TEAS Working Group   T. Saad
Internet-Draft   R. Gandhi
Intended status: Standards Track   Cisco Systems Inc
Expires: 5 August 2024   X. Liu
                                                              Alef Edge
                                                          V. P. Beeram
                                                       Juniper Networks
                                                             I. Bryskin
                                                             Individual
                                                        2 February 2024

A YANG Data Model for Traffic Engineering Tunnels, Label Switched Paths
                            and Interfaces
                      draft-ietf-teas-yang-te-36

Abstract

   This document defines a YANG data model for the provisioning and
   management of Traffic Engineering (TE) tunnels, Label Switched Paths
   (LSPs), and interfaces. The model covers data that is independent of
   any technology or dataplane encapsulation and is divided into two
   YANG modules that cover device-specific, and device independent data.

   This model covers data for configuration, operational state, remote
   procedural calls, and event notifications.

Status of This Memo

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

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on 5 August 2024.

Copyright Notice

Table of Contents

1. Introduction

   YANG [RFC6020] and [RFC7950] is a data modeling language that was
   introduced to define the contents of a conceptual data store that
   allows networked devices to be managed using NETCONF [RFC6241]. YANG
   has proved relevant beyond its initial confines, as bindings to other
   interfaces (e.g. RESTCONF [RFC8040]) and encoding other than XML
   (e.g. JSON) are being defined. Furthermore, YANG data models can be
   used as the basis of implementation for other interfaces, such as CLI
   and programmatic APIs.

   This document describes a YANG data model for Traffic Engineering
   (TE) tunnels, Label Switched Paths (LSPs), and interfaces. The data
   model is divided into two YANG modules. The module 'ietf-te.yang'
   includes data that is generic and device-independent, while the
   module 'ietf-te-device.yang' includes data that is device-specific.

   The document describes a high-level relationship between the modules
   defined in this document, as well as other external protocol YANG
   modules. The TE generic YANG data model does not include any data
   specific to a signaling protocol. It is expected other data plane
   technology model(s) will augment the TE generic YANG data model.

   Also, it is expected other YANG modules that model TE signaling
   protocols, such as RSVP-TE ([RFC3209], [RFC3473]), or Segment-Routing
   TE (SR-TE) [RFC9256] will augment the generic TE YANG module.

2. Terms and Conventions

2.1. Requirements Language

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

2.2. Terminology

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

   * client

   * configuration data

   *   state data

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

   *   augment

   *   data model

   *   data node

2.3. Prefixes in Data Node Names

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

Prefix	YANG module	Reference
===========	=============	====================================
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
rt-types	ietf-routing-types	[RFC8294]
te-types	ietf-te-types	[I-D.draft-ietf-teas-rfc8776-update]
te-packet-types	ietf-te-packet-types	[I-D.draft-ietf-teas-rfc8776-update]
te	ietf-te	this document
te-dev	ietf-te-device	this document

                  Table 1: Prefixes and corresponding YANG modules

2.4. Model Tree Diagrams

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

3. Design Considerations

   This document describes a generic TE YANG data model that is
   independent of any dataplane technology. One of the design
   objectives is to allow specific data plane technology models to reuse
   the TE generic data model and possibly augment it with technology
   specific data.

   The elements of the generic TE YANG data model, including TE Tunnels,
   LSPs, and interfaces have leaf(s) that identify the technology layer
   where they reside. For example, the LSP encoding type can identify
   the technology associated with a TE Tunnel or LSP.

   Also, the generic TE YANG data model does not cover signaling
   protocol data. The signaling protocol used to instantiate TE LSPs
   are outside the scope of this document and expected to be covered by
   augmentations defined in other document(s).

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

   * The generic TE YANG data model 'ietf-te' contains device
      independent data and can be used to model data off a device (e.g.
      on a TE controller). When the model is used to manage a specific
      device, the model contains the TE Tunnels originating from the
      specific device. When the model is used to manage a TE
      controller, the 'tunnel' list contains all TE Tunnels and TE
      tunnel segments originating from device(s) that the TE controller
      manages.

   * The device-specific TE data is defined in module 'ietf-te-device'
      as shown in Figure 1.

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

   * Suitable defaults are specified for all configurable elements.

   * The model declares a number of TE functions as features that can
      be optionally supported.

3.1. State Data Organization

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

4. Model Overview

   The data models defined in this document cover the core TE features
   that are commonly supported by different vendor implementations. The
   support of extended or vendor specific TE feature(s) is expected to
   either be in augmentations, or deviations to this model that are
   defined in separate documents.

4.1. Module Relationship

   The generic TE YANG data model that is defined in "ietf-te.yang"
   covers the building blocks that are device independent and agnostic
   of any specific technology or control plane instances. The TE device
   model defined in "ietf-te-device.yang" augments the generic TE YANG
   data model and covers data that is specific to a device -- for
   example, attributes of TE interfaces, or TE timers that are local to
   a TE node.

   The TE data models for specific instances of data plane technology
   exist in separate YANG modules that augment the generic TE YANG data
   model. The TE data models for specific instances of signaling
   protocols are outside the scope of this document and are defined in
   other documents. For example, the RSVP-TE YANG model augmentation of
   the TE model is covered in a separate document.
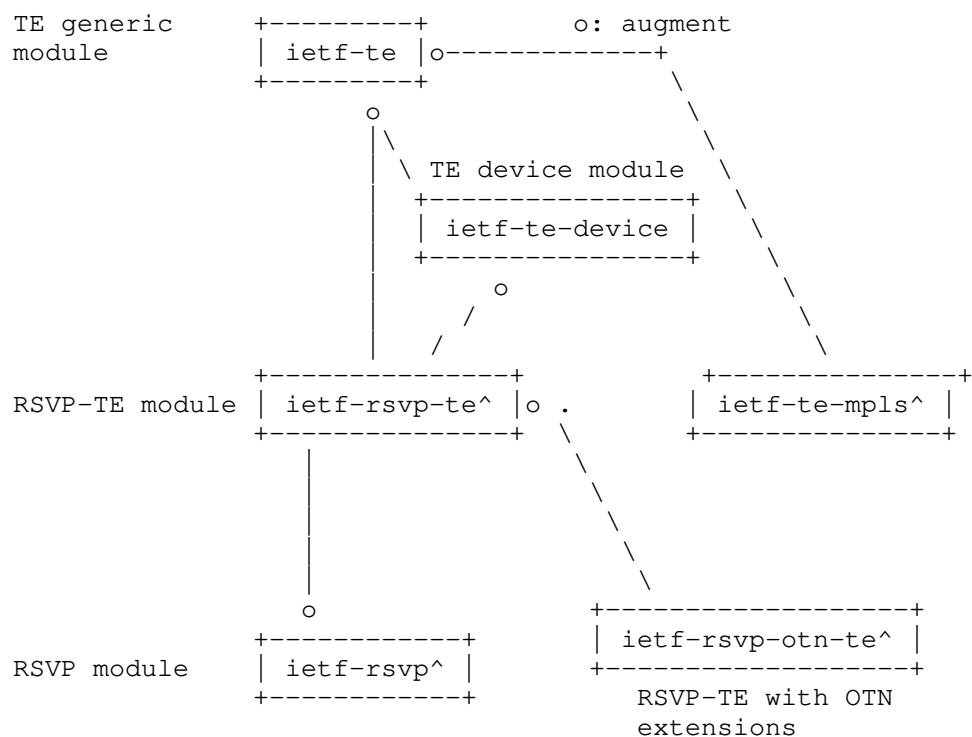
```
TE generic       +---------+          o: augment
module           | ietf-te |o------------+
                 +---------+              \
                      o                    \
                      |\                     \
                      | \ TE device module    \
                      |  +---------------+      \
                      |  | ietf-te-device |      \
                      |  +---------------+        \
                      |         o                  \
                      |        /                    \
                      |       /                      \
                 +---------------+          +---------------+
RSVP-TE module   | ietf-rsvp-te^ |o .       | ietf-te-mpls^ |
                 +---------------+   \       +---------------+
                      |               \
                      |                \
                      |                 \
                      |                  \
                      |                   \
                      o                    +------------------+
                 +------------+            | ietf-rsvp-otn-te^ |
RSVP module      | ietf-rsvp^ |            +------------------+
                 +------------+             RSVP-TE with OTN
                                               extensions


           X---oY indicates that module X augments module Y
            ^ indicates a module defined in other documents
```

         Figure 1: Relationship of TE module(s) with signaling protocol
                                   modules

5. TE YANG Model

   The generic TE YANG module ('ietf-te') is meant for the management
   and operation of a TE network. This includes creating, modifying and
   retrieving information about TE Tunnels, LSPs, and interfaces and
   their associated attributes (e.g. Administrative-Groups, SRLGs,
   etc.).

   A full tree diagram of the TE model is shown in the Appendix in
   Figure 13.

5.1. Module Structure

   The 'te' container is the top level container in the 'ietf-te'
   module. The presence of the 'te' container enables TE function
   system wide. Below provides further descriptions of containers that
   exist under the 'te' top level container.

   There are three further containers grouped under the 'te' container
   as shown in Figure 2 and described below.

   globals:

      The 'globals' container maintains the set of global TE attributes
      that can be applicable to TE Tunnels and interfaces.

   tunnels:

      The 'tunnels' container includes the list of TE Tunnels that are
      instantiated. Refer to Section 5.1.2 for further details on the
      properties of a TE Tunnel.

   lsps:

      The 'lsps' container includes the list of TE LSP(s) that are
      instantiated for TE Tunnels. Refer to Section 5.1.3 for further
      details on the properties of a TE LSP.

   The model also contains two Remote Procedure Calls (RPCs) as shown in
   Figure 13 and described below.

   tunnels-path-compute:

      A RPC to request path computation for a specific TE Tunnel. The
      RPC allows requesting path computation using atomic and stateless
      operation. A tunnel may also be configured in 'compute-only' mode
      to provide stateful path updates - see Section 5.1.2 for further
      details.

   tunnels-action:

      An RPC to request a specific action (e.g. reoptimize, or tear-and-
      setup) to be taken on a specific tunnel or all tunnels.

   Figure 13 shows the relationships of these containers and RPCs within
   the 'ietf-te' module.

```
   module: ietf-te
     +--rw te!
        +--rw globals
        |      ...
        +--rw tunnels
        |      ...
        +--ro lsps
               ...

   rpcs:
     +---x tunnels-path-compute
     |   +---w input
     |   |     ...
     |   +--ro output
     |         ...
     +---x tunnels-actions
         +---w input
         |     ...
         +--ro output
               ...
```

          Figure 2: TE Tunnel model high-level YANG tree view

5.1.1. TE Globals

   The 'globals' container covers properties that control a TE feature's
   behavior system-wide, and its respective state as shown in Figure 3
   and described in the text that follows.

```
       +--rw globals
       |   +--rw named-admin-groups
       |   |   +--rw named-admin-group* [name]
       |   |         ...
       |   +--rw named-srlgs
       |   |   +--rw named-srlg* [name]
       |   |         ...
       |   +--rw named-path-constraints
       |       +--rw named-path-constraint* [name]
```

        Figure 3: TE globals YANG subtree high-level structure

   named-admin-groups:

      A YANG container for the list of named (extended) administrative
      groups that may be applied to TE links.

   named-srlgs:

A YANG container for the list of named Shared Risk Link Groups
(SRLGs) that may be applied to TE links.

named-path-constraints:

A YANG container for a list of named path constraints. Each named
path constraint is composed of a set of constraints that can be
applied during path computation. A named path constraint can be
applied to multiple TE Tunnels. Path constraints may also be
specified directly under the TE Tunnel. The path constraints
specified under the TE Tunnel take precedence over the path
constraints derived from the referenced named path constraint. A
named path constraint entry can be formed of the path constraints
shown in Figure 4:

```
  +--rw named-path-constraints
     +--rw named-path-constraint* [name]
           {te-types:named-path-constraints}?
        +--rw name                         string
        +--rw te-bandwidth
        |     ...
        +--rw link-protection?             identityref
        +--rw setup-priority?              uint8
        +--rw hold-priority?               uint8
        +--rw signaling-type?              identityref
        +--rw path-metric-bounds
        |     ...
        +--rw path-affinities-values
        |     ...
        +--rw path-affinity-names
        |     ...
        +--rw path-srlgs-lists
        |     ...
        +--rw path-srlgs-names
        |     ...
        +--rw disjointness?
        |        te-path-disjointness
        +--rw explicit-route-objects
        |     ...
        +--rw path-in-segment!
        |     ...
        +--rw path-out-segment!
              ...
```

               Figure 4: Named path constraints YANG subtree

   o name: A YANG leaf that holds the named path constraint
         entry. This is unique in the list and used as a key.

o te-bandwidth: A YANG container that holds the technology agnostic TE bandwidth constraint.

o link-protection: A YANG leaf that holds the link protection type constraint required for the links to be included in the computed path.

o setup/hold priority: YANG leafs that hold the LSP setup and hold admission priority as defined in [RFC3209].

o signaling-type: A YANG leaf that holds the LSP setup type, such as RSVP-TE or SR.

o path-metric-bounds: A YANG container that holds the set of metric bounds applicable on the computed TE tunnel path.

o path-affinities-values: A YANG container that holds the set of affinity values and mask to be used during path computation.

o path-affinity-names: A YANG container that holds the set of named affinity constraints and corresponding inclusion or exclusion instructions for each to be used during path computation.

o path-srlgs-lists: A YANG container that holds the set of SRLG values and corresponding inclusion or exclusion instructions to be used during path computation.

o path-srlgs-names: A YANG container that holds the set of named SRLG constraints and corresponding inclusion or exclusion instructions for each to be used during path computation.

o disjointness: The level of resource disjointness constraint that the secondary path of a TE tunnel has to adhere to.

o explicit-route-objects: A YANG container that holds path constraints in the form of route entries present in following two lists:

  + 'route-object-exclude-always': a list of route entries that are always excluded from the path computation. The exclusion of a route entry in this list during path computation is not order sensitive.

+ 'route-object-include-exclude': a list of route entries
  to include or exclude route entry constraints for the
  path computation. The constraint type (include or
  exclude) is specified with each route entry. The path
  computation considers route entry constraints in the
  order they appear in this list. Once a route entry
  constraint is consumed from this list, it is not
  considered any further in the computation of the path.

The 'route-object-include-exclude' is used to configure
constraints on which route objects (e.g., nodes, links) are
included or excluded in the path computation.

The interpretation of an empty 'route-object-include-
exclude' list depends on the TE Tunnel (end-to-end or Tunnel
Segment) and on the specific path, according to the
following rules:

1. An empty 'route-object-include-exclude' list for the
    primary path of an end-to-end TE Tunnel indicates that
    there are no route objects to be included or excluded in
    the path computation.

    2. An empty 'route-object-include-exclude' list for the
        primary path of a TE Tunnel Segment indicates that no
        primary LSP is required for that TE Tunnel.

    3. An empty 'route-object-include-exclude' list for a
        reverse path means it always follows the forward path
        (i.e., the TE Tunnel is co-routed). When the 'route-
        object-include-exclude' list is not empty, the reverse
        path is routed independently of the forward path.

    4. An empty 'route-object-include-exclude' list for the
        secondary (forward) path indicates that the secondary
        path has the same endpoints as the primary path.

o path-in-segment: A YANG container that contains a list of
   label restrictions that have to be taken into considerations
   when crossing domains. This TE tunnel segment in this case
   is being stitched to the upstream TE tunnel segment.

o path-out-segment: A YANG container that contains a list of
   label restrictions that have to be taken into considerations
   when crossing domains. The TE tunnel segment in this case
   is being stitched to the downstream TE tunnel segment.

5.1.2. TE Tunnels

   The 'tunnels' container holds the list of TE Tunnels that are
   provisioned on ingress LER devices in the network as shown in
   Figure 5.

```
   module: ietf-te
     +--rw te
        +--rw tunnels
           +--rw tunnel* [name]
              +--rw name                       string
              +--rw alias?                     string
              +--rw identifier?                uint32
              +--rw color?                     uint32
              +--rw description?               string
              +--rw admin-state?               identityref
              +--ro operational-state?         identityref
              +--rw encoding?                  identityref
              +--rw switching-type?            identityref
              +--rw source
              |     ...
              +--rw destination
              |     ...
              +--rw bidirectional?             boolean
              +--rw controller
              |     ...
              +--rw reoptimize-timer?          uint16
              +--rw association-objects
              |     ...
              +--rw protection
              |     ...
              +--rw restoration
              |     ...
              +--rw network-id?                nw:network-id
              +--rw te-topology-identifier
              |     ...
              +--rw te-bandwidth
              |     ...
              +--rw link-protection?           identityref
              +--rw setup-priority?            uint8
              +--rw hold-priority?             uint8
              +--rw signaling-type?            identityref
              +--rw hierarchy
              |     ...
              +--rw primary-paths
              |     ...
              +--rw secondary-paths
              |     ...
```

```
             +--rw secondary-reverse-paths
             |       ...
             +---x tunnel-action
             |       ...
             +---x protection-external-commands
                     ...
```

                 Figure 5: TE Tunnel YANG subtree structure

   When the model is used to manage a specific device, the 'tunnel' list
   contains the TE Tunnels originating from the specific device. When
   the model is used to manage a TE controller, the 'tunnel' list
   contains all TE Tunnels and TE tunnel segments originating from
   device(s) that the TE controller manages.

   The TE Tunnel model allows the configuration and management of the
   following TE tunnel objects:

   TE Tunnel:

      A YANG container of one or more TE LSPs established between the
      source and destination TE Tunnel termination points.

   TE Path:

      An engineered path that once instantiated in the forwarding plane
      can be used to forward traffic from the source to the destination
      TE Tunnel termination points.

   TE LSP:

      A TE LSP is a connection-oriented service established over a TE
      Path and that allows the delivery of traffic between the TE Tunnel
      source and destination termination points.

   TE Tunnel Segment:

      A part of a multi-domain TE Tunnel that is within a specific
      network domain.

   The TE Tunnel has a number of attributes that are set directly under
   the tunnel (as shown in Figure 5). The main attributes of a TE
   Tunnel are described below:

   operational-state:

      A YANG leaf that holds the operational state of the tunnel.

name:

   A YANG leaf that holds the name of a TE Tunnel. The name of the
   TE Tunnel uniquely identifies the tunnel within the TE tunnel
   list. The name of the TE Tunnel can be formatted as a Uniform
   Resource Indicator (URI) by including the namespace to ensure
   uniqueness of the name amongst all the TE Tunnels present on
   devices and controllers. The configured TE Tunnels can be
   reported with the name of the device embedded within the TE Tunnel
   name. For initiated TE Tunnels from the controller, the
   controller is responsible to ensures that TE Tunnel names are
   unique.

alias:

   A YANG leaf that holds an alternate name to the TE tunnel. Unlike
   the TE tunnel name, the alias can be modified at any time during
   the lifetime of the TE tunnel.

identifier:

   A YANG leaf that holds an identifier of the tunnel. This
   identifier is unique amongst tunnels originated from the same
   ingress device.

color:

   A YANG leaf that holds the color associated with the TE tunnel.
   The color is used to map or steer services that carry matching
   color on to the TE tunnel as described in [RFC9012].

admin-state:

   A YANG leaf that holds the tunnel administrative state. The
   administrative status in state datastore transitions to 'tunnel-
   admin-up' when the tunnel used by the client layer, and to
   'tunnel-admin-down' when it is not used by the client layer.

operational-state:

   A YANG leaf that holds the tunnel operational state.

encoding/switching:

   The 'encoding' and 'switching-type' are YANG leafs that define the
   specific technology in which the tunnel operates in as described
   in [RFC3945].

source/destination:

> YANG containers that hold the tunnel source and destination node
> endpoints identities, including:
>
> - te-node-id: A YANG leaf that holds the identifier of the source
>   or destination of the TE Tunnel TE node identifiers as defined
>   in [I-D.draft-ietf-teas-rfc8776-update].
>
> - node-id: A YANG leaf that holds the identifier of the source or
>   destination of the TE Tunnel node identifiers as defined in
>   [RFC8345].
>
> - tunnel-tp-id: A YANG leaf that holds the identifier of the
>   source or destination of the TE Tunnel Termination Points
>   (TTPs) as defined in [RFC8795]. The TTP identifiers are
>   optional on nodes that have a single TTP per node. For
>   example, TTP identifiers are optional for packet (IP/MPLS)
>   routers.

bidirectional:

> A YANG leaf that when present indicates the LSP of a TE Tunnel is
> bidirectional as defined in [rfc3473].

controller:

> A YANG container that holds tunnel data relevant to an optional
> external TE controller that may initiate or control a tunnel.
> This target node may be augmented by external module(s), for
> example, to add data for PCEP initiated and/or delegated tunnels.

reoptimize-timer:

> A YANG leaf to set the interval period for tunnel reoptimization.

association-objects:

> A YANG container that holds the set of associations of the TE
> Tunnel to other TE Tunnels. Associations at the TE Tunnel level
> apply to all paths of the TE Tunnel. The TE tunnel associations
> can be overridden by associations configured directly under the TE
> Tunnel path.

protection:

> A YANG container that holds the TE Tunnel protection properties.

restoration:

   A YANG container that holds the TE Tunnel restoration properties.

te-topology-identifier:

   A YANG container that holds the topology identifier associated
   with the topology where paths for the TE tunnel are computed as
   defined in [RFC8795].

network-id:

   A YANG leaf that can optionally be used to identify the network
   topology where paths for the TE tunnel are computed as defined in
   [RFC8345].

hierarchy:

   A YANG container that holds hierarchy related properties of the TE
   Tunnel. A TE LSP can be set up in MPLS or Generalized MPLS
   (GMPLS) networks to be used as a TE link to carry traffic in other
   (client) networks [RFC6107]. In this case, the model introduces
   the TE Tunnel hierarchical link endpoint parameters to identify
   the specific link in the client layer that the underlying TE
   Tunnel is associated with. The hierarchy container includes the
   following:

   o dependency-tunnels: A set of hierarchical TE Tunnels
         provisioned or to be provisioned in the immediate lower
         layer that this TE tunnel depends on for multi-layer path
         computation. A dependency TE Tunnel is provisioned if and
         only if it is used (selected by path computation) at least
         by one client layer TE Tunnel. The TE link in the client
         layer network topology supported by a dependent TE Tunnel is
         dynamically created only when the dependency TE Tunnel is
         actually provisioned.

      o hierarchical-link: A YANG container that holds the identity
         of the hierarchical link (in the client layer) that is
         supported by this TE Tunnel. The endpoints of the
         hierarchical link are defined by TE tunnel source and
         destination node endpoints. The hierarchical link can be
         identified by its source and destination link termination
         point identifiers.

   primary-paths:

A YANG container that holds the list of primary paths. A primary
path is identified by 'name'. A primary path is selected from the
list to instantiate a primary forwarding LSP for the tunnel. The
list of primary paths is visited by order of preference. A
primary path has the following attributes:

- primary-reverse-path: A YANG container that holds properties of
  the primary reverse path. The reverse path is applicable to
  bidirectional TE Tunnels.

- candidate-secondary-paths: A YANG container that holds a list
  of candidate secondary paths which may be used for the primary
  path to support path protection. The candidate secondary
  path(s) reference path(s) from the tunnel secondary paths list.
  The preference of the secondary paths is specified within the
  list and dictates the order of visiting the secondary path from
  the list. The attributes of a secondary path can be defined
  separately from the primary path. The attributes of a
  secondary path will be inherited from the associated 'active'
  primary when not explicitly defined for the secondary path.

secondary-paths:

   A YANG container that holds the set of secondary paths. A
   secondary path is identified by 'name'. A secondary path can be
   referenced from the TE Tunnel's 'candidate-secondary-path' list.

secondary-reverse-paths:

   A YANG container that holds the set of secondary reverse paths. A
   secondary reverse path is identified by 'name'. A secondary
   reverse path can be referenced from the TE Tunnel's 'candidate-
   secondary-reverse-paths' list. A secondary reverse path contains
   attributes similar to a primary path.

The following set of common path attributes are shared for primary
(forward and reverse) and secondary paths:

path-computation-method:

   A YANG leaf that specifies the method used for computing the TE
   path.

path-computation-server:

   A YANG container that holds the path computation server properties
   when the path is externally queried.

compute-only:

   A path of a TE Tunnel is, by default, provisioned so that it can
   be instantiated in the forwarding plane so that it can carry
   traffic as soon as a valid path is computed. In some cases, a TE
   path may be configured only for the purpose of computing a path
   and reporting it without the need to instantiate the LSP or commit
   any resources. In such a case, the path is configured in
   'compute-only' mode to distinguish it from the default behavior.
   A 'compute-only' path is configured as a usual with the associated
   per path constraint(s) and properties on a device or TE
   controller. The device or TE controller computes the feasible
   path(s) subject to configured constraints. A client may query the
   'compute-only' computed path properties 'on-demand', or
   alternatively, can subscribe to be notified of computed path(s)
   and whenever the path properties change.

use-path-computation:

   A YANG leaf that indicates whether or not path computation is to
   be used for a specified path.

lockdown:

   A YANG leaf that when set indicates the existing path should not
   be reoptimized after a failure on any of its traversed links.

path-scope:

   A YANG leaf that specifies the path scope if segment or an end-to-
   end path.

preference:

   A YANG leaf that specifies the preference for the path. The lower
   the number higher the preference.

k-requested-paths:

   A YANG leaf that specifies the number of k-shortest-paths
   requested from the path computation server and returned sorted by
   its optimization objective.

association-objects:

   A YANG container that holds a list of tunnel association
   properties.

optimizations:

   A YANG container that holds the optimization objectives that path
   computation will use to select a path.

named-path-constraint:

   A YANG leafref that references an entry from the global list of
   named path constraints.

te-bandwidth:

   A YANG container that holds the path bandwidth (see
   [I-D.draft-ietf-teas-rfc8776-update]).

link-protection:

   A YANG leaf that specifies the link protection type required for
   the links to be included the computed path (see
   [I-D.draft-ietf-teas-rfc8776-update]).

setup/hold-priority:

   see description provided in Section 5.1.1. These values override
   those provided in the referenced named-path-constraint.

signaling-type:

   see description provided in Section 5.1.1. This value overrides
   the provided one in the referenced named-path-constraint.

path-metric-bounds:

   see description provided in Section 5.1.1. These values override
   those provided in the referenced named-path-constraint.

path-affinities-values:

   see description provided in Section 5.1.1. These values override
   those provided in the referenced named-path-constraint.

path-affinity-names:

   see description provided in Section 5.1.1. These values override
   those provided in the referenced named-path-constraint.

path-srlgs-lists:

see description provided in Section 5.1.1. These values override
those provided in the referenced named-path-constraint.

path-srlgs-names:

see description provided in Section 5.1.1. These values override
those provided in the referenced named-path-constraint.

disjointness:

see description provided in Section 5.1.1. These values override
those provided in the referenced named-path-constraint.

explicit-route-objects:

see description provided in Section 5.1.1. These values override
those provided in the referenced named-path-constraint.

path-in-segment:

see description provided in Section 5.1.1. These values override
those provided in the referenced named-path-constraint.

path-out-segment:

see description provided in Section 5.1.1. These values override
those provided in the referenced named-path-constraint.

computed-paths-properties:

A YANG container that holds properties for the list of computed
paths.

computed-path-error-infos:

A YANG container that holds a list of errors related to the path.

lsp-provisioning-error-infos:

A YANG container that holds the list of LSP provisioning error
information. The TE system populates entries in this list
whenever an error is encountered during the LSP provisioning.

computed-path-error-infos:

A YANG container that holds the list of path computation error
information. The TE system populates entries in this list
whenever an error is encountered during the compuation of the TE
path.

path-compute-info:

A YANG grouping that contains leafs representing the path
attributes that are passed to the TE path computation engine to be
considered during the path computation. This includes:

- path constraints,

- path optimization objectives, and

- path assocociations

Note, unless overriden under a specific path of the TE tunnel, the
TE tunnel's primary path constraints, optimization objectives, and
associations are inherited by the primary reverse path, secondary
path and secondary reverse path.

lsps:

A YANG container that holds a list of LSPs that have been
instantiated for this specific path.

In addition to the path common attributes, the primary path has the
following attributes that are not present in the secondary path:

* Only the primary path contains the list of 'candidate-secondary-
  paths' that can protect the primary path.

* Only the primary path can contain a primary-reverse-path
  associated with the primary path (and its associated list of
  'candidate-secondary-reverse-path').

5.1.3. TE LSPs

The 'lsps' container includes the set of TE LSP(s) that have been
instantiated. A TE LSP is identified by a 3-tuple ('tunnel-name',
'lsp-id', 'node').

When the model is used to manage a specific device, the 'lsps' list
contains all TE LSP(s) that traverse the device (including
ingressing, transiting and egressing the device).

   When the model is used to manage a TE controller, the 'lsps' list
   contains the TE LSP(s) on devices managed by the controller that act
   as ingress, and may optionally include TE LSPs on devices managed by
   the controller that act as transit or egress role.

5.2. Tree Diagram

   Figure 6 shows the tree diagram of depth=4 for the generic TE YANG
   model defined in modules 'ietf-te.yang'. The full tree diagram is
   shown in Section 13.

```
   module: ietf-te
     +--rw te
        +--rw enable?    boolean
        +--rw globals
        |  +--rw named-admin-groups
        |  |  +--rw named-admin-group* [name]
        |  |          {te-types:extended-admin-groups,
        |  |           te-types:named-extended-admin-groups}?
        |  |     ...
        |  +--rw named-srlgs
        |  |  +--rw named-srlg* [name] {te-types:named-srlg-groups}?
        |  |     ...
        |  +--rw named-path-constraints
        |     +--rw named-path-constraint* [name]
        |             {te-types:named-path-constraints}?
        |        ...
        +--rw tunnels
        |  +--rw tunnel* [name]
        |     +--rw name                          string
        |     +--rw alias?                        string
        |     +--rw identifier?                   uint32
        |     +--rw color?                        uint32
        |     +--rw description?                  string
        |     +--rw admin-state?                  identityref
        |     +--ro operational-state?            identityref
        |     +--rw encoding?                     identityref
        |     +--rw switching-type?               identityref
        |     +--rw source
        |     |     ...
        |     +--rw destination
        |     |     ...
        |     +--rw bidirectional?                boolean
        |     +--rw controller
        |     |     ...
        |     +--rw reoptimize-timer?             uint16
        |     +--rw association-objects
        |     |     ...
```

```
            │     +--rw protection
            │     │    ...
            │     +--rw restoration
            │     │    ...
            │     +--rw network-id?                    nw:network-id
            │     +--rw te-topology-identifier
            │     │    ...
            │     +--rw te-bandwidth
            │     │    ...
            │     +--rw link-protection?               identityref
            │     +--rw setup-priority?                uint8
            │     +--rw hold-priority?                 uint8
            │     +--rw signaling-type?                identityref
            │     +--rw hierarchy
            │     │    ...
            │     +--rw primary-paths
            │     │    ...
            │     +--rw secondary-paths
            │     │    ...
            │     +--rw secondary-reverse-paths
            │     │    ...
            │     +---x tunnel-action
            │     │    ...
            │     +---x protection-external-commands
            │          ...
            +--ro lsps
               +--ro lsp* [tunnel-name lsp-id node]
                  +--ro tunnel-name                    string
                  +--ro lsp-id                         uint16
                  +--ro node
                  │       te-types:te-node-id
                  +--ro source?
                  │       te-types:te-node-id
                  +--ro destination?
                  │       te-types:te-node-id
                  +--ro tunnel-id?                     uint16
                  +--ro extended-tunnel-id?
                  │    yang:dotted-quad
                  +--ro operational-state?             identityref
                  +--ro signaling-type?                identityref
                  +--ro origin-type?                   enumeration
                  +--ro lsp-resource-status?           enumeration
                  +--ro lockout-of-normal?             boolean
                  +--ro freeze?                        boolean
                  +--ro lsp-protection-role?           enumeration
                  +--ro lsp-protection-state?          identityref
                  +--ro protection-group-ingress-node-id?
                  │       te-types:te-node-id
```

```
                  +--ro protection-group-egress-node-id?
                  |        te-types:te-node-id
                  +--ro lsp-actual-route-information
                      ...

     rpcs:
       +---x tunnels-path-compute
       |  +---w input
       |  |  +---w path-compute-info
       |  +--ro output
       |     +--ro path-compute-result
       +---x tunnels-actions
          +---w input
          |  +---w tunnel-info
          |  |  +---w (filter-type)
          |  |     ...
          |  +---w action-info
          |     +---w action?       identityref
          |     +---w disruptive?   empty
          +--ro output
             +--ro action-result?   identityref
```

         Figure 6: Tree diagram of depth-4 of TE Tunnel YANG data model

5.3. YANG Module

   The generic TE YANG module 'ietf-te' imports the following modules:

   * ietf-te-types defined in [I-D.draft-ietf-teas-rfc8776-update]

   * ietf-yang-types and ietf-inet-types defined in [RFC6991]

   * ietf-network and ietf-network-topology defined in [RFC8345]

   This module references the following documents: [RFC4206], [RFC4427],
   [RFC4872], [RFC3209], [RFC6780], [RFC7471], [RFC9012], [RFC8570],
   [RFC8232], [RFC7271], [RFC8234], [RFC7308], and [ITU_G.808.1].

```
   <CODE BEGINS> file "ietf-te@2024-02-02.yang"
   module ietf-te {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-te";

     /* Replace with IANA when assigned */

     prefix te;

     /* Import TE generic types */
```

```
  import ietf-te-types {
    prefix te-types;
    reference
      "draft-ietf-teas-rfc8776-update: Common YANG Data Types
       for Traffic Engineering.";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC6991: Common YANG Data Types.";
  }

  import ietf-network {
    prefix "nw";
    reference "RFC 8345: A YANG Data Model for Network Topologies";
  }

  import ietf-network-topology {
    prefix "nt";
    reference "RFC 8345: A YANG Data Model for Network Topologies";
  }

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

     Editor:    Tarek Saad
                <mailto:tsaad.net@gmail.com>

     Editor:    Rakesh Gandhi
                <mailto:rgandhi@cisco.com>

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

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

     Editor:    Xufeng Liu
                <mailto: xufeng.liu.ietf@gmail.com>

     Editor:    Igor Bryskin
                <mailto:i_bryskin@yahoo.com>

     Editor:    Oscar Gonzalez de Dios
```

```
                 <mailto: oscar.gonzalezdedios@telefonica.com>";

      description
        "YANG data module for TE configuration, state, and RPCs.
         The model fully conforms to the Network Management
         Datastore Architecture (NMDA).

         Copyright (c) 2023 IETF Trust and the persons identified as
         authors of the code.  All rights reserved.

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

         This version of this YANG module is part of RFC XXXX
         (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
         for full legal notices.";

      // RFC Ed.: replace XXXX with actual RFC number and remove this
      // note.
      // RFC Ed.: update the date below with the date of RFC publication
      // and remove this note.

      revision 2024-02-02 {
        description
          "Initial revision for the TE generic YANG module.";
        reference
          "RFCXXXX: A YANG Data Model for Traffic Engineering Tunnels
           and Interfaces.";
      }

      typedef tunnel-ref {
        type leafref {
          path "/te:te/te:tunnels/te:tunnel/te:name";
          require-instance false;
        }
        description
          "This type is used by data models that need to reference
           configured TE tunnel.";
      }

      /**
       * TE tunnel generic groupings
       */
```

```
     grouping path-common-properties {
       description
         "Common path attributes.";
       leaf name {
         type string;
         description
           "TE path name.";
       }
       leaf path-computation-method {
         type identityref {
           base te-types:path-computation-method;
         }
         default "te-types:path-locally-computed";
         description
           "The method used for computing the path, either
            locally computed, queried from a server or not
            computed at all (explicitly configured).";
       }
       container path-computation-server {
         when "derived-from-or-self(../path-computation-method, "
           + "'te-types:path-externally-queried')" {
           description
            "The path-computation server when the path is
             externally queried.";
         }
         uses te-types:te-generic-node-id;
         description
           "Address of the external path computation
            server.";
       }
       leaf compute-only {
         type empty;
         description
           "When present, the path is computed and updated whenever
            the topology is updated. No resources are committed
            or reserved in the network.";
       }
       leaf use-path-computation {
         when "derived-from-or-self(../path-computation-method, "
           + "'te-types:path-locally-computed')";
         type boolean;
         default "true";
         description
           "When 'true' indicates the path is dynamically computed
            and/or validated against the Traffic-Engineering Database
            (TED), and when 'false' indicates no path expansion or
            validation against the TED is required.";
       }
```

```
      leaf lockdown {
        type empty;
        description
          "When present, indicates no reoptimization to be attempted
           for this path.";
      }
      leaf path-scope {
        type identityref {
          base te-types:path-scope-type;
        }
        default "te-types:path-scope-end-to-end";
        config false;
        description
          "Indicates whether the path is a segment or portion of
           of the full path., or is the an end-to-end path for
           the TE Tunnel.";
      }
    }

    /* This grouping is re-used in path-computation rpc */
    grouping path-compute-info {
      description
        "Attributes used for path computation request.";
      uses tunnel-associations-properties;
      uses te-types:generic-path-optimization;
      leaf named-path-constraint {
        if-feature "te-types:named-path-constraints";
        type leafref {
          path "/te:te/te:globals/te:named-path-constraints/"
             + "te:named-path-constraint/te:name";
        }
        description
          "Reference to a globally defined named path constraint set.";
      }
      uses path-constraints-common;
    }

    /* This grouping is re-used in path-computation rpc */
    grouping path-forward-properties {
      description
        "The path preference.";
      leaf preference {
        type uint8 {
          range "1..255";
        }
        default "1";
        description
          "Specifies a preference for this path. The lower the number
```

```
           higher the preference.";
       }
       leaf co-routed {
         when "/te:te/te:tunnels/te:tunnel/te:bidirectional = 'true'" {
           description
             "Applicable to bidirectional tunnels only.";
         }
         type boolean;
         default "false";
         description
           "Indicates whether the reverse path must to be co-routed
            with the primary.";
       }
     }

     /* This grouping is re-used in path-computation rpc */
     grouping k-requested-paths {
       description
         "The k-shortest paths requests.";
       leaf k-requested-paths {
         type uint8;
         default "1";
         description
           "The number of k-shortest-paths requested from the path
            computation server and returned sorted by its optimization
            objective.";
       }
     }

     grouping path-state {
       description
         "TE per path state parameters.";
       uses path-computation-response;
       container lsp-provisioning-error-infos {
         config false;
         description
           "LSP provisioning error information.";
         list lsp-provisioning-error-info {
           description
             "List of LSP provisioning error info entries.";
           leaf error-reason {
             type identityref {
               base te-types:lsp-provisioning-error-reason;
             }
             description
               "LSP provision error type.";
           }
           leaf error-description {
```

```
              type string;
              description
                "The textual representation of the error occurred during
                 path computation.";
            }
            leaf error-timestamp {
              type yang:date-and-time;
              description
                "Timestamp of when the reported error occurred.";
            }
            leaf error-node-id {
              type te-types:te-node-id;
              description
                "Node identifier of node where error occurred.";
            }
            leaf error-link-id {
              type te-types:te-tp-id;
              description
                "Link ID where the error occurred.";
            }
            leaf lsp-id {
              type uint16;
              description
                "The LSP-ID for which path computation was performed.";
            }
          }
        }
        container lsps {
          config false;
          description
            "The TE LSPs container.";
          list lsp {
            key "node lsp-id";
            description
              "List of LSPs associated with the tunnel.";
            leaf tunnel-name {
              type leafref {
                path "/te:te/te:lsps/te:lsp/te:tunnel-name";
              }
              description "TE tunnel name.";
            }
            leaf node {
              type leafref {
                path "/te:te/te:lsps/te:lsp[tunnel-name="
                    + "current()/../te:tunnel-name][lsp-id="
                    + "current()/../te:lsp-id]/te:node";
              }
              description "The node where the LSP state resides on.";
```

```
            }
          leaf lsp-id {
            type leafref {
              path "/te:te/te:lsps/te:lsp[tunnel-name="
                  + "current()/../tunnel-name]/te:lsp-id";
            }
            description "The TE LSP identifier.";
          }
        }
      }
    }

    /* This grouping is re-used in path-computation rpc */
    grouping path-computation-response {
      description
        "Attributes reported by path computation response.";
      container computed-paths-properties {
        config false;
        description
          "Computed path properties container.";
        list computed-path-properties {
          key "k-index";
          description
            "List of computed paths.";
          leaf k-index {
            type uint8;
            description
              "The k-th path returned from the computation server.
               A lower k value path is more optimal than higher k
               value path(s)";
          }
          uses te-types:generic-path-properties {
            augment "path-properties" {
              description
                "additional path properties returned by path
                 computation.";
              uses te-types:te-bandwidth;
              leaf disjointness-type {
                type te-types:te-path-disjointness;
                config false;
                description
                  "The type of resource disjointness.
                   When reported for a primary path, it represents the
                   minimum level of disjointness of all the secondary
                   paths. When reported for a secondary path, it
                   represents the disjointness of the secondary path.";
              }
            }
```

```
            }
          }
        }
        container computed-path-error-infos {
          config false;
          description
            "Path computation information container.";
          list computed-path-error-info {
            description
              "List of path computation info entries.";
            leaf error-description {
              type string;
              description
                "Textual representation of the error that occurred
                 during path computation.";
            }
            leaf error-timestamp {
              type yang:date-and-time;
              description
                "Timestamp of last path computation attempt.";
            }
            leaf error-reason {
              type identityref {
                base te-types:path-computation-error-reason;
              }
              description
                "Reason for the path computation error.";
            }
          }
        }
      }

      grouping protection-restoration-properties {
        description
          "Protection and restoration parameters.";
        container protection {
          description
            "Protection parameters.";
          leaf protection-type {
            type identityref {
              base te-types:lsp-protection-type;
            }
            default "te-types:lsp-protection-unprotected";
            description
              "LSP protection type.";
          }
          leaf protection-reversion-disable {
            type boolean;
```

```
              default "false";
              description
                "Disable protection reversion to working path.";
            }
            leaf hold-off-time {
              type uint32;
              units "milli-seconds";
              description
                "The time between the declaration of an SF or SD condition
                 and the initialization of the protection switching
                 algorithm.";
              reference
                "RFC4427";
            }
            leaf wait-to-revert {
              type uint16;
              units "seconds";
              description
                "Time to wait before attempting LSP reversion.";
              reference
                "RFC4427";
            }
            leaf aps-signal-id {
              type uint8 {
                range "1..255";
              }
              default "1";
              description
                "The APS signal number used to reference the traffic of
                 this tunnel. The default value for normal traffic is 1.
                 The default value for extra-traffic is 255. If not
                 specified, non-default values can be assigned by the
                 server, if and only if, the server controls both
                  endpoints.";
              reference
                "ITU_G.808.1";
            }
          }
          container restoration {
            description
              "Restoration parameters.";
            leaf restoration-type {
              type identityref {
                base te-types:lsp-restoration-type;
              }
              description
                "LSP restoration type.";
            }
```

```
      leaf restoration-scheme {
        type identityref {
          base te-types:restoration-scheme-type;
        }
        description
          "LSP restoration scheme.";
      }
      leaf restoration-reversion-disable {
        type boolean;
        default "false";
        description
          "Disable restoration reversion to working path.";
      }
      leaf hold-off-time {
        type uint32;
        units "milli-seconds";
        description
          "The time between the declaration of an SF or SD condition
           and the initialization of the protection switching
           algorithm.";
        reference
          "RFC4427";
      }
      leaf wait-to-restore {
        type uint16;
        units "seconds";
        description
          "Time to wait before attempting LSP restoration.";
        reference
          "RFC4427";
      }
      leaf wait-to-revert {
        type uint16;
        units "seconds";
        description
          "Time to wait before attempting LSP reversion.";
        reference
          "RFC4427";
      }
    }
  }

  grouping tunnel-associations-properties {
    description
      "TE tunnel association grouping.";
    container association-objects {
      description
        "TE tunnel associations.";
```

```
        list association-object {
          key "association-key";
          unique "type id source/id source/type";
          description
            "List of association base objects.";
          reference
            "RFC4872";
          leaf association-key {
            type string;
            description
              "Association key used to identify a specific
               association in the list";
          }
          leaf type {
            type identityref {
              base te-types:association-type;
            }
            description
              "Association type.";
            reference
              "RFC4872";
          }
          leaf id {
            type uint16;
            description
              "Association identifier.";
            reference
              "RFC4872";
          }
          container source {
            uses te-types:te-generic-node-id;
            description
              "Association source.";
            reference
              "RFC4872";
          }
        }
        list association-object-extended {
          key "association-key";
          unique
            "type id source/id source/type global-source extended-id";
          description
            "List of extended association objects.";
          reference
            "RFC6780";
          leaf association-key {
            type string;
            description
```

```
                "Association key used to identify a specific
                 association in the list";
            }
            leaf type {
              type identityref {
                base te-types:association-type;
              }
              description
                "Association type.";
              reference
                "RFC4872, RFC6780";
            }
            leaf id {
              type uint16;
              description
                "Association identifier.";
              reference
                "RFC4872, RFC6780";
            }
            container source {
              uses te-types:te-generic-node-id;
              description
                "Association source.";
              reference
                "RFC4872, RFC6780";
            }
            leaf global-source {
              type uint32;
              description
                "Association global source.";
              reference
                "RFC6780";
            }
            leaf extended-id {
              type yang:hex-string;
              description
                "Association extended identifier.";
              reference
                "RFC6780";
            }
          }
        }
      }

      grouping tunnel-end-point {
        description
          "Common grouping used to specify the tunnel source and
          destination end-points.";
```

```
      leaf node-id {
        type nw:node-id;
        description
          "The TE tunnel end-point node identifier";
      }
      leaf te-node-id {
        type te-types:te-node-id;
        description
          "The TE tunnel end-point TE node identifier";
      }
      leaf tunnel-tp-id {
        when "../node-id or ../te-node-id" {
          description
            "The TE tunnel termination point identifier is local to
            a node";
        }
        type binary;
        description
          "The TE tunnel end-point TE tunnel termination point
          identifier";
      }
    }

    /* This grouping is re-used in path-computation rpc */
    grouping tunnel-common-attributes {
      description
        "Common grouping to define the TE tunnel parameters";
      container source {
        description
          "TE tunnel source end-point.";
        uses tunnel-end-point;
      }
      container destination {
        description
          "TE tunnel destination end-point.";
        uses tunnel-end-point;
      }
      leaf bidirectional {
        type boolean;
        default "false";
        description
          "Indicates a bidirectional tunnel";
      }
    }

    /* This grouping is re-used in path-computation rpc */
    grouping tunnel-hierarchy-properties {
      description
```

```
           "A grouping for TE tunnel hierarchy information.";
        container hierarchy {
          description
            "Container for TE hierarchy related information.";
          container dependency-tunnels {
            description
              "List of tunnels that this tunnel can be potentially
               dependent on.";
            list dependency-tunnel {
              key "name";
              description
                "A tunnel entry that this tunnel can potentially depend
                 on.";
              leaf name {
                type tunnel-ref;
                description
                  "Dependency tunnel name. The tunnel may not have been
                   instantiated yet.";
              }
              uses te-types:encoding-and-switching-type;
            }
          }
          container hierarchical-link {
            description
              "Identifies a hierarchical link (in client layer)
               that this tunnel is associated with. By default, the
               topology of the hierarchical link is the same topology of
               the tunnel;";
            reference
              "RFC4206";
            leaf enable {
              type boolean;
              default "false";
              description
                "Enables the hierarchical link properties supported by
                 this tunnel";
            }
            leaf local-node-id {
              type nw:node-id;
              description
                "The local node identifier.";
            }
            leaf local-te-node-id {
              type te-types:te-node-id;
              description
                "The local TE node identifier.";
            }
            leaf local-link-tp-id {
```

```
              type nt:tp-id;
              description
                "The local link termination point identifier.";
              reference
                "RFC8345";
            }
            leaf local-te-link-tp-id {
              type te-types:te-tp-id;
              description
                "The local TE link termination point identifier.";
            }
            leaf remote-node-id {
              type nw:node-id;
              description
                "The remote node identifier.";
            }
            leaf remote-link-tp-id {
              type nt:tp-id;
              description
                "The remote link termination point identifier.";
              reference
                "RFC8345";
            }
            leaf remote-te-link-tp-id {
              type te-types:te-tp-id;
              description
                "The remote TE link termination point identifier.";
            }
            leaf remote-te-node-id {
              type te-types:te-node-id;
              description
                "Remote TE node identifier.";
            }
            leaf link-id {
              type nt:link-id;
              config false;
              description
                "A network topology assigned identifier to the link";
              reference
                "RFC8345";
            }
            leaf network-id {
              type nw:network-id;
              description
                "The network topology identifier where the hierarchical
                link supported by this TE tunnel is instantiated.";
            }
            uses te-types:te-topology-identifier {
```

```
            description
              "The TE topology identifier where the hierarchical link
               supported by this TE tunnel is instantiated.";
          }
        }
      }
    }

    grouping path-constraints-common {
      description
        "Global named path constraints configuration
         grouping.";
      uses te-types:common-path-constraints-attributes;
      uses te-types:generic-path-disjointness;
      uses te-types:path-constraints-route-objects;
      container path-in-segment {
        presence "The end-to-end tunnel starts in a previous domain;
                  this tunnel is a segment in the current domain.";
        description
          "If an end-to-end tunnel crosses multiple domains using
           the same technology, some additional constraints have to be
           taken in consideration in each domain.
           This TE tunnel segment is stitched to the upstream TE tunnel
           segment.";
        uses te-types:label-set-info;
      }
      container path-out-segment {
        presence
          "The end-to-end tunnel is not terminated in this domain;
           this tunnel is a segment in the current domain.";
        description
          "If an end-to-end tunnel crosses multiple domains using
           the same technology, some additional constraints have to be
           taken in consideration in each domain.
           This TE tunnel segment is stitched to the downstream TE
           tunnel segment.";
        uses te-types:label-set-info;
      }
    }

    /**
     * TE container
     */

    container te {
      description
        "TE global container.";
      leaf enable {
```

```
      type boolean;
      description
        "Enables the TE component features.";
    }

    /* TE Global Data */
    container globals {
      description
        "Globals TE system-wide configuration data container.";
      container named-admin-groups {
        description
          "TE named admin groups container.";
        list named-admin-group {
          if-feature "te-types:extended-admin-groups";
          if-feature "te-types:named-extended-admin-groups";
          key "name";
          description
            "List of named TE admin-groups.";
          leaf name {
            type string;
            description
              "A string name that uniquely identifies a TE
               interface named admin-group.";
          }
          leaf bit-position {
            type uint32;
            description
              "Bit position representing the administrative group.";
            reference
              "RFC3209 and RFC7308";
          }


        }
      }
      container named-srlgs {
        description
          "TE named SRLGs container.";
        list named-srlg {
          if-feature "te-types:named-srlg-groups";
          key "name";
          description
            "A list of named SRLG groups.";
          leaf name {
            type string;
            description
              "A string name that uniquely identifies a TE
               interface named SRLG.";
```

```
          }
          leaf value {
            type te-types:srlg;
            description
              "An SRLG value.";
          }
          leaf cost {
            type uint32;
            description
              "SRLG associated cost. Used during path to append
               the path cost when traversing a link with this SRLG.";
          }
        }
      }
      container named-path-constraints {
        description
          "TE named path constraints container.";
        list named-path-constraint {
          if-feature "te-types:named-path-constraints";
          key "name";
          leaf name {
            type string;
            description
              "A string name that uniquely identifies a
               path constraint set.";
          }
          uses path-constraints-common;
          description
            "A list of named path constraints.";
        }
      }
    }

    /* TE Tunnel Data */
    container tunnels {
      description
        "Tunnels TE configuration data container.";
      list tunnel {
        key "name";
        description
          "The list of TE tunnels.";
        leaf name {
          type string;
          description
            "TE tunnel name.";
        }
        leaf alias {
          type string;
```

```
              description
                "An alternate name of the TE tunnel that can be modified
                 anytime during its lifetime.";
            }
            leaf identifier {
              type uint32;
              description
                "TE tunnel Identifier.";
              reference
                "RFC3209";
            }
            leaf color {
              type uint32;
              description "The color associated with the TE tunnel.";
              reference "RFC9012";
            }
            leaf description {
              type string;
              default "None";
              description
                "Textual description for this TE tunnel.";
            }
            leaf admin-state {
              type identityref {
                base te-types:tunnel-admin-state-type;
              }
              default "te-types:tunnel-admin-state-up";
              description
                "TE tunnel administrative state.";
            }
            leaf operational-state {
              type identityref {
                base te-types:tunnel-state-type;
              }
              config false;
              description
                "TE tunnel operational state.";
            }
            uses te-types:encoding-and-switching-type;
            uses tunnel-common-attributes;
            container controller {
              description
                "Contains tunnel data relevant to external controller(s).
                 This target node may be augmented by external module(s),
                 for example, to add data for PCEP initiated and/or
                 delegated tunnels.";
              leaf protocol-origin {
                type identityref {
```

```
                  base te-types:protocol-origin-type;
                }
                description
                  "The protocol origin for instantiating the tunnel.";
              }
              leaf controller-entity-id {
                type string;
                description
                  "An identifier unique within the scope of visibility
                   that associated with the entity that controls the
                   tunnel.";
                reference "RFC8232";
              }
            }
            leaf reoptimize-timer {
              type uint16;
              units "seconds";
              description
                "Frequency of reoptimization of a traffic engineered
                 LSP.";
            }
            uses tunnel-associations-properties;
            uses protection-restoration-properties;
            uses te-types:tunnel-constraints;
            uses tunnel-hierarchy-properties;
            container primary-paths {
              description
                "The set of primary paths.";
              reference "RFC4872";
              list primary-path {
                key "name";
                description
                  "List of primary paths for this tunnel.";
                leaf active {
                  type boolean;
                  config false;
                  description
                    "Indicates an active path that
                     has been selected from the primary paths list.";
                }
                uses path-common-properties;
                uses path-forward-properties;
                uses k-requested-paths;
                uses path-compute-info;
                uses path-state;
                container primary-reverse-path {
                  when "../../../te:bidirectional = 'true'";
                  description
```

```
                    "The reverse primary path properties.";
                uses path-common-properties;
                uses path-compute-info;
                uses path-state;
                container candidate-secondary-reverse-paths {
                  description
                    "The set of referenced candidate reverse secondary
                     paths from the full set of secondary reverse paths
                     which may be used for this primary path.";
                  list candidate-secondary-reverse-path {
                    key "secondary-reverse-path";
                    ordered-by user;
                    description
                      "List of candidate secondary reverse path(s)";
                    leaf secondary-reverse-path {
                      type leafref {
                        path "../../../../../../"
                           + "te:secondary-reverse-paths/"
                           + "te:secondary-reverse-path/te:name";
                      }
                      description
                        "A reference to the secondary reverse path that
                         may be utilized when the containing primary
                         reverse path is in use.";
                    }
                    leaf active {
                      type boolean;
                      config false;
                      description
                        "Indicates an active path that has been
                         selected from the secondary reverse paths
                         list.";
                    }
                  }
                }
              }
              container candidate-secondary-paths {
                description
                  "The set of candidate secondary paths which may be
                   used for this primary path. When secondary paths are
                   specified in the list the path of the secondary LSP
                   in use must be restricted to those paths
                   referenced.
                   The priority of the secondary paths is specified
                   within the list. Higher priority values are less
                   preferred - that is to say that a path with priority
                   0 is the most preferred path. In the case that the
                   list is empty, any secondary path may be
```

```
                    utilised when the current primary path is in use.";
                  list candidate-secondary-path {
                    key "secondary-path";
                    ordered-by user;
                    description
                      "List of candidate secondary paths for this
                       tunnel.";
                    leaf secondary-path {
                      type leafref {
                        path "../../../../../te:secondary-paths/"
                          + "te:secondary-path/te:name";
                      }
                      description
                        "A reference to the secondary path that may be
                         utilised when the containing primary path is
                         in use.";
                    }
                    leaf active {
                      type boolean;
                      config false;
                      description
                        "Indicates an active path that has been selected
                         from the candidate secondary paths.";
                    }
                  }
                }
              }
            }
            container secondary-paths {
              description
                "The set of secondary paths.";
              reference "RFC4872";
              list secondary-path {
                key "name";
                description
                  "List of secondary paths for this tunnel.";
                uses path-common-properties;
                leaf preference {
                  type uint8 {
                    range "1..255";
                  }
                  default "1";
                  description
                    "Specifies a preference for this path. The lower the
                     number higher the preference.";
                }
                leaf secondary-reverse-path {
                  type leafref {
```

```
                     path "../../../"
                         + "te:secondary-reverse-paths/"
                         + "te:secondary-reverse-path/te:name";
                   }
                   description
                     "A reference to the reverse secondary path when
                      co-routed with the secondary path.";
                 }
               uses path-compute-info;
               uses protection-restoration-properties;
               uses path-state;
             }
           }
           container secondary-reverse-paths {
             description
               "The set of secondary reverse paths.";
             list secondary-reverse-path {
               key "name";
               description
                 "List of secondary paths for this tunnel.";
               uses path-common-properties;
               leaf preference {
                 type uint8 {
                   range "1..255";
                 }
                 default "1";
                 description
                   "Specifies a preference for this path. The lower the
                    number higher the preference. Paths that have the
                    same preference will be activated together.";
               }
               uses path-compute-info;
               uses protection-restoration-properties;
               uses path-state;
             }
           }
           action tunnel-action {
             description
               "Action commands to manipulate the TE tunnel state.";
             reference
               "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
                Section 2.5";
             input {
               leaf action-type {
                 type identityref {
                   base te-types:tunnel-action-type;
                 }
                 description
```

```
                      "The action to be invoked on the TE tunnel.";
                }
              }
              output {
                leaf action-result {
                  type identityref {
                    base te-types:te-action-result;
                  }
                  description
                    "The result of the tunnel action operation.";
                }
              }
            }
            action protection-external-commands {
              description
                "Actions to manipulate the protection external
                 commands of the TE tunnel.";
              reference
                "RFC 4427: Recovery (Protection and Restoration)
                 Terminology for Generalized Multi-Protocol Label
                 Switching (GMPLS)";
              input {
                leaf protection-external-command {
                  type identityref {
                    base te-types:protection-external-commands;
                  }
                  description
                    "Protection external command.";
                }
                leaf protection-group-ingress-node {
                  type boolean;
                  default "true";
                  description
                    "When 'true', indicates that the action is
                     applied on ingress node.
                     By default, the action applies to the ingress node
                     only.";
                }
                leaf protection-group-egress-node {
                  type boolean;
                  default "false";
                  description
                    "When set to 'true', indicates that the action is
                     applied on egress node.
                     By default, the action applies to the ingress node
                     only.";
                }
                leaf path-name {
```

```
                      type string;
                      description
                        "The name of the path that the external command
                        applies to.";
                    }
                    leaf path-type {
                      type te-types:path-type;
                      description
                        "The type of the path that the external command
                        applies to.";
                    }
                    leaf traffic-type {
                      type enumeration {
                        enum normal-traffic {
                          description
                            "The manual-switch or forced-switch command
                             applies to the normal traffic (this Tunnel).";
                        }
                        enum null-traffic {
                          description
                            "The manual-switch or forced-switch command
                             applies to the null traffic.";
                        }
                        enum extra-traffic {
                          description
                            "The manual-switch or forced-switch command
                             applies to the extra traffic (the extra-traffic
                             Tunnel sharing protection bandwidth with this
                             Tunnel).";
                        }
                      }
                      description
                        "Indicates whether the manual-switch or forced-switch
                         commands applies to the normal traffic, the null
                         traffic or the extra-traffic.";
                      reference
                        "RFC4427";
                    }
                    leaf extra-traffic-tunnel-ref {
                      type tunnel-ref;
                      description
                        "In case there are multiple extra-traffic tunnels
                         sharing protection bandwidth with this Tunnel
                         (m:n protection), represents which extra-traffic
                         Tunnel the manual-switch or forced-switch to
                         extra-traffic command applies to.";
                    }
                  }
```

```
          }
        }
      }

      /* TE LSPs Data */
      container lsps {
        config false;
        description
          "TE LSPs state container.";
        list lsp {
          key "tunnel-name lsp-id node";
          unique "source destination tunnel-id lsp-id "
            + "extended-tunnel-id";
          description
            "List of LSPs associated with the tunnel.";
          leaf tunnel-name {
            type string;
            description "The TE tunnel name.";
          }
          leaf lsp-id {
            type uint16;
            description
              "Identifier used in the SENDER_TEMPLATE and the
               FILTER_SPEC that can be changed to allow a sender to
               share resources with itself.";
            reference
              "RFC3209";
          }
          leaf node {
            type te-types:te-node-id;
            description
              "The node where the TE LSP state resides on.";
          }
          leaf source {
            type te-types:te-node-id;
            description
              "Tunnel sender address extracted from
               SENDER_TEMPLATE  object.";
            reference
              "RFC3209";
          }
          leaf destination {
            type te-types:te-node-id;
            description
              "The tunnel endpoint address.";
            reference
              "RFC3209";
          }
```

```
            leaf tunnel-id {
              type uint16;
              description
                "The tunnel identifier that remains
                 constant over the life of the tunnel.";
              reference
                "RFC3209";
            }
            leaf extended-tunnel-id {
              type yang:dotted-quad;
              description
                "The LSP Extended Tunnel ID.";
              reference
                "RFC3209";
            }
            leaf operational-state {
              type identityref {
                base te-types:lsp-state-type;
              }
              description
                "The LSP operational state.";
            }
            leaf signaling-type {
              type identityref {
                base te-types:path-signaling-type;
              }
              description
                "The signaling protocol used to set up this LSP.";
            }
            leaf origin-type {
              type enumeration {
                enum ingress {
                  description
                    "Origin ingress.";
                }
                enum egress {
                  description
                    "Origin egress.";
                }
                enum transit {
                  description
                    "Origin transit.";
                }
              }
              description
                "The origin of the LSP relative to the location of the
                 local switch in the path.";
            }
```

```
          leaf lsp-resource-status {
            type enumeration {
              enum primary {
                description
                  "A primary LSP is a fully established LSP for which
                   the resource allocation has been committed at the
                   data plane.";
              }
              enum secondary {
                description
                  "A secondary LSP is an LSP that has been provisioned
                   in the control plane only; e.g. resource allocation
                   has not been committed at the data plane.";
              }
            }
            description
              "LSP resource allocation state.";
            reference
              "RFC4872, section 4.2.1";
          }
          leaf lockout-of-normal {
            type boolean;
            description
              "When set to 'true', it represents a lockout of normal
               traffic external command. When set to 'false', it
               represents a clear lockout of normal traffic external
               command. The lockout of normal traffic command applies
               to this Tunnel.";
            reference
              "RFC4427";
          }
          leaf freeze {
            type boolean;
            description
              "When set to 'true', it represents a freeze external
               command.  When set to 'false', it represents a clear
               freeze external command. The freeze command applies to
               all the Tunnels which are sharing the protection
               resources with this Tunnel.";
            reference
              "RFC4427";
          }
          leaf lsp-protection-role {
            type enumeration {
              enum working {
                description
                  "A working LSP must be a primary LSP whilst a
                   protecting LSP can be either a primary or a
```

```
                 secondary LSP. Also, known as protected LSPs when
                 working LSPs are associated with protecting LSPs.";
           }
           enum protecting {
             description
               "A secondary LSP is an LSP that has been provisioned
                in the control plane only; e.g. resource allocation
                has not been committed at the data plane.";
           }
         }
         description
           "LSP role type.";
         reference
           "RFC4872, section 4.2.1";
       }
       leaf lsp-protection-state {
         type identityref {
           base te-types:lsp-protection-state;
         }
         config false;
         description
           "The reported protection state controlling which
            tunnel is using the resources of the protecting LSP.";
       }
       leaf protection-group-ingress-node-id {
         type te-types:te-node-id;
         description
           "Indicates the te-node-id of the protection group
            ingress node when the APS state represents an external
            command (LoP, SF, MS) applied to it or a WTR timer
            running on it. If the external command is not applied to
            the ingress node or the WTR timer is not running on it,
            this attribute is not specified. A value 0.0.0.0 is used
            when the te-node-id of the protection group ingress node
            is unknown (e.g., because the ingress node is outside
            the scope of control of the server)";
       }
       leaf protection-group-egress-node-id {
         type te-types:te-node-id;
         description
           "Indicates the te-node-id of the protection group egress
            node when the APS state represents an external command
            (LoP, SF, MS) applied to it or a WTR timer running on
            it. If the external command is not applied to the
            ingress node or the WTR timer is not running on it, this
            attribute is not specified. A value 0.0.0.0 is used when
            the te-node-id of the protection group ingress node is
            unknown (e.g., because the ingress node is outside the
```

```
                   scope of control of the server)";
              }
            container lsp-actual-route-information {
              description
                "RSVP recorded route object information.";
              list lsp-actual-route-information {
                when "../../origin-type = 'ingress'" {
                  description
                    "Applicable on ingress LSPs only.";
                }
                key "index";
                description
                  "Record route list entry.";
                uses te-types:record-route-state;
              }
            }
          }
        }
      }
    }

    /* TE Tunnel RPCs/execution Data */

    rpc tunnels-path-compute {
      description
        "This RPC is a generic API whose
         input and output parameters are expected to be provided by
         augments to this module.";
      reference
        "RFC 4655: A Path Computation Element (PCE)-Based
         Architecture.";
      input {
        container path-compute-info {
          /*
           * An external path compute module may augment this
           * target.
           */
          description
            "RPC input information.";
        }
      }
      output {
        container path-compute-result {
          /*
           * An external path compute module may augment this
           * target.
           */
          description
            "RPC output information.";
```

```
              }
            }
          }

      rpc tunnels-actions {
        description
          "RPC that manipulates the state of a TE tunnel.";
        reference
          "RFC 3209: RSVP-TE: Extensions to RSVP for LSP Tunnels,
           Section 2.5";
        input {
          container tunnel-info {
            description
              "TE tunnel information.";
            choice filter-type {
              mandatory true;
              description
                "Filter choice.";
              case all-tunnels {
                leaf all {
                  type empty;
                  mandatory true;
                  description
                    "When present, applies the action on all TE
                     tunnels.";
                }
              }
              case one-tunnel {
                leaf tunnel {
                  type tunnel-ref;
                  description
                    "Apply action on the specific TE tunnel.";
                }
              }
            }
          }
          container action-info {
            description
              "TE tunnel action information.";
            leaf action {
              type identityref {
                base te-types:tunnel-action-type;
              }
              description
                "The action type.";
            }
            leaf disruptive {
              when "derived-from-or-self(../action, "
```

```
                 + "'te-types:tunnel-action-reoptimize')";
              type empty;
              description
                "When present, specifies whether or not the
                 reoptimization
                 action is allowed to be disruptive.";
            }
          }
        }
        output {
          leaf action-result {
            type identityref {
              base te-types:te-action-result;
            }
            description
              "The result of the tunnel action operation.";
          }
        }
      }
    }
  }
  <CODE ENDS>
```

                  Figure 7: TE Tunnel data model YANG module

6. TE Device YANG Model

   The device TE YANG module ('ietf-te-device') models data that is
   specific to managing a TE device. This module augments the generic
   TE YANG module.

6.1. Module Structure

6.1.1. TE Interfaces

   This branch of the model manages TE interfaces that are present on a
   device. Examples of TE interface properties are:

   * Maximum reservable bandwidth, bandwidth constraints (BC)

   * Flooding parameters

      - Flooding intervals and threshold values

   * Interface attributes

      - (Extended) administrative groups

      - SRLG values

       - TE metric value

   * Fast reroute backup tunnel properties (such as static, auto-
      tunnel)

   The derived state associated with interfaces is grouped under the
   interface "state" sub-container as shown in Figure 8. This covers
   state data such as:

   * Bandwidth information: maximum bandwidth, available bandwidth at
      different priorities and for each class-type (CT)

   * List of admitted LSPs

      - Name, bandwidth value and pool, time, priority

   * Statistics: state counters, flooding counters, admission counters
      (accepted/rejected), preemption counters

   * Adjacency information

      - Neighbor address

      - Metric value

```
   module: ietf-te-device
     augment /te:te:
        +--rw interfaces
              .
           +-- rw te-dev:te-attributes
                 <<intended configuration>>
              .
              +-- ro state
                 <<derived state associated with the TE interface>>
```

              Figure 8: TE interface state YANG subtree

6.2. Tree Diagram

   Figure 9 shows the tree diagram of the device TE YANG model defined
   in modules 'ietf-te-device.yang'.

```
   module: ietf-te-device

     augment /te:te:
       +--rw interfaces
          +--rw threshold-type?          enumeration
          +--rw delta-percentage?        rt-types:percentage
```

```
           +--rw threshold-specification?    enumeration
           +--rw up-thresholds*             rt-types:percentage
           +--rw down-thresholds*           rt-types:percentage
           +--rw up-down-thresholds*        rt-types:percentage
           +--rw interface* [name]
              +--rw name                              if:interface-ref
              +--rw te-metric?
              |      te-types:te-metric
              +--rw (admin-group-type)?
              |  +--:(value-admin-groups)
              |  |  +--rw (value-admin-group-type)?
              |  |     +--:(admin-groups)
              |  |     |  +--rw admin-group?
              |  |     |        te-types:admin-group
              |  |     +--:(extended-admin-groups)
              |  |              {te-types:extended-admin-groups}?
              |  |        +--rw extended-admin-group?
              |  |              te-types:extended-admin-group
              |  +--:(named-admin-groups)
              |     +--rw named-admin-groups* [named-admin-group]
              |             {te-types:extended-admin-groups,
              |              te-types:named-extended-admin-groups}?
              |        +--rw named-admin-group    leafref
              +--rw (srlg-type)?
              |  +--:(value-srlgs)
              |  |  +--rw values* [value]
              |  |     +--rw value    uint32
              |  +--:(named-srlgs)
              |     +--rw named-srlgs* [named-srlg]
              |             {te-types:named-srlg-groups}?
              |        +--rw named-srlg    leafref
              +--rw threshold-type?                  enumeration
              +--rw delta-percentage?
              |      rt-types:percentage
              +--rw threshold-specification?         enumeration
              +--rw up-thresholds*
              |      rt-types:percentage
              +--rw down-thresholds*
              |      rt-types:percentage
              +--rw up-down-thresholds*
              |      rt-types:percentage
              +--rw switching-capabilities* [switching-capability]
              |  +--rw switching-capability    identityref
              |  +--rw encoding?               identityref
              +--ro te-advertisements-state
                 +--ro flood-interval?        uint32
                 +--ro last-flooded-time?     uint32
                 +--ro next-flooded-time?     uint32
```

```
                  +--ro last-flooded-trigger?      enumeration
                  +--ro advertised-level-areas* [level-area]
                     +--ro level-area     uint32
        augment /te:te/te:globals:
          +--rw lsp-install-interval?       uint32
          +--rw lsp-cleanup-interval?       uint32
          +--rw lsp-invalidation-interval?  uint32
        augment /te:te/te:tunnels/te:tunnel:
          +--rw path-invalidation-action?   identityref
          +--rw lsp-install-interval?       uint32
          +--rw lsp-cleanup-interval?       uint32
          +--rw lsp-invalidation-interval?  uint32
        augment /te:te/te:lsps/te:lsp:
          +--ro lsp-timers
          |  +--ro uptime?           uint32
          |  +--ro time-to-install?  uint32
          |  +--ro time-to-destroy?  uint32
          +--ro downstream-info
          |  +--ro nhop?                  te-types:te-tp-id
          |  +--ro outgoing-interface?    if:interface-ref
          |  +--ro neighbor
          |  |  +--ro id?     te-gen-node-id
          |  |  +--ro type?   enumeration
          |  +--ro label?                 rt-types:generalized-label
          +--ro upstream-info
             +--ro phop?        te-types:te-tp-id
             +--ro neighbor
             |  +--ro id?     te-gen-node-id
             |  +--ro type?   enumeration
             +--ro label?       rt-types:generalized-label

     rpcs:
       +---x link-state-update
          +---w input
             +---w (filter-type)
                +--:(match-all)
                |  +---w all          empty
                +--:(match-one-interface)
                   +---w interface?   if:interface-ref
```

            Figure 9: TE Tunnel device model YANG tree diagram

6.3. YANG Module

   The device TE YANG module 'ietf-te-device' imports the following
   module(s):

   * ietf-interfaces defined in [RFC8343]

   * ietf-routing-types defined in [RFC8294]

   * ietf-te-types defined in [I-D.draft-ietf-teas-rfc8776-update]

   * ietf-te defined in this document

   <CODE BEGINS> file "ietf-te-device@2024-02-02.yang"
   module ietf-te-device {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-te-device";

     /* Replace with IANA when assigned */

     prefix te-dev;

     /* Import TE module */

     import ietf-te {
       prefix te;
       reference
         "RFCXXXX: A YANG Data Model for Traffic Engineering
          Tunnels and Interfaces";
     }

     /* Import TE types */

     import ietf-te-types {
       prefix te-types;
       reference
         "draft-ietf-teas-rfc8776-update: Common YANG Data Types
              for Traffic Engineering.";
     }
     import ietf-interfaces {
       prefix if;
       reference
         "RFC8343: A YANG Data Model for Interface Management";
     }
     import ietf-routing-types {
       prefix rt-types;
       reference
         "RFC8294: Common YANG Data Types for the Routing Area";
     }

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

         WG List: <mailto:teas@ietf.org>

         Editor: Tarek Saad
                    <mailto:tsaad.net@gmail.com>

         Editor: Rakesh Gandhi
                    <mailto:rgandhi@cisco.com>

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

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

         Editor: Xufeng Liu
                    <mailto: xufeng.liu.ietf@gmail.com>

         Editor: Igor Bryskin
                    <mailto:i_bryskin@yahoo.com>

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

     description
       "This module defines a data model for TE device configurations,
        state, and RPCs. The model fully conforms to the
        Network Management Datastore Architecture (NMDA).

     // RFC Ed.: replace XXXX with actual RFC number and remove this
     // note.
     // RFC Ed.: update the date below with the date of RFC publication
     // and remove this note.

     revision 2024-02-02 {

```
        description
          "Initial revision for the TE device YANG module.";
        reference
          "RFCXXXX: A YANG Data Model for Traffic Engineering Tunnels
           and Interfaces";
      }

      grouping lsp-device-timers {
        description
          "Device TE LSP timers configs.";
        leaf lsp-install-interval {
          type uint32;
          units "seconds";
          description
            "TE LSP installation delay time.";
        }
        leaf lsp-cleanup-interval {
          type uint32;
          units "seconds";
          description
            "TE LSP cleanup delay time.";
        }
        leaf lsp-invalidation-interval {
          type uint32;
          units "seconds";
          description
            "TE LSP path invalidation before taking action delay time.";
        }
      }

      grouping te-igp-flooding-bandwidth-config {
        description
          "Configurable items for igp flooding bandwidth
           threshold configuration.";
        leaf threshold-type {
          type enumeration {
            enum delta {
              description
                "'delta' indicates that the local
                 system should flood IGP updates when a
                 change in reserved bandwidth >= the specified
                 delta occurs on the interface.";
            }
            enum threshold-crossed {
              description
                "THRESHOLD-CROSSED indicates that
                 the local system should trigger an update (and
                 hence flood) the reserved bandwidth when the
```

```
                  reserved bandwidth changes such that it crosses,
                  or becomes equal to one of the threshold values.";
            }
          }
          description
            "The type of threshold that should be used to specify the
             values at which bandwidth is flooded. 'delta' indicates that
             the local system should flood IGP updates when a change in
             reserved bandwidth >= the specified delta occurs on the
             interface. Where 'threshold-crossed' is specified, the local
             system should trigger an update (and hence flood) the
             reserved bandwidth when the reserved bandwidth changes such
             that it crosses, or becomes equal to one of the threshold
             values.";
        }
        leaf delta-percentage {
          when "../threshold-type = 'delta'" {
            description
              "The percentage delta can only be specified when the
               threshold type is specified to be a percentage delta of
               the reserved bandwidth.";
          }
          type rt-types:percentage;
          description
            "The percentage of the maximum-reservable-bandwidth
             considered as the delta that results in an IGP update
             being flooded.";
        }
        leaf threshold-specification {
          when "../threshold-type = 'threshold-crossed'" {
            description
              "The selection of whether mirrored or separate threshold
               values are to be used requires user specified thresholds
               to be set.";
          }
          type enumeration {
            enum mirrored-up-down {
              description
                "mirrored-up-down indicates that a single set of
                 threshold values should be used for both increasing
                 and decreasing bandwidth when determining whether
                 to trigger updated bandwidth values to be flooded
                 in the IGP TE extensions.";
            }
            enum separate-up-down {
              description
                "separate-up-down indicates that a separate
                 threshold values should be used for the increasing
```

```
                 and decreasing bandwidth when determining whether
                 to trigger updated bandwidth values to be flooded
                 in the IGP TE extensions.";
           }
         }
         description
           "This value specifies whether a single set of threshold
            values should be used for both increasing and decreasing
            bandwidth when determining whether to trigger updated
            bandwidth values to be flooded in the IGP TE extensions.
            'mirrored-up-down' indicates that a single value (or set of
            values) should be used for both increasing and decreasing
            values, where 'separate-up-down' specifies that the
            increasing and decreasing values will be separately
            specified.";
       }
       leaf-list up-thresholds {
         when "../threshold-type = 'threshold-crossed'"
             + "and ../threshold-specification = 'separate-up-down'" {
           description
             "A list of up-thresholds can only be specified when the
              bandwidth update is triggered based on crossing a
              threshold and separate up and down thresholds are
              required.";
         }
         type rt-types:percentage;
         description
           "The thresholds (expressed as a percentage of the maximum
            reservable bandwidth) at which bandwidth updates are to be
            triggered when the bandwidth is increasing.";
       }
       leaf-list down-thresholds {
         when "../threshold-type = 'threshold-crossed'"
             + "and ../threshold-specification = 'separate-up-down'" {
           description
             "A list of down-thresholds can only be specified when the
              bandwidth update is triggered based on crossing a
              threshold and separate up and down thresholds are
              required.";
         }
         type rt-types:percentage;
         description
           "The thresholds (expressed as a percentage of the maximum
            reservable bandwidth) at which bandwidth updates are to be
            triggered when the bandwidth is decreasing.";
       }
       leaf-list up-down-thresholds {
         when "../threshold-type = 'threshold-crossed'"
```

```
            + "and ../threshold-specification = 'mirrored-up-down'" {
          description
            "A list of thresholds corresponding to both increasing
             and decreasing bandwidths can be specified only when an
             update is triggered based on crossing a threshold, and
             the same up and down thresholds are required.";
        }
        type rt-types:percentage;
        description
          "The thresholds (expressed as a percentage of the maximum
           reservable bandwidth of the interface) at which bandwidth
           updates are flooded - used both when the bandwidth is
           increasing and decreasing.";
      }
    }

    /**
     * TE device augmentations
     */
    augment "/te:te" {
      description
        "TE global container.";
      /* TE Interface Configuration Data */
      container interfaces {
        description
          "Configuration data model for TE interfaces.";
        uses te-igp-flooding-bandwidth-config;
        list interface {
          key "name";
          description
            "The list of interfaces enabled for TE.";
          leaf name {
            type if:interface-ref;
            description
              "The reference to interface enabled for TE.";
          }
          /* TE interface parameters */
          leaf te-metric {
            type te-types:te-metric;
            description
              "TE interface metric.";
          }
          choice admin-group-type {
            description
              "TE interface administrative groups
               representation type.";
            case value-admin-groups {
              choice value-admin-group-type {
```

```
                    description
                      "choice of admin-groups.";
                    case admin-groups {
                      description
                        "Administrative group/Resource
                         class/Color.";
                      leaf admin-group {
                        type te-types:admin-group;
                        description
                          "TE interface administrative group.";
                      }
                    }
                    case extended-admin-groups {
                      if-feature "te-types:extended-admin-groups";
                      description
                        "Extended administrative group/Resource
                         class/Color.";
                      leaf extended-admin-group {
                        type te-types:extended-admin-group;
                        description
                          "TE interface extended administrative group.";
                      }
                    }
                  }
                }
                case named-admin-groups {
                  list named-admin-groups {
                    if-feature "te-types:extended-admin-groups";
                    if-feature "te-types:named-extended-admin-groups";
                    key "named-admin-group";
                    description
                      "A list of named admin-group entries.";
                    leaf named-admin-group {
                      type leafref {
                        path "../../../../te:globals/"
                            + "te:named-admin-groups/te:named-admin-group/"
                            + "te:name";
                      }
                      description
                        "A named admin-group entry.";
                    }
                  }
                }
              }
              choice srlg-type {
                description
                  "Choice of SRLG configuration.";
                case value-srlgs {
```

```
            list values {
              key "value";
              description
                "List of SRLG values that
                 this link is part of.";
              leaf value {
                type uint32 {
                  range "0..4294967295";
                }
                description
                  "Value of the SRLG";
              }
            }
          }
          case named-srlgs {
            list named-srlgs {
              if-feature "te-types:named-srlg-groups";
              key "named-srlg";
              description
                "A list of named SRLG entries.";
              leaf named-srlg {
                type leafref {
                  path "../../../../te:globals/"
                     + "te:named-srlgs/te:named-srlg/te:name";
                }
                description
                  "A named SRLG entry.";
              }
            }
          }
        }
        uses te-igp-flooding-bandwidth-config;
        list switching-capabilities {
          key "switching-capability";
          description
            "List of interface capabilities for this interface.";
          leaf switching-capability {
            type identityref {
              base te-types:switching-capabilities;
            }
            description
              "Switching Capability for this interface.";
          }
          leaf encoding {
            type identityref {
              base te-types:lsp-encoding-types;
            }
            description
```

```
                    "Encoding supported by this interface.";
              }
            }
            container te-advertisements-state {
              config false;
              description
                "TE interface advertisements state container.";
              leaf flood-interval {
                type uint32;
                description
                  "The periodic flooding interval.";
              }
              leaf last-flooded-time {
                type uint32;
                units "seconds";
                description
                  "Time elapsed since last flooding in seconds.";
              }
              leaf next-flooded-time {
                type uint32;
                units "seconds";
                description
                  "Time remained for next flooding in seconds.";
              }
              leaf last-flooded-trigger {
                type enumeration {
                  enum link-up {
                    description
                      "Link-up flooding trigger.";
                  }
                  enum link-down {
                    description
                      "Link-down flooding trigger.";
                  }
                  enum threshold-up {
                    description
                      "Bandwidth reservation up threshold.";
                  }
                  enum threshold-down {
                    description
                      "Bandwidth reservation down threshold.";
                  }
                  enum bandwidth-change {
                    description
                      "Bandwidth capacity change.";
                  }
                  enum user-initiated {
                    description
```

```
                      "Initiated by user.";
                  }
                  enum srlg-change {
                    description
                      "SRLG property change.";
                  }
                  enum periodic-timer {
                    description
                      "Periodic timer expired.";
                  }
                }
                default "periodic-timer";
                description
                  "Trigger for the last flood.";
              }
              list advertised-level-areas {
                key "level-area";
                description
                  "List of level-areas that the TE interface is
                   advertised in.";
                leaf level-area {
                  type uint32;
                  description
                    "The IGP area or level where the TE interface link
                     state is advertised in.";
                }
              }
            }
          }
        }
      }

      /* TE globals device augmentation */

      augment "/te:te/te:globals" {
        description
          "Global TE device specific configuration parameters.";
        uses lsp-device-timers;
      }

      /* TE tunnels device configuration augmentation */

      augment "/te:te/te:tunnels/te:tunnel" {
        description
          "Tunnel device dependent augmentation.";
        leaf path-invalidation-action {
          type identityref {
            base te-types:path-invalidation-action-type;
```

```
          }
          description
            "Tunnel path invalidation action.";
        }
        uses lsp-device-timers;
      }

      /* TE LSPs device state augmentation */

      augment "/te:te/te:lsps/te:lsp" {
        description
          "TE LSP device dependent augmentation.";
        container lsp-timers {
          when "../te:origin-type = 'ingress'" {
            description
              "Applicable to ingress LSPs only.";
          }
          description
            "Ingress LSP timers.";
          leaf uptime {
            type uint32;
            units "seconds";
            description
              "The LSP uptime.";
          }
          leaf time-to-install {
            type uint32;
            units "seconds";
            description
              "The time remaining for a new LSP to be instantiated
               in forwarding to carry traffic.";
          }
          leaf time-to-destroy {
            type uint32;
            units "seconds";
            description
              "The time remaining for a existing LSP to be deleted
               from forwarding.";
          }
        }
        container downstream-info {
          when "../te:origin-type != 'egress'" {
            description
              "Downstream information of the LSP.";
          }
          description
            "downstream information.";
          leaf nhop {
```

```
            type te-types:te-tp-id;
            description
              "downstream next-hop address.";
          }
          leaf outgoing-interface {
            type if:interface-ref;
            description
              "downstream interface.";
          }
          container neighbor {
            uses te-types:te-generic-node-id;
            description
              "downstream neighbor address.";
          }
          leaf label {
            type rt-types:generalized-label;
            description
              "downstream label.";
          }
        }
        container upstream-info {
          when "../te:origin-type != 'ingress'" {
            description
              "Upstream information of the LSP.";
          }
          description
            "upstream information.";
          leaf phop {
            type te-types:te-tp-id;
            description
              "upstream next-hop or previous-hop address.";
          }
          container neighbor {
            uses te-types:te-generic-node-id;
            description
              "upstream neighbor address.";
          }
          leaf label {
            type rt-types:generalized-label;
            description
              "upstream label.";
          }
        }
      }

      /* TE interfaces RPCs/execution Data */

      rpc link-state-update {
```

```
      description
        "Triggers a link state update for the specific interface.";
      input {
        choice filter-type {
          mandatory true;
          description
            "Filter choice.";
          case match-all {
            leaf all {
              type empty;
              mandatory true;
              description
                "Match all TE interfaces.";
            }
          }
          case match-one-interface {
            leaf interface {
              type if:interface-ref;
              description
                "Match a specific TE interface.";
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

                 Figure 10: TE device data model YANG module

7. Notifications

   Notifications are a key component of any topology data model.

   [RFC8639] and [RFC8641] define a subscription mechanism and a push
   mechanism for YANG datastores. These mechanisms currently allow the
   user to:

   * Subscribe to notifications on a per-client basis.

   * Specify subtree filters or XML Path Language (XPath) filters so
      that only contents of interest will be sent.

   * Specify either periodic or on-demand notifications.

8. IANA Considerations

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

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

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

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

      Name: ietf-te
      Namespace: urn:ietf:params:xml:ns:yang:ietf-te
      Prefix: te
      Reference: RFCXXXX

      Name: ietf-te-device
      Namespace: urn:ietf:params:xml:ns:yang:ietf-te-device
      Prefix: te-device
      Reference: RFCXXXX

9. Security Considerations

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

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

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

"/te/globals": This module specifies the global TE configurations on a device. Unauthorized access to this container could cause the device to ignore packets it should receive and process.

"/te/tunnels": This list specifies the configuration and state of TE Tunnels present on the device or controller. Unauthorized access to this list could cause the device to ignore packets it should receive and process. An attacker may also use state to derive information about the network topology, and subsequently orchestrate further attacks.

"/te/interfaces": This list specifies the configuration and state TE interfaces on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

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

"/te/lsps": this list contains information state about established LSPs in the network. An attacker can use this information to derive information about the network topology, and subsequently orchestrate further attacks.

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

"/te/tunnels-actions": using this RPC, an attacker can modify existing paths that may be carrying live traffic, and hence result to interruption to services carried over the network.

"/te/tunnels-path-compute": using this RPC, an attacker can retrieve secured information about the network provider which can be used to orchestrate further attacks.

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

10. Acknowledgement

The authors would like to thank the members of the multi-vendor YANG
design team who are involved in the definition of this model.

The authors would like to thank Tom Petch and Adrian Farrel for
reviewing and providing useful feedback about the document. The
authors would also like to thank Loa Andersson, Lou Berger, Sergio
Belotti, Italo Busi, Carlo Perocchio, Francesco Lazzeri, Aihua Guo,
Dhruv Dhody, and Raqib Jones for providing feedback on this document.

11. Contributors

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

Himanshu Shah
Ciena

Email: hshah@ciena.com

Xia Chen
Huawei Technologies

Email: jescia.chenxia@huawei.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com

12. Appendix A: Data Tree Examples

This section contains examples of use of the model with RESTCONF
[RFC8040] and JSON encoding.

For the example we will use a 4 node MPLS network were RSVP-TE MPLS
Tunnels can be setup. The loopbacks of each router are shown. The
network in Figure 11 will be used in the examples described in the
following sections.

```
   192.0.2.1          192.0.2.2          192.0.2.4
    +-------+          +-------+          +-------+
    |       |          |       |          |       |
    |   A   +--------+ |   B   +------+ |  D    |
    +---+---+          +-------+          +---+---+
        |                                     |
        |              +-------+              |
        |              |       |              |
        +-----------+  |   C   +----------+   |
                       |       |
                       +-------+
                       192.0.2.3
```
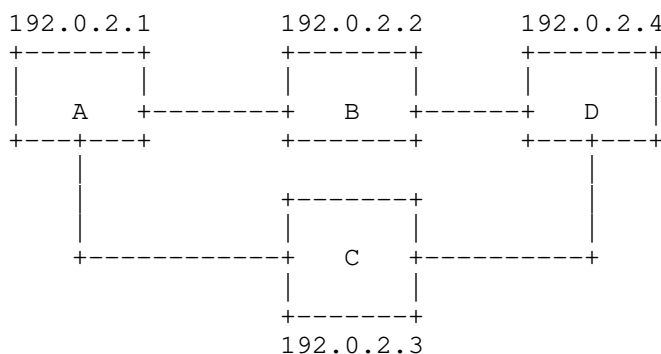
                Figure 11: TE network used in data tree examples

12.1. Basic Tunnel Setup

   This example uses the TE Tunnel YANG data model defined in this
   document to create an RSVP-TE signaled Tunnel of packet LSP encoding
   type. First, the TE Tunnel is created with no specific restrictions
   or constraints (e.g., protection or restoration). The TE Tunnel
   ingresses on router A and egresses on router D.

   In this case, the TE Tunnel is created without specifying additional
   information about the primary paths.

   POST /restconf/data/ietf-te:te/tunnels HTTP/1.1
      Host: example.com
      Accept: application/yang-data+json
      Content-Type: application/yang-data+json

   {
     "ietf-te:tunnel": [
       {
         "name": "Example_LSP_Tunnel_A_2",
         "encoding": "te-types:lsp-encoding-packet",
         "admin-state": "te-types:tunnel-state-up",
         "source": "192.0.2.1",
         "destination": "192.0.2.4",
         "bidirectional": "false",
         "signaling-type": "te-types:path-setup-rsvp"
       }
     ]
   }

12.2. Global Named Path Constraints

   This example uses the YANG data model to create a 'named path
   constraint' that can be reference by TE Tunnels. The path
   constraint, in this case, limits the TE Tunnel hops for the computed
   path.

   POST /restconf/data/ietf-te:te/globals/named-path-constraints
       HTTP/1.1
      Host: example.com
      Accept: application/yang-data+json
      Content-Type: application/yang-data+json

    "ietf-te:named-path-constraint": {
            "name": "max-hop-3",
            "path-metric-bounds": {
              "path-metric-bound": {
                 "metric-type": "te-types:path-metric-hop",
                 "upper-bound": "3"
        }
       }
      }
     }

12.3. Tunnel with Global Path Constraint

   In this example, the previously created 'named path constraint' is
   applied to the TE Tunnel created in Section 12.1.

```
POST /restconf/data/ietf-te:te/tunnels HTTP/1.1
    Host: example.com
    Accept: application/yang-data+json
    Content-Type: application/yang-data+json

{
  "ietf-te:ietf-tunnel": [
    {
      "name": "Example_LSP_Tunnel_A_4_1",
      "encoding": "te-types:lsp-encoding-packet",
      "description": "Simple_LSP_with_named_path",
      "admin-state": "te-types:tunnel-state-up",
      "source": "192.0.2.1",
      "destination": "192.0.2.4",
      "signaling-type": "path-setup-rsvp",
      "primary-paths": [
        {
          "primary-path": {
            "name": "Simple_LSP_1",
            "use-path-computation": "true",
            "named-path-constraint": "max-hop-3"
          }
        }
      ]
    }
  ]
}
```

12.4. Tunnel with Per-tunnel Path Constraint

In this example, the a per tunnel path constraint is explicitly
indicated under the TE Tunnel created in Section 12.1 to constrain
the computed path for the tunnel.

```
POST /restconf/data/ietf-te:te/tunnels HTTP/1.1
    Host: example.com
    Accept: application/yang-data+json
    Content-Type: application/yang-data+json

{
  "ietf-te:tunnel": [
    {
      "name": "Example_LSP_Tunnel_A_4_2",
      "encoding": "te-types:lsp-encoding-packet",
      "admin-state": "te-types:tunnel-state-up",
      "source": "192.0.2.1",
      "destination": "192.0.2.4",
      "signaling-type": "te-types:path-setup-rsvp",
      "primary-paths": {
        "primary-path": [
          {
            "name": "path1",
            "path-metric-bounds": {
              "path-metric-bound": [
                {
                  "metric-type": "te-types:path-metric-hop",
                  "upper-bound": "3"
                }
              ]
            }
          }
        ]
      }
    }
  ]
}
```

12.5. Tunnel State

In this example, the 'GET' query is sent to return the state stored
about the tunnel.

```
GET  /restconf/data/ietf-te:te/tunnels +
     /tunnel="Example_LSP_Tunnel_A_4_1"
     /primary-paths/ HTTP/1.1
    Host: example.com
    Accept: application/yang-data+json
```

The request, with status code 200 would include, for example, the
following json:

```
  {
    "ietf-te:primary-paths": {
      "primary-path": [
        {
          "name": "path1",
          "path-computation-method": "te-types:path-locally-computed",
          "computed-paths-properties": {
            "computed-path-properties": [
              {
                "k-index": "1",
                "path-properties": {
                  "path-route-objects": {
                    "path-route-object": [
                      {
                        "index": "1",
                        "numbered-node-hop": {
                          "node-id": "192.0.2.2"
                        }
                      },
                      {
                        "index": "2",
                        "numbered-node-hop": {
                          "node-id": "192.0.2.4"
                        }
                      }
                    ]
                  }
                }
              }
            ]
          },
          "lsps": {
            "lsp": [
              {
                "tunnel-name": "Example_LSP_Tunnel_A_4_1",
                "node": "192.0.2.1 ",
                "lsp-id": "25356"
              }
            ]
          }
        }
      ]
    }
  }
```

12.6. Example TE Tunnel with Primary and Secondary Paths

```
                                +----------+
                              +-|192.0.2.9 |---+      +----------+
                              | +----------+   |      |192.0.2.10|
                              |                |      +----------+
    +----------+    +----------+    +----------+   |        |    |
    |192.0.2.8 |----|192.0.2.3 |-----|192.0.2.4 |----+       |    |
    +----------+    +----------+    +----------+        |        |
         |               |               |        |
    +----------+          |               |        +----------+
    |192.0.2.1 |---------+               |        |192.0.2.5 |
    +----------+                         +----------|          |
        | |                                         +----------+
        | |          +----------+                        | |
        | +----------|192.0.2.2 |--------------------+    |
        |            +----------+                    |    |
        |               | |                          |    |
    +----------+        | |  +-------------------+----------+
    |192.0.2.6 |---------------+  +----------------|192.0.2.7 |
    +----------+                                   +----------+
```
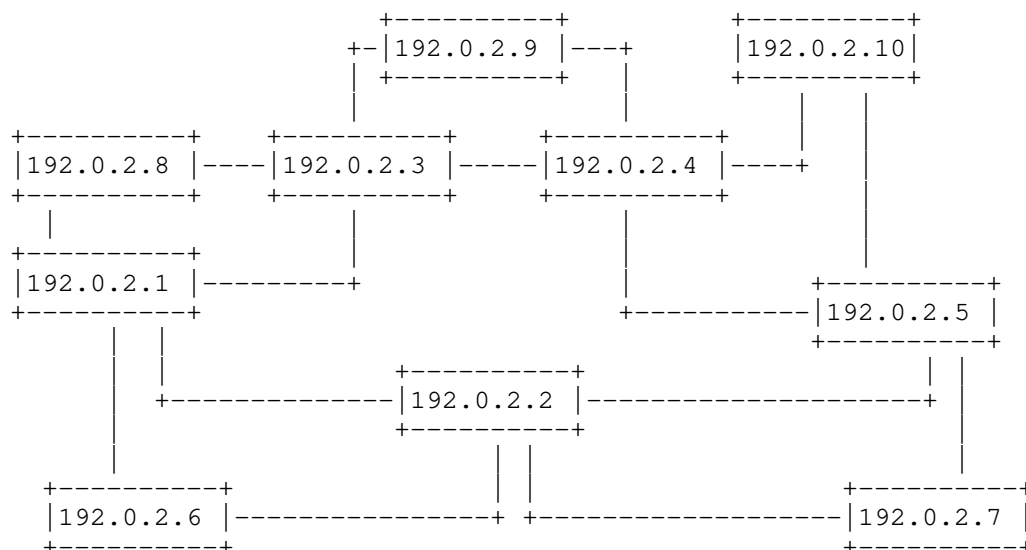
              Figure 12: TE network used in data tree examples

    Below is the state retrieved for a TE tunnel from source 192.0.2.1 to
    192.0.2.5 with primary, secondary, reverse, and secondary reverse
    paths as shown in Figure 12.

```
{
  "ietf-te:te": {
    "tunnels": {
      "tunnel": [
        {
          "name": "example-1",
          "description": "Example in slide 1",
          "source": "192.0.2.1",
          "destination": "192.0.2.5",
          "bidirectional": false,
          "primary-paths": {
            "primary-path": [
              {
                "name": "primary-1 (fwd)",
                "explicit-route-objects": {
                  "route-object-include-exclude": [
                    {
                      "index": 1,
                      "explicit-route-usage" : "route-include-object",
                      "numbered-node-hop": {
                        "node-id": "192.0.2.2",
                        "hop-type": "loose"
```

```
                    }
                  }
                ]
              },
              "primary-reverse-path": {
                "name": "primary-2 (rev)",
                "explicit-route-objects": {
                  "route-object-include-exclude": [
                    {
                      "index": 1,
                      "explicit-route-usage" : "route-include-object",
                      "numbered-node-hop": {
                        "node-id": "192.0.2.3",
                        "hop-type": "loose"
                      }
                    }
                  ]
                },
                "candidate-secondary-reverse-paths": {
                  "candidate-secondary-reverse-path": [
                    "secondary-3 (rev)",
                    "secondary-4 (rev)",
                    "secondary-5 (rev)"
                  ]
                }
              },
              "candidate-secondary-paths": {
                "candidate-secondary-path": [
                  "secondary-1 (fwd)",
                  "secondary-2 (fwd)"
                ]
              }
            }
          ]
        },
        "secondary-paths": {
          "secondary-path": [
            {
              "name": "secondary-1 (fwd)",
              "explicit-route-objects": {
                "route-object-include-exclude": [
                  {
                    "index": 1,
                    "explicit-route-usage" : "route-include-object",
                    "numbered-node-hop": {
                      "node-id": "192.0.2.1"
                    }
                  },
```

```
                    {
                      "index": 2,
                      "numbered-node-hop": {
                        "node-id": "192.0.2.2",
                        "hop-type": "loose"
                      }
                    }
                  ]
                }
              },
              {
                "name": "secondary-2 (fwd)",
                "explicit-route-objects": {
                  "route-object-include-exclude": [
                    {
                      "index": 1,
                      "explicit-route-usage" : "route-include-object",
                      "numbered-node-hop": {
                        "node-id": "192.0.2.2"
                      }
                    },
                    {
                      "index": 2,
                      "numbered-node-hop": {
                        "node-id": "192.0.2.5",
                        "hop-type": "loose"
                      }
                    }
                  ]
                }
              }
            ]
          },
          "secondary-reverse-paths": {
            "secondary-reverse-path": [
              {
                "name": "secondary-3 (rev)",
                "explicit-route-objects": {
                  "route-object-include-exclude": [
                    {
                      "index": 1,
                      "explicit-route-usage" : "route-include-object",
                      "numbered-node-hop": {
                        "node-id": "192.0.2.5"
                      }
                    },
                    {
                      "index": 2,
```

```
                   "numbered-node-hop": {
                     "node-id": "192.0.2.4",
                     "hop-type": "loose"
                   }
                 }
               ]
             }
           },
           {
             "name": "secondary-4 (rev)",
             "explicit-route-objects": {
               "route-object-include-exclude": [
                 {
                   "index": 1,
                   "explicit-route-usage" : "route-include-object",
                   "numbered-node-hop": {
                     "node-id": "192.0.2.4"
                   }
                 },
                 {
                   "index": 2,
                   "numbered-node-hop": {
                     "node-id": "192.0.2.3",
                     "hop-type": "loose"
                   }
                 }
               ]
             }
           },
           {
             "name": "secondary-5 (rev)",
             "explicit-route-objects": {
               "route-object-include-exclude": [
                 {
                   "index": 1,
                   "explicit-route-usage" : "route-include-object",
                   "numbered-node-hop": {
                     "node-id": "192.0.2.3"
                   }
                 },
                 {
                   "index": 2,
                   "numbered-node-hop": {
                     "node-id": "192.0.2.1",
                     "hop-type":"loose"
                   }
                 }
               ]
```

```
              }
            }
          ]
        }
      },
      {
        "name": "example-3",
        "description": "Example in slide 3",
        "source": "192.0.2.1",
        "destination": "192.0.2.5",
        "bidirectional": true,
        "primary-paths": {
          "primary-path": [
            {
              "name": "primary-1 (bidir)",
              "explicit-route-objects": {
                "route-object-include-exclude": [
                  {
                    "index": 1,
                    "explicit-route-usage" : "route-include-object",
                    "numbered-node-hop": {
                      "node-id": "192.0.2.2",
                      "hop-type": "loose"
                    }
                  }
                ]
              },
              "candidate-secondary-paths": {
                "candidate-secondary-path": [
                  "secondary-1 (bidir)",
                  "secondary-2 (bidir)"
                ]
              }
            }
          ]
        },
        "secondary-paths": {
          "secondary-path": [
            {
              "name": "secondary-1 (bidir)",
              "explicit-route-objects": {
                "route-object-include-exclude": [
                  {
                    "index": 1,
                    "explicit-route-usage" : "route-include-object",
                    "numbered-node-hop": {
                      "node-id": "192.0.2.1"
                    }
```

```
                      },
                      {
                        "index": 2,
                        "numbered-node-hop": {
                          "node-id": "192.0.2.2",
                          "hop-type": "loose"
                        }
                      }
                    ]
                  }
                },
                {
                  "name": "secondary-2 (bidir)",
                  "explicit-route-objects": {
                    "route-object-include-exclude": [
                      {
                        "index": 1,
                        "explicit-route-usage" : "route-include-object",
                        "numbered-node-hop": {
                          "node-id": "192.0.2.2"
                        }
                      },
                      {
                        "index": 2,
                        "numbered-node-hop": {
                          "node-id": "192.0.2.5",
                          "hop-type": "loose"
                        }
                      }
                    ]
                  }
                }
              ]
            }
          },
          {
            "name": "example-4",
            "description": "Example in slide 4",
            "source": "192.0.2.1",
            "destination": "192.0.2.5",
            "bidirectional": false,
            "primary-paths": {
              "primary-path": [
                {
                  "name": "primary-1 (fwd)",
                  "co-routed": [null],
                  "explicit-route-objects": {
                    "route-object-include-exclude": [
```

```
                    {
                      "index": 1,
                      "explicit-route-usage" : "route-include-object",
                      "numbered-node-hop": {
                        "node-id": "192.0.2.2",
                        "hop-type": "loose"
                      }
                    }
                  ]
                },
                "primary-reverse-path": {
                  "name": "primary-2 (rev)",
                  "candidate-secondary-reverse-paths": {
                    "candidate-secondary-reverse-path": [
                      "secondary-3 (rev)",
                      "secondary-4 (rev)"
                    ]
                  }
                },
                "candidate-secondary-paths": {
                  "candidate-secondary-path": [
                    "secondary-1 (fwd)",
                    "secondary-2 (fwd)"
                  ]
                }
              }
            ]
          },
          "secondary-paths": {
            "secondary-path": [
              {
                "name": "secondary-1 (fwd)",
                "co-routed": [null],
                "explicit-route-objects": {
                  "route-object-include-exclude": [
                    {
                      "index": 1,
                      "explicit-route-usage" : "route-include-object",
                      "numbered-node-hop": {
                        "node-id": "192.0.2.1"
                      }
                    },
                    {
                      "index": 2,
                      "numbered-node-hop": {
                        "node-id": "192.0.2.2",
                        "hop-type": "loose"
                      }
```

```
                    }
                  ]
                }
              },
              {
                "name": "secondary-2 (fwd)",
                "co-routed": [null],
                "explicit-route-objects": {
                  "route-object-include-exclude": [
                    {
                      "index": 1,
                      "explicit-route-usage" : "route-include-object",
                      "numbered-node-hop": {
                        "node-id": "192.0.2.2"
                      }
                    },
                    {
                      "index": 2,
                      "numbered-node-hop": {
                        "node-id": "192.0.2.5",
                        "hop-type": "loose"
                      }
                    }
                  ]
                }
              }
            ]
          },
          "secondary-reverse-paths": {
            "secondary-reverse-path": [
              {
                "name": "secondary-3 (rev)"
              },
              {
                "name": "secondary-4 (rev)"
              }
            ]
          }
        }
      }
    ]
  }
}
```

13. Appendix B: Full Model Tree Diagram

   Figure 13 shows the full tree diagram of the TE YANG model defined in
   module 'ietf-te.yang'.

```
module: ietf-te
  +--rw te
     +--rw enable?    boolean
     +--rw globals
     |  +--rw named-admin-groups
     |  |  +--rw named-admin-group* [name]
     |  |          {te-types:extended-admin-groups,
     |  |           te-types:named-extended-admin-groups}?
     |  |     +--rw name            string
     |  |     +--rw bit-position?   uint32
     |  +--rw named-srlgs
     |  |  +--rw named-srlg* [name] {te-types:named-srlg-groups}?
     |  |     +--rw name     string
     |  |     +--rw value?   te-types:srlg
     |  |     +--rw cost?    uint32
     |  +--rw named-path-constraints
     |     +--rw named-path-constraint* [name]
     |             {te-types:named-path-constraints}?
     |        +--rw name                        string
     |        +---u path-constraints-common
     +--rw tunnels
     |  +--rw tunnel* [name]
     |     +--rw name                           string
     |     +--rw alias?                         string
     |     +--rw identifier?                    uint32
     |     +--rw color?                         uint32
     |     +--rw description?                   string
     |     +--rw admin-state?                   identityref
     |     +--ro operational-state?             identityref
     |     +---u te-types:encoding-and-switching-type
     |     +---u tunnel-common-attributes
     |     +--rw controller
     |     |  +--rw protocol-origin?        identityref
     |     |  +--rw controller-entity-id?   string
     |     +--rw reoptimize-timer?              uint16
     |     +---u tunnel-associations-properties
     |     +---u protection-restoration-properties
     |     +---u te-types:tunnel-constraints
     |     +---u tunnel-hierarchy-properties
     |     +--rw primary-paths
     |     |  +--rw primary-path* [name]
     |     |     +--ro active?                 boolean
     |     |     +---u path-common-properties
     |     |     +---u path-forward-properties
     |     |     +---u k-requested-paths
     |     |     +---u path-compute-info
     |     |     +---u path-state
     |     |     +--rw primary-reverse-path
```

```
   │   │            │   +---u path-common-properties
   │   │            │   +---u path-compute-info
   │   │            │   +---u path-state
   │   │            │   +--rw candidate-secondary-reverse-paths
   │   │            │      +--rw candidate-secondary-reverse-path*
   │   │            │              [secondary-reverse-path]
   │   │            │         +--rw secondary-reverse-path   leafref
   │   │            │         +--ro active?                  boolean
   │   │         +--rw candidate-secondary-paths
   │   │            +--rw candidate-secondary-path* [secondary-path]
   │   │               +--rw secondary-path   leafref
   │   │               +--ro active?          boolean
   │      +--rw secondary-paths
   │      │  +--rw secondary-path* [name]
   │      │     +---u path-common-properties
   │      │     +--rw preference?                      uint8
   │      │     +--rw secondary-reverse-path?          leafref
   │      │     +---u path-compute-info
   │      │     +---u protection-restoration-properties
   │      │     +---u path-state
   │      +--rw secondary-reverse-paths
   │      │  +--rw secondary-reverse-path* [name]
   │      │     +---u path-common-properties
   │      │     +--rw preference?                      uint8
   │      │     +---u path-compute-info
   │      │     +---u protection-restoration-properties
   │      │     +---u path-state
   │      +---x tunnel-action
   │      │  +---w input
   │      │  │  +---w action-type?   identityref
   │      │  +--ro output
   │      │     +--ro action-result?   identityref
   │      +---x protection-external-commands
   │         +---w input
   │            +---w protection-external-command?     identityref
   │            +---w protection-group-ingress-node?   boolean
   │            +---w protection-group-egress-node?    boolean
   │            +---w path-name?                       string
   │            +---w path-type?
   │            │       te-types:path-type
   │            +---w traffic-type?                    enumeration
   │            +---w extra-traffic-tunnel-ref?        tunnel-ref
   +--ro lsps
      +--ro lsp* [tunnel-name lsp-id node]
         +--ro tunnel-name                         string
         +--ro lsp-id                              uint16
         +--ro node
         │       te-types:te-node-id
```

```
                +--ro source?
                |       te-types:te-node-id
                +--ro destination?
                |       te-types:te-node-id
                +--ro tunnel-id?                        uint16
                +--ro extended-tunnel-id?               yang:dotted-quad
                +--ro operational-state?                identityref
                +--ro signaling-type?                   identityref
                +--ro origin-type?                      enumeration
                +--ro lsp-resource-status?              enumeration
                +--ro lockout-of-normal?                boolean
                +--ro freeze?                           boolean
                +--ro lsp-protection-role?              enumeration
                +--ro lsp-protection-state?             identityref
                +--ro protection-group-ingress-node-id?
                |       te-types:te-node-id
                +--ro protection-group-egress-node-id?
                |       te-types:te-node-id
                +--ro lsp-actual-route-information
                   +--ro lsp-actual-route-information* [index]
                      +---u te-types:record-route-state

     rpcs:
       +---x tunnels-path-compute
       |  +---w input
       |  |  +---w path-compute-info
       |  +--ro output
       |     +--ro path-compute-result
       +---x tunnels-actions
          +---w input
          |  +---w tunnel-info
          |  |  +---w (filter-type)
          |  |     +--:(all-tunnels)
          |  |     |  +---w all        empty
          |  |     +--:(one-tunnel)
          |  |        +---w tunnel?    tunnel-ref
          |  +---w action-info
          |     +---w action?        identityref
          |     +---w disruptive?    empty
          +--ro output
             +--ro action-result?    identityref

     grouping path-common-properties:
       +-- name?                        string
       +-- path-computation-method?    identityref
       +-- path-computation-server
       |  +---u te-types:te-generic-node-id
       +-- compute-only?               empty
```

```
      +-- use-path-computation?      boolean
      +-- lockdown?                  empty
      +--ro path-scope?              identityref
    grouping path-compute-info:
      +---u tunnel-associations-properties
      +---u te-types:generic-path-optimization
      +-- named-path-constraint?                  leafref
      |       {te-types:named-path-constraints}?
      +---u path-constraints-common
    grouping path-forward-properties:
      +-- preference?   uint8
      +-- co-routed?    boolean
    grouping k-requested-paths:
      +-- k-requested-paths?   uint8
    grouping path-state:
      +---u path-computation-response
      +--ro lsp-provisioning-error-infos
      |  +--ro lsp-provisioning-error-info* []
      |     +--ro error-reason?        identityref
      |     +--ro error-description?   string
      |     +--ro error-timestamp?     yang:date-and-time
      |     +--ro error-node-id?       te-types:te-node-id
      |     +--ro error-link-id?       te-types:te-tp-id
      |     +--ro lsp-id?              uint16
      +--ro lsps
         +--ro lsp* [node lsp-id]
            +--ro tunnel-name?   -> /te/lsps/lsp/tunnel-name
            +--ro node?          leafref
            +--ro lsp-id?        leafref
    grouping path-computation-response:
      +--ro computed-paths-properties
      |  +--ro computed-path-properties* [k-index]
      |     +--ro k-index?                           uint8
      |     +---u te-types:generic-path-properties
      +--ro computed-path-error-infos
         +--ro computed-path-error-info* []
            +--ro error-description?   string
            +--ro error-timestamp?     yang:date-and-time
            +--ro error-reason?        identityref
    grouping protection-restoration-properties:
      +-- protection
      |  +-- protection-type?                identityref
      |  +-- protection-reversion-disable?   boolean
      |  +-- hold-off-time?                  uint32
      |  +-- wait-to-revert?                 uint16
      |  +-- aps-signal-id?                  uint8
      +-- restoration
         +-- restoration-type?                 identityref
```

```
        +-- restoration-scheme?              identityref
        +-- restoration-reversion-disable?   boolean
        +-- hold-off-time?                   uint32
        +-- wait-to-restore?                 uint16
        +-- wait-to-revert?                  uint16
  grouping tunnel-associations-properties:
    +-- association-objects
        +-- association-object* [association-key]
        |  +-- association-key?   string
        |  +-- type?             identityref
        |  +-- id?               uint16
        |  +-- source
        |     +---u te-types:te-generic-node-id
        +-- association-object-extended* [association-key]
           +-- association-key?   string
           +-- type?             identityref
           +-- id?               uint16
           +-- source
           |  +---u te-types:te-generic-node-id
           +-- global-source?    uint32
           +-- extended-id?       yang:hex-string
  grouping tunnel-end-point:
    +-- node-id?        nw:node-id
    +-- te-node-id?     te-types:te-node-id
    +-- tunnel-tp-id?   binary
  grouping tunnel-common-attributes:
    +-- source
    |  +---u tunnel-end-point
    +-- destination
    |  +---u tunnel-end-point
    +-- bidirectional?   boolean
  grouping tunnel-hierarchy-properties:
    +-- hierarchy
        +-- dependency-tunnels
        |  +-- dependency-tunnel* [name]
        |     +-- name?                           tunnel-ref
        |     +---u te-types:encoding-and-switching-type
        +-- hierarchical-link
           +-- enable?                        boolean
           +-- local-node-id?                 nw:node-id
           +-- local-te-node-id?              te-types:te-node-id
           +-- local-link-tp-id?              nt:tp-id
           +-- local-te-link-tp-id?           te-types:te-tp-id
           +-- remote-node-id?                nw:node-id
           +-- remote-link-tp-id?             nt:tp-id
           +-- remote-te-link-tp-id?          te-types:te-tp-id
           +-- remote-te-node-id?             te-types:te-node-id
           +--ro link-id?                       nt:link-id
```

```
             +-- network-id?                        nw:network-id
             +---u te-types:te-topology-identifier
     grouping path-constraints-common:
       +---u te-types:common-path-constraints-attributes
       +---u te-types:generic-path-disjointness
       +---u te-types:path-constraints-route-objects
       +-- path-in-segment!
       |  +---u te-types:label-set-info
       +-- path-out-segment!
          +---u te-types:label-set-info
```

       Figure 13: Full tree diagram of TE Tunnel YANG data model

14. References

14.1. Normative References

   [I-D.draft-ietf-teas-rfc8776-update]
              Busi, I., Guo, A., Liu, X., Saad, T., and I. Bryskin,
              "Common YANG Data Types for Traffic Engineering", Work in
              Progress, Internet-Draft, draft-ietf-teas-rfc8776-update-
              09, 29 January 2024,
              <https://datatracker.ietf.org/doc/html/draft-ietf-teas-
              rfc8776-update-09>.

   [ITU_G.808.1]
              ITU-T Recommendation G.808.1, "Generic protection
              switching - Linear trail and subnetwork protection", ITU-T
              G.808.1 , May 2014.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <https://www.rfc-editor.org/info/rfc2119>.

   [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
              and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP
              Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001,
              <https://www.rfc-editor.org/info/rfc3209>.

   [RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label
              Switching (GMPLS) Signaling Resource ReserVation Protocol-
              Traffic Engineering (RSVP-TE) Extensions", RFC 3473,
              DOI 10.17487/RFC3473, January 2003,
              <https://www.rfc-editor.org/info/rfc3473>.

   [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
              DOI 10.17487/RFC3688, January 2004,
              <https://www.rfc-editor.org/info/rfc3688>.

   [RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP)
              Hierarchy with Generalized Multi-Protocol Label Switching
              (GMPLS) Traffic Engineering (TE)", RFC 4206,
              DOI 10.17487/RFC4206, October 2005,
              <https://www.rfc-editor.org/info/rfc4206>.

   [RFC4427] Mannie, E., Ed. and D. Papadimitriou, Ed., "Recovery
              (Protection and Restoration) Terminology for Generalized
              Multi-Protocol Label Switching (GMPLS)", RFC 4427,
              DOI 10.17487/RFC4427, March 2006,
              <https://www.rfc-editor.org/info/rfc4427>.

   [RFC4872] Lang, J.P., Ed., Rekhter, Y., Ed., and D. Papadimitriou,
              Ed., "RSVP-TE Extensions in Support of End-to-End
              Generalized Multi-Protocol Label Switching (GMPLS)
              Recovery", RFC 4872, DOI 10.17487/RFC4872, May 2007,
              <https://www.rfc-editor.org/info/rfc4872>.

   [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
              the Network Configuration Protocol (NETCONF)", RFC 6020,
              DOI 10.17487/RFC6020, October 2010,
              <https://www.rfc-editor.org/info/rfc6020>.

   [RFC6107] Shiomoto, K., Ed. and A. Farrel, Ed., "Procedures for
              Dynamically Signaled Hierarchical Label Switched Paths",
              RFC 6107, DOI 10.17487/RFC6107, February 2011,
              <https://www.rfc-editor.org/info/rfc6107>.

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

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

   [RFC6780] Berger, L., Le Faucheur, F., and A. Narayanan, "RSVP
              ASSOCIATION Object Extensions", RFC 6780,
              DOI 10.17487/RFC6780, October 2012,
              <https://www.rfc-editor.org/info/rfc6780>.

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

   [RFC7271] Ryoo, J., Ed., Gray, E., Ed., van Helvoort, H.,
              D'Alessandro, A., Cheung, T., and E. Osborne, "MPLS
              Transport Profile (MPLS-TP) Linear Protection to Match the
              Operational Expectations of Synchronous Digital Hierarchy,
              Optical Transport Network, and Ethernet Transport Network
              Operators", RFC 7271, DOI 10.17487/RFC7271, June 2014,
              <https://www.rfc-editor.org/info/rfc7271>.

   [RFC7308] Osborne, E., "Extended Administrative Groups in MPLS
              Traffic Engineering (MPLS-TE)", RFC 7308,
              DOI 10.17487/RFC7308, July 2014,
              <https://www.rfc-editor.org/info/rfc7308>.

   [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S.
              Previdi, "OSPF Traffic Engineering (TE) Metric
              Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015,
              <https://www.rfc-editor.org/info/rfc7471>.

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

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

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

   [RFC8232] Crabbe, E., Minei, I., Medved, J., Varga, R., Zhang, X.,
              and D. Dhody, "Optimizations of Label Switched Path State
              Synchronization Procedures for a Stateful PCE", RFC 8232,
              DOI 10.17487/RFC8232, September 2017,
              <https://www.rfc-editor.org/info/rfc8232>.

   [RFC8234] Ryoo, J., Cheung, T., van Helvoort, H., Busi, I., and G.
              Wen, "Updates to MPLS Transport Profile (MPLS-TP) Linear
              Protection in Automatic Protection Switching (APS) Mode",
              RFC 8234, DOI 10.17487/RFC8234, August 2017,
              <https://www.rfc-editor.org/info/rfc8234>.

   [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger,
              "Common YANG Data Types for the Routing Area", RFC 8294,
              DOI 10.17487/RFC8294, December 2017,
              <https://www.rfc-editor.org/info/rfc8294>.

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

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

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

   [RFC8343] Bjorklund, M., "A YANG Data Model for Interface
              Management", RFC 8343, DOI 10.17487/RFC8343, March 2018,
              <https://www.rfc-editor.org/info/rfc8343>.

   [RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N.,
              Ananthakrishnan, H., and X. Liu, "A YANG Data Model for
              Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March
              2018, <https://www.rfc-editor.org/info/rfc8345>.

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

   [RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward,
              D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE)
              Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March
              2019, <https://www.rfc-editor.org/info/rfc8570>.

   [RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard,
              E., and A. Tripathy, "Subscription to YANG Notifications",
              RFC 8639, DOI 10.17487/RFC8639, September 2019,
              <https://www.rfc-editor.org/info/rfc8639>.

   [RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications
              for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641,
              September 2019, <https://www.rfc-editor.org/info/rfc8641>.

   [RFC8795] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and
              O. Gonzalez de Dios, "YANG Data Model for Traffic
              Engineering (TE) Topologies", RFC 8795,
              DOI 10.17487/RFC8795, August 2020,
              <https://www.rfc-editor.org/info/rfc8795>.

   [RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder,
              "The BGP Tunnel Encapsulation Attribute", RFC 9012,
              DOI 10.17487/RFC9012, April 2021,
              <https://www.rfc-editor.org/info/rfc9012>.

14.2. Informative References

   [rfc3473] Berger, L., Ed., "Generalized Multi-Protocol Label
              Switching (GMPLS) Signaling Resource ReserVation Protocol-
              Traffic Engineering (RSVP-TE) Extensions", RFC 3473,
              DOI 10.17487/RFC3473, January 2003,
              <https://www.rfc-editor.org/info/rfc3473>.

   [RFC3945] Mannie, E., Ed., "Generalized Multi-Protocol Label
              Switching (GMPLS) Architecture", RFC 3945,
              DOI 10.17487/RFC3945, October 2004,
              <https://www.rfc-editor.org/info/rfc3945>.

   [RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov,
              A., and P. Mattes, "Segment Routing Policy Architecture",
              RFC 9256, DOI 10.17487/RFC9256, July 2022,
              <https://www.rfc-editor.org/info/rfc9256>.

Authors' Addresses

   Tarek Saad
   Cisco Systems Inc
   Email: tsaad.net@gmail.com

   Rakesh Gandhi
   Cisco Systems Inc
   Email: rgandhi@cisco.com

   Xufeng Liu
   Alef Edge
   Email: xufeng.liu.ietf@gmail.com

   Vishnu Pavan Beeram
   Juniper Networks

      Email: vbeeram@juniper.net

      Igor Bryskin
      Individual
      Email: i_bryskin@yahoo.com