |     +--ro te-nsrlgs {nsrlg}?
          |  |        +--ro id*   uint32
          |  +--ro geolocation
          |     +--ro altitude?    int64
          |     +--ro latitude?    geographic-coordinate-degree
          |     +--ro longitude?   geographic-coordinate-degree
          +--ro statistics
          |  +--ro discontinuity-time          yang:date-and-time
```

```
          |  +--ro tunnel-termination-point
          |  |  +--ro disables?             yang:counter32
          |  |  +--ro enables?              yang:counter32
          |  |  +--ro maintenance-clears?   yang:counter32
          |  |  +--ro maintenance-sets?     yang:counter32
          |  |  +--ro modifies?             yang:counter32
          |  |  +--ro downs?                yang:counter32
          |  |  +--ro ups?                  yang:counter32
          |  |  +--ro in-service-clears?    yang:counter32
          |  |  +--ro in-service-sets?      yang:counter32
          |  +--ro local-link-connectivity
          |     +--ro creates?    yang:counter32
          |     +--ro deletes?    yang:counter32
          |     +--ro disables?   yang:counter32
          |     +--ro enables?    yang:counter32
          |     +--ro modifies?   yang:counter32
          +--rw supporting-tunnel-termination-point* [node-ref tunnel-
tp-ref]
             +--rw node-ref         inet:uri
             +--rw tunnel-tp-ref    binary
augment /nw:networks/nw:network/nt:link:
   +--rw te!
      +--rw config
      |  +--rw (bundle-stack-level)?
      |  |  +--:(bundle)
      |  |  |  +--rw bundled-links
      |  |  |     +--rw bundled-link* [sequence]
      |  |  |        +--rw sequence      uint32
      |  |  |        +--rw src-tp-ref?   leafref
      |  |  |        +--rw des-tp-ref?   leafref
      |  |  +--:(component)
      |  |     +--rw component-links
      |  |        +--rw component-link* [sequence]
      |  |           +--rw sequence             uint32
      |  |           +--rw src-interface-ref?   string
      |  |           +--rw des-interface-ref?   string
      |  +--rw te-link-template*     leafref {template}?
      |  +--rw te-link-attributes
      |     +--rw access-type?                      te-types:te-link-
access-type
      |     +--rw external-domain
      |     |  +--rw network-ref?            leafref
      |     |  +--rw remote-te-node-id?      te-types:te-node-id
      |     |  +--rw remote-te-link-tp-id?   te-types:te-tp-id
      |     |  +--rw plug-id?                uint32
      |     +--rw is-abstract?                      empty
```

```
          |     +--rw name?                              string
          |     +--rw underlay! {te-topology-hierarchy}?
          |     |  +--rw primary-path
          |     |  |  +--rw network-ref?    leafref
          |     |  |  +--rw path-element* [path-element-id]
          |     |  |     +--rw path-element-id    uint32
          |     |  |     +--rw index?             uint32
          |     |  |     +--rw (type)?
          |     |  |        +--:(ip-address)
          |     |  |        |  +--rw ip-address-hop
          |     |  |        |     +--rw address?    inet:ip-address
          |     |  |        |     +--rw hop-type?   te-hop-type
          |     |  |        +--:(as-number)
          |     |  |        |  +--rw as-number-hop
          |     |  |        |     +--rw as-number?   binary
          |     |  |        |     +--rw hop-type?    te-hop-type
          |     |  |        +--:(unnumbered-link)
          |     |  |        |  +--rw unnumbered-hop
          |     |  |        |     +--rw router-id?      inet:ip-address
          |     |  |        |     +--rw interface-id?   uint32
          |     |  |        |     +--rw hop-type?       te-hop-type
          |     |  |        +--:(label)
          |     |  |        |  +--rw label-hop
          |     |  |        |     +--rw value?   rt-types:generalized-
   label
          |     |  |        +--:(sid)
          |     |  |           +--rw sid-hop
          |     |  |              +--rw sid?   rt-types:generalized-label
          |     |  +--rw backup-path* [index]
          |     |  |  +--rw index           uint32
          |     |  |  +--rw network-ref?    leafref
          |     |  |  +--rw path-element* [path-element-id]
          |     |  |     +--rw path-element-id    uint32
          |     |  |     +--rw index?             uint32
          |     |  |     +--rw (type)?
          |     |  |        +--:(ip-address)
          |     |  |        |  +--rw ip-address-hop
          |     |  |        |     +--rw address?    inet:ip-address
          |     |  |        |     +--rw hop-type?   te-hop-type
          |     |  |        +--:(as-number)
          |     |  |        |  +--rw as-number-hop
          |     |  |        |     +--rw as-number?   binary
          |     |  |        |     +--rw hop-type?    te-hop-type
          |     |  |        +--:(unnumbered-link)
          |     |  |        |  +--rw unnumbered-hop
          |     |  |        |     +--rw router-id?      inet:ip-address
```

```
         |     |  |        |     +--rw interface-id?   uint32
         |     |  |        |     +--rw hop-type?       te-hop-type
         |     |  |        +--:(label)
         |     |  |        |  +--rw label-hop
         |     |  |        |     +--rw value?   rt-types:generalized-
   label
         |     |  |        +--:(sid)
         |     |  |           +--rw sid-hop
         |     |  |              +--rw sid?   rt-types:generalized-label
         |     |  +--rw protection-type?   uint16
         |     |  +--rw tunnels
         |     |     +--rw sharing?   boolean
         |     |     +--rw tunnel* [tunnel-name]
         |     |        +--rw tunnel-name    string
         |     |        +--rw sharing?       boolean
         |     +--rw admin-status?                     te-types:te-
   admin-status
         |     +--rw link-index?                       uint64
         |     +--rw administrative-group?             te-types:admin-
   groups
         |     +--rw interface-switching-capability* [switching-
   capability encoding]
         |     |  +--rw switching-capability    identityref
         |     |  +--rw encoding                identityref
         |     |  +--rw max-lsp-bandwidth* [priority]
         |     |     +--rw priority     uint8
         |     |     +--rw bandwidth?   te-bandwidth
         |     +--rw link-protection-type?             enumeration
         |     +--rw max-link-bandwidth?               te-bandwidth
         |     +--rw max-resv-link-bandwidth?          te-bandwidth
         |     +--rw unreserved-bandwidth* [priority]
         |     |  +--rw priority     uint8
         |     |  +--rw bandwidth?   te-bandwidth
         |     +--rw te-default-metric?                uint32
         |     +--rw te-delay-metric?                  uint32
         |     +--rw te-srlgs
         |     |  +--rw value*   te-types:srlg
         |     +--rw te-nsrlgs {nsrlg}?
         |        +--rw id*   uint32
         +--ro state
         |  +--ro (bundle-stack-level)?
         |  |  +--:(bundle)
         |  |  |  +--ro bundled-links
         |  |  |     +--ro bundled-link* [sequence]
         |  |  |        +--ro sequence      uint32
         |  |  |        +--ro src-tp-ref?   leafref
```

```
         |  |  |        +--ro des-tp-ref?   leafref
         |  |  +--:(component)
         |  |     +--ro component-links
         |  |        +--ro component-link* [sequence]
         |  |           +--ro sequence             uint32
         |  |           +--ro src-interface-ref?   string
         |  |           +--ro des-interface-ref?   string
         |  +--ro te-link-template*           leafref {template}?
         |  +--ro te-link-attributes
         |  |  +--ro access-type?                     te-types:te-link-
   access-type
         |  |  +--ro external-domain
         |  |  |  +--ro network-ref?            leafref
         |  |  |  +--ro remote-te-node-id?      te-types:te-node-id
         |  |  |  +--ro remote-te-link-tp-id?   te-types:te-tp-id
         |  |  |  +--ro plug-id?                uint32
         |  |  +--ro is-abstract?                    empty
         |  |  +--ro name?                           string
         |  |  +--ro underlay! {te-topology-hierarchy}?
         |  |  |  +--ro primary-path
         |  |  |  |  +--ro network-ref?    leafref
         |  |  |  |  +--ro path-element* [path-element-id]
         |  |  |  |     +--ro path-element-id    uint32
         |  |  |  |     +--ro index?             uint32
         |  |  |  |     +--ro (type)?
         |  |  |  |        +--:(ip-address)
         |  |  |  |        |  +--ro ip-address-hop
         |  |  |  |        |     +--ro address?    inet:ip-address
         |  |  |  |        |     +--ro hop-type?   te-hop-type
         |  |  |  |        +--:(as-number)
         |  |  |  |        |  +--ro as-number-hop
         |  |  |  |        |     +--ro as-number?   binary
         |  |  |  |        |     +--ro hop-type?    te-hop-type
         |  |  |  |        +--:(unnumbered-link)
         |  |  |  |        |  +--ro unnumbered-hop
         |  |  |  |        |     +--ro router-id?      inet:ip-address
         |  |  |  |        |     +--ro interface-id?   uint32
         |  |  |  |        |     +--ro hop-type?       te-hop-type
         |  |  |  |        +--:(label)
         |  |  |  |        |  +--ro label-hop
         |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
         |  |  |  |        +--:(sid)
         |  |  |  |           +--ro sid-hop
         |  |  |  |              +--ro sid?   rt-types:generalized-label
         |  |  |  +--ro backup-path* [index]
```

```
         |  |  |  |  +--ro index           uint32
         |  |  |  |  +--ro network-ref?    leafref
         |  |  |  |  +--ro path-element* [path-element-id]
         |  |  |  |     +--ro path-element-id    uint32
         |  |  |  |     +--ro index?             uint32
         |  |  |  |     +--ro (type)?
         |  |  |  |        +--:(ip-address)
         |  |  |  |        |  +--ro ip-address-hop
         |  |  |  |        |     +--ro address?    inet:ip-address
         |  |  |  |        |     +--ro hop-type?   te-hop-type
         |  |  |  |        +--:(as-number)
         |  |  |  |        |  +--ro as-number-hop
         |  |  |  |        |     +--ro as-number?   binary
         |  |  |  |        |     +--ro hop-type?    te-hop-type
         |  |  |  |        +--:(unnumbered-link)
         |  |  |  |        |  +--ro unnumbered-hop
         |  |  |  |        |     +--ro router-id?      inet:ip-address
         |  |  |  |        |     +--ro interface-id?   uint32
         |  |  |  |        |     +--ro hop-type?       te-hop-type
         |  |  |  |        +--:(label)
         |  |  |  |        |  +--ro label-hop
         |  |  |  |        |     +--ro value?   rt-types:generalized-
   label
         |  |  |  |        +--:(sid)
         |  |  |  |           +--ro sid-hop
         |  |  |  |              +--ro sid?   rt-types:generalized-label
         |  |  |  +--ro protection-type?   uint16
         |  |  |  +--ro tunnels
         |  |  |     +--ro sharing?   boolean
         |  |  |     +--ro tunnel* [tunnel-name]
         |  |  |        +--ro tunnel-name    string
         |  |  |        +--ro sharing?       boolean
         |  |  +--ro admin-status?                    te-types:te-
   admin-status
         |  |  +--ro link-index?                      uint64
         |  |  +--ro administrative-group?            te-types:admin-
   groups
         |  |  +--ro interface-switching-capability* [switching-
   capability encoding]
         |  |  |  +--ro switching-capability    identityref
         |  |  |  +--ro encoding                identityref
         |  |  |  +--ro max-lsp-bandwidth* [priority]
         |  |  |     +--ro priority     uint8
         |  |  |     +--ro bandwidth?   te-bandwidth
         |  |  +--ro link-protection-type?            enumeration
         |  |  +--ro max-link-bandwidth?              te-bandwidth
```

```
        |  |  +--ro max-resv-link-bandwidth?          te-bandwidth
        |  |  +--ro unreserved-bandwidth* [priority]
        |  |  |  +--ro priority     uint8
        |  |  |  +--ro bandwidth?   te-bandwidth
        |  |  +--ro te-default-metric?                uint32
        |  |  +--ro te-delay-metric?                  uint32
        |  |  +--ro te-srlgs
        |  |  |  +--ro value*   te-types:srlg
        |  |  +--ro te-nsrlgs {nsrlg}?
        |  |     +--ro id*   uint32
        |  +--ro oper-status?                te-types:te-oper-status
        |  +--ro is-transitional?            empty
        |  +--ro information-source?         te-info-source
        |  +--ro information-source-state
        |  |  +--ro credibility-preference?    uint16
        |  |  +--ro logical-network-element?   string
        |  |  +--ro network-instance?          string
        |  |  +--ro topology
        |  |     +--ro network-ref?   leafref
        |  |     +--ro link-ref?      leafref
        |  +--ro information-source-entry* [information-source]
        |  |  +--ro information-source                te-info-source
        |  |  +--ro information-source-state
        |  |  |  +--ro credibility-preference?    uint16
        |  |  |  +--ro logical-network-element?   string
        |  |  |  +--ro network-instance?          string
        |  |  |  +--ro topology
        |  |  |     +--ro network-ref?   leafref
        |  |  |     +--ro link-ref?      leafref
        |  |  +--ro link-index?                       uint64
        |  |  +--ro administrative-group?             te-types:admin-
   groups
        |  |  +--ro interface-switching-capability* [switching-
   capability encoding]
        |  |  |  +--ro switching-capability    identityref
        |  |  |  +--ro encoding                identityref
        |  |  |  +--ro max-lsp-bandwidth* [priority]
        |  |  |     +--ro priority     uint8
        |  |  |     +--ro bandwidth?   te-bandwidth
        |  |  +--ro link-protection-type?             enumeration
        |  |  +--ro max-link-bandwidth?               te-bandwidth
        |  |  +--ro max-resv-link-bandwidth?          te-bandwidth
        |  |  +--ro unreserved-bandwidth* [priority]
        |  |  |  +--ro priority     uint8
        |  |  |  +--ro bandwidth?   te-bandwidth
        |  |  +--ro te-default-metric?                uint32
```

```
         |  |  +--ro te-delay-metric?                  uint32
         |  |  +--ro te-srlgs
         |  |  |  +--ro value*   te-types:srlg
         |  |  +--ro te-nsrlgs {nsrlg}?
         |  |     +--ro id*   uint32
         |  +--ro recovery
         |  |  +--ro restoration-status?   te-types:te-recovery-status
         |  |  +--ro protection-status?    te-types:te-recovery-status
         |  +--ro underlay {te-topology-hierarchy}?
         |     +--ro dynamic?     boolean
         |     +--ro committed?   boolean
         +--ro statistics
            +--ro discontinuity-time                yang:date-and-time
            +--ro disables?                         yang:counter32
            +--ro enables?                          yang:counter32
            +--ro maintenance-clears?               yang:counter32
            +--ro maintenance-sets?                 yang:counter32
            +--ro modifies?                         yang:counter32
            +--ro downs?                            yang:counter32
            +--ro ups?                              yang:counter32
            +--ro fault-clears?                     yang:counter32
            +--ro fault-detects?                    yang:counter32
            +--ro protection-switches?              yang:counter32
            +--ro protection-reverts?               yang:counter32
            +--ro restoration-failures?             yang:counter32
            +--ro restoration-starts?               yang:counter32
            +--ro restoration-successes?            yang:counter32
            +--ro restoration-reversion-failures?   yang:counter32
            +--ro restoration-reversion-starts?     yang:counter32
            +--ro restoration-reversion-successes?  yang:counter32
   augment /nw:networks/nw:network/nw:node/nt:termination-point:
      +--rw te-tp-id?   te-types:te-tp-id
      +--rw te!
         +--rw config
         |  +--rw interface-switching-capability* [switching-capability
   encoding]
         |  |  +--rw switching-capability    identityref
         |  |  +--rw encoding                identityref
         |  |  +--rw max-lsp-bandwidth* [priority]
         |  |     +--rw priority     uint8
         |  |     +--rw bandwidth?   te-bandwidth
         |  +--rw inter-layer-lock-id?              uint32
         +--ro state
            +--ro interface-switching-capability* [switching-capability
   encoding]
            |  +--ro switching-capability    identityref
```

```
           |  +--ro encoding               identityref
           |  +--ro max-lsp-bandwidth* [priority]
           |     +--ro priority     uint8
           |     +--ro bandwidth?   te-bandwidth
           +--ro inter-layer-lock-id?              uint32
           +--ro geolocation
              +--ro altitude?    int64
              +--ro latitude?    geographic-coordinate-degree
              +--ro longitude?   geographic-coordinate-degree
```

7. TE Topology Yang Module

```
<CODE BEGINS> file "ietf-te-topology@2017-03-12.yang"
module ietf-te-topology {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-te-topology";

  prefix "tet";

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

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

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

  import ietf-network {
    prefix "nw";
  }

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

  import ietf-routing-types {
    prefix "rt-types";
  }
```

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

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

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

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

        Editor:   Xufeng Liu
                  <mailto:Xufeng_Liu@jabil.com>

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

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

        Editor:   Tarek Saad
                  <mailto:tsaad@cisco.com>

        Editor:   Himanshu Shah
                  <mailto:hshah@ciena.com>

        Editor:   Oscar Gonzalez De Dios
                  <mailto:oscar.gonzalezdedios@telefonica.com>";

     description "TE topology model";

     revision "2017-03-12" {
       description "Initial revision";
       reference "TBD";
     }
```

```
     /*
      * Features
      */
     feature nsrlg {
       description
         "This feature indicates that the system supports NSRLG
          (Not Sharing Risk Link Group).";
     }

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

     feature template {
       description
         "This feature indicates that the system supports
          template configuration.";
     }

     /*
      * Typedefs
      */
     typedef geographic-coordinate-degree {
         type decimal64 {
           fraction-digits 8;
         }
         description
           "Decimal degree (DD) used to express latitude and longitude
            geographic coordinates.";
     } // geographic-coordinate-degree

     typedef te-bandwidth {
       type string {
         pattern
           '0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
         + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?[pP](\+)?(12[0-7]|'
         + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}|\d+'
         + '(,(0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
```

```
        + '1(\.([\da-fA-F]{0,5}[02468aAcCeE]?)?)?[pP](\+)?(12[0-7]|'
        + '1[01]\d|0?\d?\d)?)|0[xX][\da-fA-F]{1,8}|\d+))*';
      }
      description
        "This is the generic bandwidth type that is a string containing
         a list of numbers separated by commas, with each of these
         number can be non-negative decimal, hex integer, or hex float:
         (dec | hex | float)[*(','(dec | hex | float))]
         For packet switching type, a float number is used, such as
         0x1p10.
         For OTN switching type, a list of integers can be used, such
         as '0,2,3,1', indicating 2 odu0's and 1 odu3.
         For DWDM, a list of pairs of slot number and width can be
         used, such as '0, 2, 3, 3', indicating a frequency slot 0 with
         slot width 2 and a frequency slot 3 with slot width 3.";
    } // te-bandwidth

    typedef te-info-source {
      type enumeration {
        enum "unknown" {
          description "The source is unknown.";
        }
        enum "locally-configured" {
          description "Configured entity.";
        }
        enum "ospfv2" {
          description "OSPFv2.";
        }
        enum "ospfv3" {
          description "OSPFv3.";
        }
        enum "isis" {
          description "ISIS.";
        }
        enum "bgp-ls" {
          description "BGP-LS.";
          reference
            "RFC7752: North-Bound Distribution of Link-State and
             Traffic Engineering (TE) Information Using BGP";
        }
```

```
        enum "system-processed" {
          description "System processed entity.";
        }
        enum "other" {
          description "Other source.";
        }
      }
      description
        "Describining the type of source that has provided the
         related information, and the source credibility.";
    } // te-info-source

    typedef te-path-disjointness {
      type bits {
        bit node {
          position 0;
          description "Node disjoint.";
        }
        bit link {
          position 1;
          description "Link disjoint.";
        }
        bit srlg {
          position 2;
          description "SRLG (Shared Risk Link Group) disjoint.";
        }
      }
      description
        "Type of the resource disjointness for a TE tunnel path.";
      reference
        "RFC4872: RSVP-TE Extensions in Support of End-to-End
         Generalized Multi-Protocol Label Switching (GMPLS)
         Recovery";
    } // te-path-disjointness

    /*
     * Groupings
     */
    grouping connectivity-label-restriction-list {
      description
```

```
        "List of abel restrictions specifying what labels may or may
         not be used on a link connectivity.";
      list label-restriction {
        key "inclusive-exclusive label-start";
        description
          "List of abel restrictions specifying what labels may or may
           not be used on a link connectivity.";
        reference
          "RFC7579: General Network Element Constraint Encoding
           for GMPLS-Controlled Networks";
        leaf inclusive-exclusive {
          type enumeration {
            enum inclusive {
              description "The label or label range is inclusive.";
            }
            enum exclusive {
              description "The label or label range is exclusive.";
            }
          }
          description
            "Whether the list item is inclusive or exclusive.";
        }
        leaf label-start {
          type rt-types:generalized-label;
          description
            "This is the starting lable if a lable range is specified.
             This is the lable value if a single lable is specified,
             in which case, attribute 'label-end' is not set.";
        }
        leaf label-end {
          type rt-types:generalized-label;
          description
            "The ending lable if a lable range is specified;
             This attribute is not set, If a single lable is
             specified.";
        }
        leaf range-bitmap {
          type binary;
          description
            "When there are gaps between label-start and label-end,
```

```
            this attribute is used to specified the possitions
            of the used labels.";
        }
      }
    } // connectivity-label-restrictions

    grouping connectivity-matrix-entry-attributes {
      description
        "Attributes of connectivity matrix entry.";
      leaf is-allowed {
        type boolean;
        description
          "true  - switching is allowed,
           false - switching is disallowed.";
      }
      uses connectivity-label-restriction-list;
      container underlay {
        if-feature te-topology-hierarchy;
        presence
          "Indicates the underlay exists for this link.";
        description "Attributes of the te-link underlay.";
        reference
          "RFC4206: Label Switched Paths (LSP) Hierarchy with
           Generalized Multi-Protocol Label Switching (GMPLS)
           Traffic Engineering (TE)";

        uses te-link-underlay-attributes;
      } // underlay
      uses te-link-iscd-attributes;
      uses te-link-connectivity-attributes;
    } // connectivity-matrix-entry-attributes

    grouping geolocation-container {
      description
        "A container containing a GPS location.";
      container geolocation{
        description
          "A container containing a GPS location.";
        leaf altitude {
          type int64;
```

```
          units millimeter;
          description
            "Distance above the sea level.";
        }
        leaf latitude {
          type geographic-coordinate-degree {
            range "-90..90";
          }
          description
            "Relative position north or south on the Earth's surface.";
        }
        leaf longitude {
          type geographic-coordinate-degree {
            range "-180..180";
          }
          description
            "Angular distance east or west on the Earth's surface.";
        }
      } // gps-location
    } // geolocation-container

    grouping information-source-state-attributes {
      description
        "The attributes identifying source that has provided the
         related information, and the source credibility.";
      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.";
      }
      leaf logical-network-element {
        type string;
        description
          "When applicable, this is the name of a logical network
           element from which the information is learned.";
      } // logical-network-element
```

```
      leaf network-instance {
        type string;
        description
          "When applicable, this is the name of a network-instance
           from which the information is learned.";
      } // network-instance
    } // information-source-state-attributes

    grouping information-source-per-link-attributes {
      description
        "Per node container of the attributes identifying source that
         has provided the related information, and the source
         credibility.";
      leaf information-source {
        type te-info-source;
        description
          "Indicates the source of the information.";
      }
      container information-source-state {
        description
          "The container contains state attributes related to
           the information source.";
        uses information-source-state-attributes;
        container topology {
          description
            "When the information is processed by the system,
             the attributes in this container indicate which topology
             is used to process to generate the result information.";
          uses te-topology-ref;
          leaf link-ref {
            type leafref {
              path "/nw:networks/nw:network[nw:network-id = "
                + "current()/../network-ref]/nt:link/nt:link-id";
              require-instance false;
            }
            description
              "A reference to a link-id.";
          }
        } // topology
      } // information-source-state
```

```
    } // information-source-per-link-attributes

    grouping information-source-per-node-attributes {
      description
        "Per node container of the attributes identifying source that
         has provided the related information, and the source
         credibility.";
      leaf information-source {
        type te-info-source;
        description
          "Indicates the source of the information.";
      }
      container information-source-state {
        description
          "The container contains state attributes related to
           the information source.";
        uses information-source-state-attributes;
        container topology {
          description
            "When the information is processed by the system,
             the attributes in this container indicate which topology
             is used to process to generate the result information.";
          uses te-topology-ref;
          leaf node-ref {
            type leafref {
              path "/nw:networks/nw:network[nw:network-id = "
                + "current()/../network-ref]/nw:node/nw:node-id";
              require-instance false;
            }
            description
              "A reference to a node-id.";
          }
        } // topology
      } // information-source-state
    } // information-source-per-node-attributes

    grouping interface-switching-capability-list {
      description
        "List of Interface Switching Capabilities Descriptors (ISCD)";
```

```
      list interface-switching-capability {
        key "switching-capability encoding";
        description
          "List of Interface Switching Capabilities Descriptors (ISCD)
           for this link.";
        reference
          "RFC3471: Generalized Multi-Protocol Label Switching (GMPLS)
           Signaling Functional Description.
           RFC4203: OSPF Extensions in Support of Generalized
           Multi-Protocol Label Switching (GMPLS).";
        leaf switching-capability {
          type identityref {
            base te-types:switching-capabilities;
          }
          description
            "Switching Capability for this interface.";
        }
        leaf encoding {
          type identityref {
            base te-types:lsp-encoding-types;
          }
          description
            "Encoding supported by this interface.";
        }
        uses te-link-iscd-attributes;
      } // interface-switching-capability
    } // interface-switching-capability-list

    grouping statistics-per-link {
      description
        "Statistics attributes per TE link.";
      leaf discontinuity-time {
        type yang:date-and-time;
        mandatory true;
        description
          "The time on the most recent occasion at which any one or
           more of this interface's counters suffered a
           discontinuity.  If no such discontinuities have occurred
           since the last re-initialization of the local management
           subsystem, then this node contains the time the local
```

```
           management subsystem re-initialized itself.";
       }
       /* Administrative attributes */
       leaf disables {
         type yang:counter32;
         description
           "Number of times that link was disabled.";
       }
       leaf enables {
         type yang:counter32;
         description
           "Number of times that link was enabled.";
       }
       leaf maintenance-clears {
         type yang:counter32;
         description
           "Number of times that link was put out of maintenance.";
       }
       leaf maintenance-sets {
         type yang:counter32;
         description
           "Number of times that link was put in maintenance.";
         }
       leaf modifies {
         type yang:counter32;
         description
           "Number of times that link was modified.";
       }
       /* Operational attributes */
       leaf downs {
         type yang:counter32;
         description
           "Number of times that link was set to operational down.";
       }
       leaf ups {
         type yang:counter32;
         description
           "Number of times that link was set to operational up.";
       }
       /* Recovery attributes */
```

```
        leaf fault-clears {
          type yang:counter32;
          description
            "Number of times that link experienced fault clear event.";
        }
        leaf fault-detects {
          type yang:counter32;
          description
            "Number of times that link experienced fault detection.";
        }
        leaf protection-switches {
          type yang:counter32;
          description
            "Number of times that link experienced protection
             switchover.";
        }
        leaf protection-reverts {
          type yang:counter32;
          description
            "Number of times that link experienced protection
             reversion.";
        }
        leaf restoration-failures {
          type yang:counter32;
          description
            "Number of times that link experienced restoration
             failure.";
        }
        leaf restoration-starts {
          type yang:counter32;
          description
            "Number of times that link experienced restoration
             start.";
        }
        leaf restoration-successes {
          type yang:counter32;
          description
            "Number of times that link experienced restoration
             success.";
        }
```

```
       leaf restoration-reversion-failures {
         type yang:counter32;
         description
           "Number of times that link experienced restoration reversion
            failure.";
       }
       leaf restoration-reversion-starts {
         type yang:counter32;
         description
           "Number of times that link experienced restoration reversion
            start.";
       }
       leaf restoration-reversion-successes {
         type yang:counter32;
         description
           "Number of times that link experienced restoration reversion
            success.";
       }
     } // statistics-per-link

     grouping statistics-per-node {
       description
         "Statistics attributes per TE node.";
       leaf discontinuity-time {
         type yang:date-and-time;
         mandatory true;
         description
           "The time on the most recent occasion at which any one or
            more of this interface's counters suffered a
            discontinuity.  If no such discontinuities have occurred
            since the last re-initialization of the local management
            subsystem, then this node contains the time the local
            management subsystem re-initialized itself.";
       }
       container node {
         description
           "Containing TE node level statistics attributes.";
         leaf disables {
           type yang:counter32;
           description
```

```
            "Number of times that node was disabled.";
          }
          leaf enables {
            type yang:counter32;
            description
              "Number of times that node was enabled.";
          }
          leaf maintenance-sets {
            type yang:counter32;
            description
              "Number of times that node was put in maintenance.";
          }
          leaf maintenance-clears {
            type yang:counter32;
            description
              "Number of times that node was put out of maintenance.";
          }
          leaf modifies {
            type yang:counter32;
            description
              "Number of times that node was modified.";
          }
        } // node
        container connectivity-matrix-entry {
          description
            "Containing connectivity matrix entry level statistics
             attributes.";
          leaf creates {
            type yang:counter32;
            description
              "Number of times that a connectivity matrix entry was
               created.";
            reference
              "RFC6241. Section 7.2 for 'create' operation. ";
          }
          leaf deletes {
            type yang:counter32;
            description
              "Number of times that a connectivity matrix entry was
               deleted.";
```

```
          reference
            "RFC6241. Section 7.2 for 'delete' operation. ";
        }
        leaf disables {
          type yang:counter32;
          description
            "Number of times that a connectivity matrix entry was
             disabled.";
        }
        leaf enables {
          type yang:counter32;
          description
            "Number of times that a connectivity matrix entry was
             enabled.";
        }
        leaf modifies {
          type yang:counter32;
          description
            "Number of times that a connectivity matrix entry was
             modified.";
        }
      } // connectivity-matrix-entry
    } // statistics-per-node

    grouping statistics-per-ttp {
      description
        "Statistics attributes per TE TTP (Tunnel Termination Point).";
      leaf discontinuity-time {
        type yang:date-and-time;
        mandatory true;
        description
          "The time on the most recent occasion at which any one or
           more of this interface's counters suffered a
           discontinuity.  If no such discontinuities have occurred
           since the last re-initialization of the local management
           subsystem, then this node contains the time the local
           management subsystem re-initialized itself.";
      }
      container tunnel-termination-point {
        description
```

```
          "Containing TE TTP (Tunnel Termination Point) level
           statistics attributes.";
        /* Administrative attributes */
        leaf disables {
          type yang:counter32;
          description
            "Number of times that TTP was disabled.";
        }
        leaf enables {
          type yang:counter32;
          description
            "Number of times that TTP was enabled.";
        }
        leaf maintenance-clears {
          type yang:counter32;
          description
            "Number of times that TTP was put out of maintenance.";
        }
        leaf maintenance-sets {
          type yang:counter32;
          description
            "Number of times that TTP was put in maintenance.";
        }
        leaf modifies {
          type yang:counter32;
          description
            "Number of times that TTP was modified.";
        }
        /* Operational attributes */
        leaf downs {
          type yang:counter32;
          description
            "Number of times that TTP was set to operational down.";
        }
        leaf ups {
          type yang:counter32;
          description
            "Number of times that TTP was set to operational up.";
        }
        leaf in-service-clears {
```

```
            type yang:counter32;
            description
              "Number of times that TTP was taken out of service
               (TE tunnel was released).";
          }
          leaf in-service-sets {
            type yang:counter32;
            description
              "Number of times that TTP was put in service by a TE
               tunnel (TE tunnel was set up).";
          }
        } // tunnel-termination-point

        container local-link-connectivity {
          description
            "Containing TE LLCL (Local Link Connectivity List) level
             statistics attributes.";
          leaf creates {
            type yang:counter32;
            description
              "Number of times that an LLCL entry was created.";
            reference
              "RFC6241. Section 7.2 for 'create' operation. ";
          }
          leaf deletes {
            type yang:counter32;
            description
              "Number of times that an LLCL entry was deleted.";
            reference
              "RFC6241. Section 7.2 for 'delete' operation.";
          }
          leaf disables {
            type yang:counter32;
            description
              "Number of times that an LLCL  entry was disabled.";
          }
          leaf enables {
            type yang:counter32;
            description
              "Number of times that an LLCL  entry was enabled.";
```

```
        }
        leaf modifies {
          type yang:counter32;
          description
            "Number of times that an LLCL  entry was modified.";
        }
      } // local-link-connectivity
    } // statistics-per-ttp

    grouping te-link-augment {
      description
        "Augmentation for TE link.";

      container te {
        must "count(../nt:supporting-link)<=1" {
          description
            "For a link in a TE topology, there cannot be more
             than 1 supporting link. If one or more link paths are
             abstracted, the underlay is used.";
        }
        presence "TE support.";
        description
          "Indicates TE support.";

        container config {
          description
            "Configuration data.";
          uses te-link-config;
        } // config
        container state {
          config false;
          description
            "Operational state data.";
          uses te-link-config;
          uses te-link-state-derived;
        } // state
        container statistics {
          config false;
          description
            "Statistics data.";
```

```
          uses statistics-per-link;
        } // statistics
      } // te
    } // te-link-augment

    grouping te-link-config {
      description
        "TE link configuration grouping.";
      choice bundle-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.";
            reference
              "RFC4201: Link Bundling in MPLS Traffic Engineering
              (TE).";
            list bundled-link {
              key "sequence";
              description
                "Specify a bundled interface that is
                 further partitioned.";
              leaf sequence {
                type uint32;
                description
                  "Identify the sequence in the bundle.";
              }
              leaf src-tp-ref {
                type leafref {
                  path "../../../../../../nw:node[nw:node-id = "
                    + "current()/../../../../../nt:source/"
                    + "nt:source-node]/"
                    + "nt:termination-point/nt:tp-id";
                  require-instance true;
                }
                description
                  "Reference to another TE termination point on the
                   same souruce node.";
```

```
            }
            leaf des-tp-ref {
              type leafref {
                path "../../../../../../nw:node[nw:node-id = "
                  + "current()/../../../../../nt:destination/"
                  + "nt:dest-node]/"
                  + "nt:termination-point/nt:tp-id";
                require-instance true;
              }
              description
                "Reference to another TE termination point on the
                 same destination node.";
            }
          } // list bundled-link
        }
      }
      case component {
        container component-links {
          description
            "A set of component links";
          list component-link {
            key "sequence";
            description
              "Specify a component interface that is
               sufficient to unambiguously identify the
               appropriate resources";

            leaf sequence {
              type uint32;
              description
                "Identify the sequence in the bundle.";
            }
            leaf src-interface-ref {
              type string;
              description
                "Reference to component link interface on the
                 source node.";
            }
            leaf des-interface-ref {
              type string;
```

```
                description
                  "Reference to component link interface on the
                   destinatioin node.";
              }
            }
          }
        }
      } // bundle-stack-level

      leaf-list te-link-template {
        if-feature template;
        type leafref {
          path "../../../../../te/templates/link-template/name";
        }
        description
          "The reference to a TE link template.";
      }
      uses te-link-config-attributes;
    } // te-link-config

    grouping te-link-config-attributes {
      description
        "Link configuration attributes in a TE topology.";
      container te-link-attributes {
        description "Link attributes in a TE topology.";
        leaf access-type {
          type te-types:te-link-access-type;
          description
            "Link access type, which can be point-to-point or
             multi-access.";
        }
        container external-domain {
          description
            "For an inter-domain link, specify the attributes of
             the remote end of link, to facilitate the signalling at
             local end.";
          uses te-topology-ref;
          leaf remote-te-node-id {
            type te-types:te-node-id;
            description
```

```
            "Remote TE node identifier, used together with
             remote-te-link-id to identify the remote link
             termination point in a different domain.";
          }
          leaf remote-te-link-tp-id {
            type te-types:te-tp-id;
            description
              "Remote TE link termination point identifier, used
               together with remote-te-node-id to identify the remote
               link termination point in a different domain.";
          }
          leaf plug-id {
            type uint32;
            description
              "A topology-wide unique number that identifies on the
               network a connectivity supporting a given inter-domain
               TE link. This is more flexible alternative to specifying
               remote-te-node-id and remote-te-link-tp-id, when the
               provider does not know remote-te-node-id and
               remote-te-link-tp-id or need to give client the
               flexibility to mix-n-match multiple topologies.";
          }
        }
        leaf is-abstract {
          type empty;
          description "Present if the link is abstract.";
        }
        leaf name {
          type string;
          description "Link Name.";
        }
        container underlay {
          if-feature te-topology-hierarchy;
          presence
            "Indicates the underlay exists for this link.";
          description "Attributes of the te-link underlay.";
          reference
            "RFC4206: Label Switched Paths (LSP) Hierarchy with
             Generalized Multi-Protocol Label Switching (GMPLS)
             Traffic Engineering (TE)";
```

```
          uses te-link-underlay-attributes;
        } // underlay
        leaf admin-status {
          type te-types:te-admin-status;
          description
            "The administrative state of the link.";
        }

        uses te-link-info-attributes;
      } // te-link-attributes
    } // te-link-config-attributes

    grouping te-link-connectivity-attributes {
      description
        "Advertised TE connectivity attributes.";
      leaf max-link-bandwidth {
        type te-bandwidth;
        description
          "Maximum bandwidth that can be seen on this link in this
           direction. Units in bytes per second.";
        reference
          "RFC3630: Traffic Engineering (TE) Extensions to OSPF
           Version 2.
           RFC5305: IS-IS Extensions for Traffic Engineering.";
      }
      leaf max-resv-link-bandwidth {
        type te-bandwidth;
        description
          "Maximum amount of bandwidth that can be reserved in this
           direction in this link. Units in bytes per second.";
        reference
          "RFC3630: Traffic Engineering (TE) Extensions to OSPF
           Version 2.
           RFC5305: IS-IS Extensions for Traffic Engineering.";
      }
      list unreserved-bandwidth {
        key "priority";
        max-elements "8";
        description
```

```
            "Unreserved bandwidth for 0-7 priority levels. Units in
             bytes per second.";
          reference
            "RFC3630: Traffic Engineering (TE) Extensions to OSPF
             Version 2.
             RFC5305: IS-IS Extensions for Traffic Engineering.";
          leaf priority {
            type uint8 {
              range "0..7";
            }
            description "Priority.";
          }
          leaf bandwidth {
            type te-bandwidth;
            description
              "Unreserved bandwidth for this level.";
          }
        }
        leaf te-default-metric {
          type uint32;
          description
            "Traffic engineering metric.";
        }
        leaf te-delay-metric {
          type uint32;
          description
            "Traffic engineering delay metric.";
        }
        container te-srlgs {
          description
            "Containing a list of SLRGs.";
          leaf-list value {
            type te-types:srlg;
            description "SRLG value.";
            reference
              "RFC4202: Routing Extensions in Support of
               Generalized Multi-Protocol Label Switching (GMPLS).";
          }
        }
        container te-nsrlgs {
```

```
        if-feature nsrlg;
        description
          "Containing a list of NSRLGs (Not Sharing Risk Link
           Groups).
           When an abstract TE link is configured, this list specifies
           the request that underlay TE paths need to be mutually
           disjoint with other TE links in the same groups.";
        leaf-list id {
          type uint32;
          description
            "NSRLG ID, uniquely configured within a topology.";
          reference
            "RFC4872: RSVP-TE Extensions in Support of End-to-End
             Generalized Multi-Protocol Label Switching (GMPLS)
             Recovery";
        }
      }
    } // te-link-connectivity-attributes

    grouping te-link-info-attributes {
      description
        "Advertised TE information attributes.";
      leaf link-index {
        type uint64;
        description
          "The link identifier.  If OSPF is used, this represents an
           ospfLsdbID.  If IS-IS is used, this represents an isisLSPID.
           If a locally configured link is used, this object represents
           a unique value, which is locally defined in a router.";
      }
      leaf administrative-group {
        type te-types:admin-groups;
        description
          "Administrative group or color of the link.
           This attribute covers both administrative group (defined in
           RFC3630, RFC5329, and RFC5305), and extended administrative
           group (defined in RFC7308).";
      }
      uses interface-switching-capability-list;
      leaf link-protection-type {
```

```
          type enumeration {
            enum "unprotected" {
              description "Unprotected.";
            }
            enum "extra-traffic" {
              description "Extra traffic.";
            }
            enum "shared" {
              description "Shared.";
            }
            enum "1-for-1" {
              description "One for one protection.";
            }
            enum "1-plus-1" {
              description "One plus one protection.";
            }
            enum "enhanced" {
              description "Enhanced protection.";
            }
          }
          description
            "Link Protection Type desired for this link.";
          reference
            "RFC4202: Routing Extensions in Support of
             Generalized Multi-Protocol Label Switching (GMPLS).";
        }
        uses te-link-connectivity-attributes;
      } // te-link-info-attributes

      grouping te-link-iscd-attributes {
        description
          "TE link ISCD (Interface Switching Capability Descriptor)
           attributes.";
        reference
          "Sec 1.4, RFC4203: OSPF Extensions in Support of Generalized
           Multi-Protocol Label Switching (GMPLS). Section 1.4.";
        list max-lsp-bandwidth {
          key "priority";
          max-elements "8";
          description
```

```
          "Maximum LSP Bandwidth at priorities 0-7.";
        leaf priority {
          type uint8 {
            range "0..7";
          }
          description "Priority.";
        }
        leaf bandwidth {
          type te-bandwidth;
          description
            "Max LSP Bandwidth for this level";
        }
      }
    } // te-link-iscd-attributes

    grouping te-link-state-derived {
      description
        "Link state attributes in a TE topology.";
      leaf oper-status {
        type te-types:te-oper-status;
        description
          "The current operational state of the link.";
      }
      leaf is-transitional {
        type empty;
        description
          "Present if the link is transitional, used as an
           alternative approach in lieu of inter-layer-lock-id
           for path computation in a TE topology covering multiple
           layers or multiple regions.";
        reference
          "RFC5212: Requirements for GMPLS-Based Multi-Region and
           Multi-Layer Networks (MRN/MLN).
           RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
           for Multi-Layer and Multi-Region Networks (MLN/MRN).";
      }
      uses information-source-per-link-attributes;
      list information-source-entry {
        key "information-source";
        description
```

```
          "A list of information sources learned, including the one
           used.";
        uses information-source-per-link-attributes;
        uses te-link-info-attributes;
      }
      container recovery {
        description
          "Status of the recovery process.";
        leaf restoration-status {
          type te-types:te-recovery-status;
          description
            "Restoration status.";
        }
        leaf protection-status {
          type te-types:te-recovery-status;
          description
            "Protection status.";
        }
      }
      container underlay {
        if-feature te-topology-hierarchy;
        description "State attributes for te-link underlay.";
        uses te-link-state-underlay-attributes;
      }
    } // te-link-state-derived

    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-underlay-attributes {
       description "Attributes for  te-link underlay.";
       reference
         "RFC4206: Label Switched Paths (LSP) Hierarchy with
          Generalized Multi-Protocol Label Switching (GMPLS)
          Traffic Engineering (TE)";
       container primary-path {
         description
           "The service path on the underlay topology that
            supports this link.";
         uses te-topology-ref;
         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;
         }
       } // primary-path
       list backup-path {
         key "index";
         description
           "A list of backup service paths on the underlay topology that
            protect the underlay primary path. If the primary path is
            not protected, the list contains zero elements. If the
            primary path is protected, the list contains one or more
            elements.";
         leaf index {
           type uint32;
           description
             "A sequence number to identify a backup path.";
         }
         uses te-topology-ref;
         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 protection-type {
        type uint16;
        description
          "Underlay protection type desired for this link";
      }
      container tunnels {
        description
          "Underlay TE tunnels supporting this TE link.";
        leaf sharing {
          type boolean;
          default true;
          description
            "'true' if the underlay tunnel can be shared with other
             TE links;
             'false' if the underlay tunnel is dedicated to this
             TE link.
             This leaf is the default option for all TE tunnels,
             and may be overridden by the per TE tunnel value.";
        }
        list tunnel {
          key "tunnel-name";
          description
            "Zero, one or more underlay TE tunnels that support this TE
             link.";
          leaf tunnel-name {
            type string;
            description
              "A tunnel name uniquely identifies an underlay TE tunnel,
               used together with the source-node of this link.
               The detailed information of this tunnel can be retrieved
               from the ietf-te model.";
            reference "RFC3209";
```

```
          }
          leaf sharing {
            type boolean;
            description
              "'true' if the underlay tunnel can be shared with other
               TE links;
               'false' if the underlay tunnel is dedicated to this
               TE link.";
          }
        } // tunnel
      } // tunnels
    } // te-link-underlay-attributes

    grouping te-node-augment {
      description
        "Augmentation for TE node.";

      leaf te-node-id {
        type te-types:te-node-id;
        description
          "The identifier of a node in the TE topology.
           A node is specific to a topology to which it belongs.";
      }

      container te {
        must "../te-node-id" {
          description
            "te-node-id is mandatory.";
        }
        must "count(../nw:supporting-node)<=1" {
          description
            "For a node in a TE topology, there cannot be more
             than 1 supporting node. If multiple nodes are abstracted,
             the underlay-topology is used.";
        }
        presence "TE support.";
        description
          "Indicates TE support.";

        container config {
```

```
          description
            "Configuration data.";
          uses te-node-config;
        } // config
        container state {
          config false;
          description
            "Operational state data.";

          uses te-node-config;
          uses te-node-state-derived;
        } // state
        container statistics {
          config false;
          description
            "Statistics data.";
          uses statistics-per-node;
        } // statistics

        list tunnel-termination-point {
          key "tunnel-tp-id";
          description
            "A termination point can terminate a tunnel.";
          leaf tunnel-tp-id {
            type binary;
            description
              "Tunnel termination point identifier.";
          }

          container config {
            description
              "Configuration data.";
            uses te-node-tunnel-termination-attributes;
          }
          container state {
            config false;
            description
              "Operational state data.";

            uses te-node-tunnel-termination-attributes;
```

```
            uses geolocation-container;
          } // state
          container statistics {
            config false;
            description
              "Statistics data.";
            uses statistics-per-ttp;
          } // statistics

          // Relations to other tunnel termination points
          list supporting-tunnel-termination-point {
            key "node-ref tunnel-tp-ref";
            description
              "Identifies the tunnel termination points, that this
               tunnel termination point is depending on.";
            leaf node-ref {
              type inet:uri;
              /* The followings are the intended valications
               * but some Yang validation tools fail on them.
              type union {
                type leafref {
                  path "../../../../nw:supporting-node/nw:node-ref";
                  require-instance false;
                }
                type leafref {
                  path "/nw:networks/nw:network"+
                    "[nw:network-id="+
                    "current()/../../../../te/config/"+
                    "te-node-attributes/underlay-topology/"+
                    "network-ref]/nw:node/nw:node-id";
                  require-instance false;
                }
                type leafref {
                  path "/nw:networks/nw:network"+
                    "[nw:network-id="+
                    "current()/../../../../te/state/"+
                    "te-node-attributes/underlay-topology/"+
                    "network-ref]/nw:node/nw:node-id";
                  require-instance false;
                }
```

```
            }
            **************************************************/
            description
              "This leaf identifies in which node the supporting
               tunnel termination point is present.";
          }
          leaf tunnel-tp-ref {
            type binary;
            /* The followings are the intended valications
             * but some Yang validation tools fail on them.
            type union {
              type leafref {
                path "/nw:networks/nw:network"+
                  "[nw:network-id="+
                  "current()/../../../../nw:supporting-node/"+
                  "nw:network-ref]/"+
                  "nw:node[nw:node-id=current()/../node-ref]/te/"+
                  "tunnel-termination-point/tunnel-tp-id";
                require-instance false;
              }
              type leafref {
                path "/nw:networks/nw:network"+
                  "[nw:network-id="+
                  "current()/../../../../te/config/"+
                  "te-node-attributes/underlay-topology/"+
                  "network-ref]/"+
                  "nw:node[nw:node-id=current()/../node-ref]/te/"+
                  "tunnel-termination-point/tunnel-tp-id";
                require-instance false;
              }
              type leafref {
                path "/nw:networks/nw:network"+
                  "[nw:network-id="+
                  "current()/../../../../te/state/"+
                  "te-node-attributes/underlay-topology/"+
                  "network-ref]/"+
                  "nw:node[nw:node-id=current()/../node-ref]/te/"+
                  "tunnel-termination-point/tunnel-tp-id";
                require-instance false;
              }
```

```
              }
              ************************************************/
              description
                "Reference to a tunnel terminiation point, which is
                 either in the supporting node or a node in an
                 underlay topology.";
            }
          } // supporting-tunnel-termination-point
        } // tunnel-termination-point
      } // te
    } // te-node-augment

    grouping te-node-config {
      description "TE node configuration grouping.";

      leaf-list te-node-template {
        if-feature template;
        type leafref {
          path "../../../../../te/templates/node-template/name";
        }
        description
          "The reference to a TE node template.";
      }
      uses te-node-config-attributes;
    } // te-node-config

    grouping te-node-config-attributes {
      description "Configuration node attributes in a TE topology.";
      container te-node-attributes {
        description "Containing node attributes in a TE topology.";
        leaf admin-status {
          type te-types:te-admin-status;
          description
            "The administrative state of the link.";
        }
        uses te-node-connectivity-matrix;
        uses te-node-info-attributes;
      } // te-node-attributes
    } // te-node-config-attributes
```

```
    grouping te-node-config-attributes-template {
      description
        "Configuration node attributes for template in a TE topology.";
      container te-node-attributes {
        description "Containing node attributes in a TE topology.";
        leaf admin-status {
          type te-types:te-admin-status;
          description
            "The administrative state of the link.";
        }
        uses te-node-info-attributes;
      } // te-node-attributes
    } // te-node-config-attributes-template

    grouping te-node-connectivity-matrix {
      description "Connectivity matrix on a TE node.";
      container connectivity-matrices {
        description
          "Containing connectivity matrix on a TE node.";
        leaf number-of-entries {
          type uint16;
          description
            "The number of connectivity matrix entries.
             If this number is speficied in the configuration request,
             the number is requested number of entries, which may not
             all be listed in the list;
             if this number is reported in the state data,
             the number is the current number of operational entries.";
        }
        uses connectivity-matrix-entry-attributes;
        list connectivity-matrix {
          key "id";
          description
            "Represents node's switching limitations, i.e. limitations
             in interconnecting network TE links across the node.";
          reference
            "RFC7579: General Network Element Constraint Encoding
             for GMPLS-Controlled Networks.";
          leaf id {
            type uint32;
```

```
            description "Identifies the connectivity-matrix entry.";
          }
          container from {
            leaf tp-ref {
              type leafref {
                path "../../../../../../../nt:termination-point/"+
                  "nt:tp-id";
              }
              description
                "Relative reference to source termination point.";
            }
            description
              "Reference to source NTP.";
          }
          container to {
            leaf tp-ref {
              type leafref {
                path "../../../../../../../nt:termination-point/"+
                  "nt:tp-id";
              }
              description
                "Relative reference to destination termination point.";
            }
            description
              "Reference to destination NTP.";
          }
          uses connectivity-matrix-entry-attributes;
        } // connectivity-matrix
      } // connectivity-matrices
    } // te-node-connectivity-matrix

    grouping te-node-connectivity-matrix-abs {
      description
        "Connectivity matrix on a TE node, using absolute
         paths to reference termination points.";
      list connectivity-matrix {
        key "id";
        description
          "Represents node's switching limitations, i.e. limitations
           in interconnecting network TE links across the node.";
```

```
        reference
          "RFC7579: General Network Element Constraint Encoding
           for GMPLS-Controlled Networks.";
        leaf id {
          type uint32;
          description "Identifies the connectivity-matrix entry.";
        }
        container from {
          uses nt:tp-ref;
          description
            "Reference to source NTP.";
        }
        container to {
          uses nt:tp-ref;
          description
            "Reference to destination NTP.";
        }
        leaf is-allowed {
          type boolean;
          description
            "true  - switching is allowed,
             false - switching is disallowed.";
        }
      }
    } // te-node-connectivity-matrix-abs

    grouping te-node-info-attributes {
      description
        "Advertised TE information attributes.";
      leaf domain-id {
        type uint32;
        description
          "Identifies the domain that this node belongs.
           This attribute is used to support inter-domain links.";
        reference
          "RFC5152: A Per-Domain Path Computation Method for
           Establishing Inter-Domain Traffic Engineering (TE)
           Label Switched Paths (LSPs).
           RFC5392: OSPF Extensions in Support of Inter-Autonomous
           System (AS) MPLS and GMPLS Traffic Engineering.
```

```
            RFC5316: ISIS Extensions in Support of Inter-Autonomous
            System (AS) MPLS and GMPLS Traffic Engineering.";
        }
        leaf is-abstract {
          type empty;
          description
            "Present if the node is abstract, not present if the node
             is actual.";
        }
        leaf name {
          type inet:domain-name;
          description "Node name.";
        }
        leaf-list signaling-address {
          type inet:ip-address;
          description "Node signaling address.";
        }
        container underlay-topology {
          if-feature te-topology-hierarchy;
          description
            "When an abstract node encapsulates a topology,
             the attributes in this container point to said topology.";
          uses te-topology-ref;
        }
      } // te-node-info-attributes

      grouping te-node-state-derived {
        description "Node state attributes in a TE topology.";
        leaf oper-status {
          type te-types:te-oper-status;
          description
            "The current operational state of the node.";
        }
        uses geolocation-container;
        leaf is-multi-access-dr {
          type empty;
          description
            "The presence of this attribute indicates that this TE node
             is a pseudonode elected as a designated router.";
          reference
```

```
           "RFC3630: Traffic Engineering (TE) Extensions to OSPF
            Version 2.
            RFC1195: Use of OSI IS-IS for Routing in TCP/IP and Dual
            Environments.";
        }
        uses information-source-per-node-attributes;
        list information-source-entry {
          key "information-source";
          description
            "A list of information sources learned, including the one
             used.";
          uses information-source-per-node-attributes;
          uses te-node-connectivity-matrix;
          uses te-node-info-attributes;
        }
      } // te-node-state-derived

      grouping te-node-state-derived-notification {
        description "Node state attributes in a TE topology.";
        leaf oper-status {
          type te-types:te-oper-status;
          description
            "The current operational state of the node.";
        }
        leaf is-multi-access-dr {
          type empty;
          description
            "The presence of this attribute indicates that this TE node
             is a pseudonode elected as a designated router.";
          reference
            "RFC3630: Traffic Engineering (TE) Extensions to OSPF
             Version 2.
             RFC1195: Use of OSI IS-IS for Routing in TCP/IP and Dual
             Environments.";
        }
        uses information-source-per-node-attributes;
        list information-source-entry {
          key "information-source";
          description
            "A list of information sources learned, including the one
```

```
            used.";
          uses information-source-per-node-attributes;
          uses te-node-connectivity-matrix-abs;
          uses te-node-info-attributes;
        }
      } // te-node-state-derived-notification

      grouping te-node-tunnel-termination-attributes {
        description
          "Termination capability of a tunnel termination point on a
           TE node.";

        leaf switching-capability {
          type identityref {
            base te-types:switching-capabilities;
          }
          description
            "Switching Capability for this interface.";
        }
        leaf encoding {
          type identityref {
            base te-types:lsp-encoding-types;
          }
          description
            "Encoding supported by this interface.";
        }
        leaf inter-layer-lock-id {
          type uint32;
          description
            "Inter layer lock ID, used for path computation in a TE
             topology covering multiple layers or multiple regions.";
          reference
            "RFC5212: Requirements for GMPLS-Based Multi-Region and
             Multi-Layer Networks (MRN/MLN).
             RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
             for Multi-Layer and Multi-Region Networks (MLN/MRN).";
        }
        leaf protection-type {
          type identityref {
            base te-types:lsp-prot-type;
```

```
        }
        description
          "The protection type that this tunnel termination point
           is capable of.";
      }

      container client-layer-adaptation {
        description
          "Containing capability information to support a client layer
           adaption in multi-layer topology.";
        list switching-capability {
          key "switching-capability encoding";
          description
            "List of supported switching capabilities";
          reference
            "RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
             for Multi-Layer and Multi-Region Networks (MLN/MRN).
             RFC4202: Routing Extensions in Support of
             Generalized Multi-Protocol Label Switching (GMPLS).";
          leaf switching-capability {
            type identityref {
              base te-types:switching-capabilities;
            }
            description
              "Switching Capability for the client layer adaption.";
          }
          leaf encoding {
            type identityref {
              base te-types:lsp-encoding-types;
            }
            description
              "Encoding supported by the client layer adaption.";
          }
          leaf bandwidth {
            type te-bandwidth;
            description
              "Bandwidth available for the client layer adaption.";
          }
        }
      }
```

```
      container local-link-connectivities {
        description
          "Containing local link connectivity list for
           a tunnel termination point on a TE node.";
        leaf number-of-entries {
          type uint16;
          description
            "The number of local link connectivity list entries.
             If this number is speficied in the configuration request,
             the number is requested number of entries, which may not
             all be listed in the list;
             if this number is reported in the state data,
             the number is the current number of operational entries.";
        }
        uses connectivity-matrix-entry-attributes;
        list local-link-connectivity {
          key "link-tp-ref";
          description
            "The termination capabilities between
             tunnel-termination-point and link termination-point.
             The capability information can be used to compute
             the tunnel path.
             The Interface Adjustment Capability Descriptors (IACD)
             [RFC6001] on each link-tp can be derived from this
             local-link-connectivity list.";
          reference
            "RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
             for Multi-Layer and Multi-Region Networks (MLN/MRN).";
          leaf link-tp-ref {
            type leafref {
              path "../../../../../../nt:termination-point/nt:tp-id";
            }
            description
              "Link termination point.";
          }

          uses connectivity-matrix-entry-attributes;
        } // local-link-connectivity
      } // local-link-connectivities
```

```
    } // te-node-tunnel-termination-attributes

    grouping te-path-element {
      description
        "A group of attributes defining an element in a TE path
         such as TE node, TE link, TE atomic resource or label.";
      uses te-types:explicit-route-hop_config;
    } // te-path-element

    grouping te-termination-point-augment {
      description
        "Augmentation for TE termination point.";

      leaf te-tp-id {
        type te-types:te-tp-id;
        description
          "An identifier to uniquely identify a TE termination
           point.";
      }

      container te {
        must "../te-tp-id";
        presence "TE support.";
        description
          "Indicates TE support.";

        container config {
          description
            "Configuration data.";
          uses te-termination-point-config;
        } // config
        container state {
          config false;
          description
            "Operational state data.";
          uses te-termination-point-config;
          uses geolocation-container;
        } // state
      } // te
    } // te-termination-point-augment
```

```
    grouping te-termination-point-config {
      description
        "TE termination point configuration grouping.";
      uses interface-switching-capability-list;
      leaf inter-layer-lock-id {
        type uint32;
        description
          "Inter layer lock ID, used for path computation in a TE
           topology covering multiple layers or multiple regions.";
        reference
          "RFC5212: Requirements for GMPLS-Based Multi-Region and
           Multi-Layer Networks (MRN/MLN).
           RFC6001: Generalized MPLS (GMPLS) Protocol Extensions
           for Multi-Layer and Multi-Region Networks (MLN/MRN).";
      }
    } // te-termination-point-config

    grouping te-topologies-augment {
      description
        "Augmentation for TE topologies.";

      container te {
        presence "TE support.";
        description
          "Indicates TE support.";

        container templates {
          description
            "Configuration parameters for templates used for TE
             topology.";

          list node-template {
            if-feature template;
            key "name";
            leaf name {
              type te-types:te-template-name;
              description
                "The name to identify a TE node template.";
            }
```

```
            description
              "The list of TE node templates used to define sharable
               and reusable TE node attributes.";
            uses template-attributes;
            uses te-node-config-attributes-template;
          } // node-template

          list link-template {
            if-feature template;
            key "name";
            leaf name {
              type te-types:te-template-name;
              description
                "The name to identify a TE link template.";
            }
            description
              "The list of TE link templates used to define sharable
               and reusable TE link attributes.";
            uses template-attributes;
            uses te-link-config-attributes;
          } // link-template
        } // templates
      } // te
    } // te-topologies-augment

    grouping te-topology-augment {
      description
        "Augmentation for TE topology.";

      leaf provider-id {
        type te-types:te-global-id;
        description
          "An identifier to uniquely identify a provider.";
      }
      leaf client-id {
        type te-types:te-global-id;
        description
          "An identifier to uniquely identify a client.";
      }
      leaf te-topology-id {
```

```
        type te-types:te-topology-id;
        description
          "It is presumed that a datastore will contain many
           topologies. To distinguish between topologies it is
           vital to have UNIQUE topology identifiers.";
      }

      container te {
        must "../provider-id and ../client-id and ../te-topology-id";
        presence "TE support.";
        description
          "Indicates TE support.";

        container config {
          description
            "Configuration data.";
          uses te-topology-config;
        } // config
        container state {
          config false;
          description
            "Operational state data.";
          uses te-topology-config;
          uses geolocation-container;
        } // state
      } // te
    } // te-topology-augment

    grouping te-topology-config {
      description
        "TE topology configuration grouping.";
      leaf preference {
        type uint8 {
          range "1..255";
        }
        description
          "Specifies a preference for this topology. A lower number
           indicates a higher preference.";
      }
      leaf optimization-criterion {
```

```
        type identityref {
          base te-types:te-optimization-criterion;
        }
        description
          "Optimization criterion applied to this topology.";
        reference
          "RFC3272: Overview and Principles of Internet Traffic
           Engineering.";
      }
      list nsrlg {
        if-feature nsrlg;
        key "id";
        description
          "List of NSRLGs (Not Sharing Risk Link Groups).";
        reference
          "RFC4872: RSVP-TE Extensions in Support of End-to-End
           Generalized Multi-Protocol Label Switching (GMPLS)
           Recovery";
        leaf id {
          type uint32;
          description
            "Identify the NSRLG entry.";
        }
        leaf disjointness {
          type te-path-disjointness;
          description
            "The type of resource disjointness.";
        }
      } // nsrlg
    } // te-topology-config

    grouping te-topology-ref {
      description
        "References a TE topology.";
      leaf network-ref {
        type leafref {
          path "/nw:networks/nw:network/nw:network-id";
          require-instance false;
        }
        description
```

```
          "A reference to a network-id in base ietf-network module.";
      }
    } // te-topology-ref

    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.";
      }
    } // te-topology-type

    grouping template-attributes {
      description
        "Common attributes for all templates.";

      leaf priority {
        type uint16;
        description
          "The preference value to resolve conflicts between different
           templates. When two or more templates specify values for
           one configuration attribute, the value from the template
           with the highest priority is used.";
      }
      leaf reference-change-policy {
        type enumeration {
          enum no-action {
            description
              "When an attribute changes in this template, the
               configuration node referring to this template does
               not take any action.";
          }
          enum not-allowed {
            description
              "When any configuration object has a reference to this
               template, changing this template is not allowed.";
          }
          enum cascade {
```

```
            description
              "When an attribute changes in this template, the
               configuration object referring to this template applies
               the new attribute value to the corresponding
               configuration.";
          }
        }
        description
          "This attribute specifies the action taken to a configuration
           node that has a reference to this template.";
      }
    } // template-attributes

    /*
     * Configuration data nodes
     */
    augment "/nw:networks/nw:network/nw:network-types" {
      description
        "Introduce new network type for TE topology.";
      uses te-topology-type;
    }

    augment "/nw:networks" {
      description
        "Augmentation parameters for TE topologies.";
      uses te-topologies-augment;
    }

    augment "/nw:networks/nw:network" {
      when "nw:network-types/te-topology" {
        description
          "Augmentation parameters apply only for networks with
           TE topology type.";
      }
      description
        "Configuration parameters for TE topology.";
      uses te-topology-augment;
    }

    augment "/nw:networks/nw:network/nw:node" {
```

```
      when "../nw:network-types/te-topology" {
        description
          "Augmentation parameters apply only for networks with
           TE topology type.";
      }
      description
        "Configuration parameters for TE at node level.";
      uses te-node-augment;
    }

    augment "/nw:networks/nw:network/nt:link" {
      when "../nw:network-types/te-topology" {
        description
          "Augmentation parameters apply only for networks with
           TE topology type.";
      }
      description
        "Configuration parameters for TE at link level";
      uses te-link-augment;
    }

    augment "/nw:networks/nw:network/nw:node/"
          + "nt:termination-point" {
      when "../../nw:network-types/te-topology" {
        description
          "Augmentation parameters apply only for networks with
           TE topology type.";
      }
      description
        "Configuration parameters for TE at termination point level";
      uses te-termination-point-augment;
    }
  }
  <CODE ENDS>
```

8. 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.

9. 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

10. References

10.1. 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.

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

[RFC3945] Mannie, E., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", October 2004.

[YANG-NET-TOPO] Clemm, A., "A Data Model for Network Topologies", draft-ietf-i2rs-yang-network-topo (Work in Progress).

[YANG-PUSH] Clemm, A., "Subscribing to YANG datastore push updates", draft-clemm-netconf-yang-push (Work in Progress).

[RFC5277bis] Clemm, A., "Subscribing to Event Notifications", draft-ietf-netconf-rfc5277bis (Work in Progress).

[YANG-SCHEDULE] Liu, X., " A YANG Data Model for Configuration Scheduling", draft-liu-netmod-yang-schedule-00 (Work in Progress).

10.2. Informative References

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

11. Acknowledgments

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

Contributors

Sergio Belotti
Nokia
Email: sergio.belotti@nokia.com

Dieter Beller
Nokia
Email: Dieter.Beller@nokia.com

Authors' Addresses

Xufeng Liu
Jabil
Email: Xufeng_Liu@jabil.com

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

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

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

Himanshu Shah
Ciena
Email: hshah@ciena.com

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