TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2017

T. Saad, Ed.
R. Gandhi
Cisco Systems Inc
X. Liu
Ericsson
V. Beeram
Juniper Networks
H. Shah
Ciena
I. Bryskin
X. Chen
Huawei Technologies
R. Jones
Brocade
B. Wen
Comcast

July 05, 2016

A YANG Data Model for Traffic Engineering Tunnels and Interfaces
draft-ietf-teas-yang-te-04

Abstract

This document defines a YANG data model for the configuration and management of Traffic Engineering (TE) interfaces, tunnels and Label Switched Paths (LSPs). The model is divided into YANG modules that classify data into generic, device-specific, technology agnostic, and technology-specific elements. The model also includes module(s) that contain reusable TE data types and data groupings.

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 6, 2017.

Table of Contents

1. Introduction

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

This document covers the YANG data model for the TE generic, TE device-specific, TE MPLS technology, TE Segment Routing (SR) data models. It also describes helper modules that define TE generic grouping(s) and data types that can be imported by other modules whenever necessary..

The document defines the high-level relationship between these modules and to other external protocol modules. It is expected that other data plane technology model(s) will augment the TE generic model. Also, the generic model does not include any signaling protocol specific data, and it is expected other TE signaling protocol modules (e.g. RSVP-TE ([RFC3209], [RFC3473]), and Segment-Routing TE (SR-TE)) will augment the TE generic model.

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

1.2. Tree Diagram

A simplified graphical representation of the data model is presented in each section of the model. The following notations are used for the YANG model data tree representation.

```
   <status> <flags> <name> <opts> <type>

    <status> is one of:
      +  for current
      x  for deprecated
      o  for obsolete

    <flags> is one of:
      rw  for read-write configuration data
      ro  for read-only non-configuration data
      -x  for execution rpcs
      -n  for notifications

    <name> is the name of the node

   If the node is augmented into the tree from another module, its name
   is printed as <prefix>:<name>

    <opts> is one of:
      ? for an optional leaf or node
      ! for a presence container
      * for a leaf-list or list
      Brackets [<keys>] for a list's keys
      Curly braces {<condition>} for optional feature that make node
   conditional
      Colon : for marking case nodes
      Ellipses ("...") subtree contents not shown

      Parentheses enclose choice and case nodes, and case nodes are also
      marked with a colon (":").

    <type> is the name of the type for leafs and leaf-lists.
```

1.3. Prefixes in Data Node Names

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

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

Table 1: Prefixes and corresponding YANG modules

1.4. Open Issues and Next Steps

This section describes the number of open issues that are under consideration. As issues are resolved, this section will be updated to reflect this and be left there for reference. It is expected that all the issues in this section will be addressed before the document will be ready for final publication.

1.4.1. TE Technology Models

This document describes the generic TE YANG data model that is independent of any dataplane technology. One of the design objectives is to allow specific data plane technologies models to reuse the generic TE data model and possibly augment it with technology specific data model(s).There are multiple options being considered to achieve this:

- The generic TE model, including the lists of TE tunnels, LSPs, and interfaces is defined and rooted at the top of the YANG tree. Specific leaf(s) under the TE tunnel, LSP, or interface identify the technology layer that it belongs to. This approach implies a single list for each of TE tunnel(s), LSP(s), and interface(s) in the model carries elements of different technology layers.

- An instance of the generic TE YANG model can be mounted in the YANG tree once for each TE technology layer(s). This approach provides separation of elements belonging to different technology layers into separate lists per layer in the data model. For example, the proposal(s) in [I-D.bjorklund-netmod-structural-mount] and [I-D.clemm-netmod-mount] allow for this capability by "mounting" the YANG model at a specific target.

- The generic TE data node(s) and TE list(s) for tunnels, LSPs, and interfaces are defined as grouping(s) in a separate module. The specific technology layer imports the generic TE groupings and uses them their respective technology specific module.

The options above are currently under investigation to determine the best suited approach.

Finally, the TE generic model does not include signaling protocol data. It is expected TE signaling protocol modules defined in other document(s) that cover RSVP-TE ([RFC3209], [RFC3473]), and Segment-Routing TE (SR-TE) will augment the TE generic model.

1.4.2. State Data Organization

Pure state data (for example, ephemeral or protocol derived state objects) can be modeled using one of the options below:

- Contained inside a read-write container, in a "state" sub-container, as shown in Figure 3

- Contained inside a separate read-only container, for example a lsps-state container

The first option allows for placing configuration data in the read-write "config" sub-container, and by placing state data under the read-only "state" sub-container of the parent container. However, when using approach for ephemeral or purely derived state (e.g. auto tunnels), and since in this case the state sub-container hangs off a read-write parent container, it will be possible to delete or modify the parent container and subsequently the ephemeral read-only state contained within (see Figure 3).

The second option entails defining a new read-only parent container in the model (e.g. neighbors-state) that holds the data.

This revision of the draft adopts the first option for ephemeral or state derived tunnels. Further discussions on this topic are expected to close on the best choice to adopt.

2. Model Overview

The data model defined in this document covers the core TE features that are commonly supported across different vendor implementations. The support of extended or vendor specific TE feature(s) are expected to be in augmentations to the data models defined in this document.

Throughout the model, the approach described in [I-D.openconfig-netmod-opstate] is adopted to represent data pertaining to configuration intended state, applied state and derived state data elements. Each container in the model hold a "config" and "state" sub-container. The "config" sub-container is used to represent the intended configurable parameters, and the state sub-container is used to represent both the applied configurable parameters and any derived state, such as counters or statistics information.

The decision to use this approach was made to better align with the MPLS consolidated model in [I-D.openconfig-mpls-consolidated-model] and maximize reusability of groupings defined in this document and allow for possible convergence between the two models.

2.1. Module(s) Relationship

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

The TE data relevant to a specific instantiations of data plane technology exists in a separate YANG module(s) that augment the TE generic model. For example, the MPLS-TE module "ietf-te-mpls.yang" is defined in Figure 10 and augments the TE generic model as shown in Figure 1. Similarly, the module "ietf-te-sr-mpls.yang" models the Segment Routing (SR) TE specific data and augments the TE generic and MPLS-TE model(s).

The TE data relevant to a TE specific signaling protocol instantiation is outside the scope and is covered in other documents. For example, the RSVP-TE [RFC3209] YANG model augmentation of the TE model is covered in [I-D.ietf-teas-yang-rsvp], and other signaling protocol model(s) (e.g. for Segment-Routing TE) are expected to also augment the TE generic model.

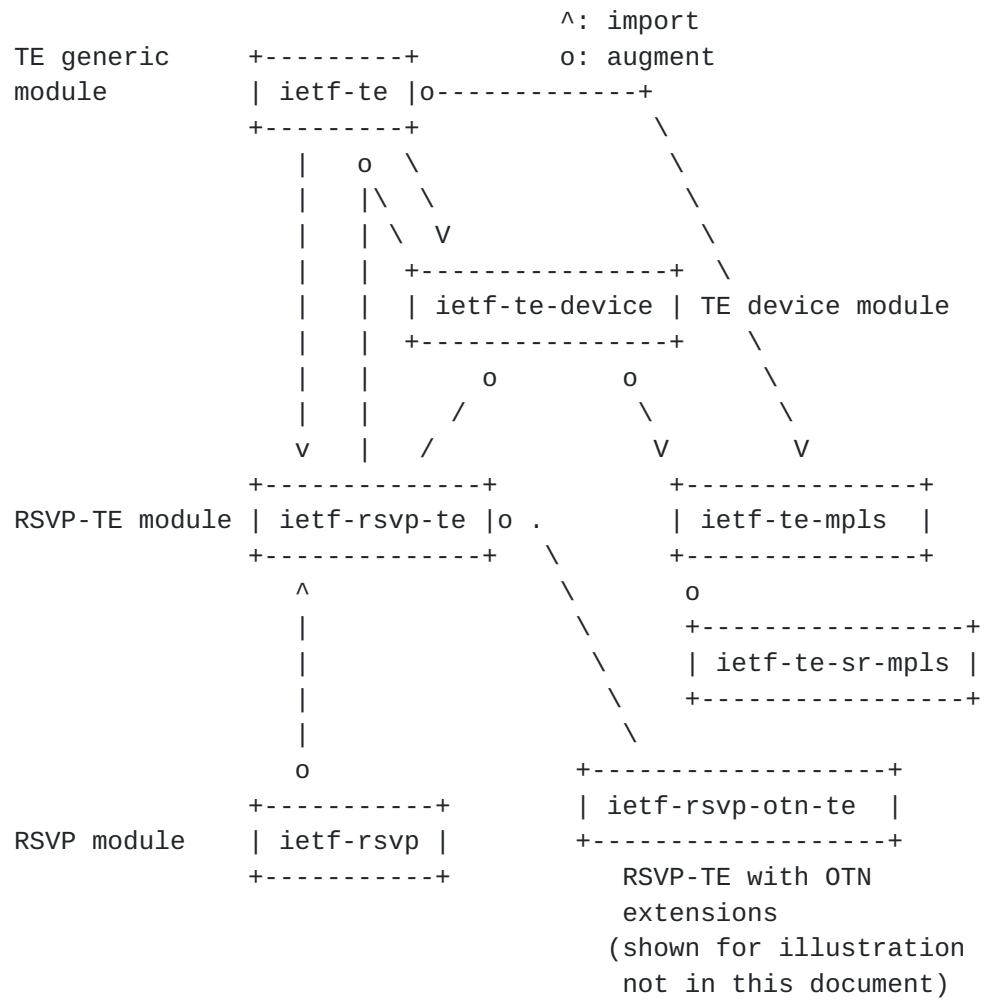

Figure 1: Relationship of TE module(s) with other signaling protocol modules

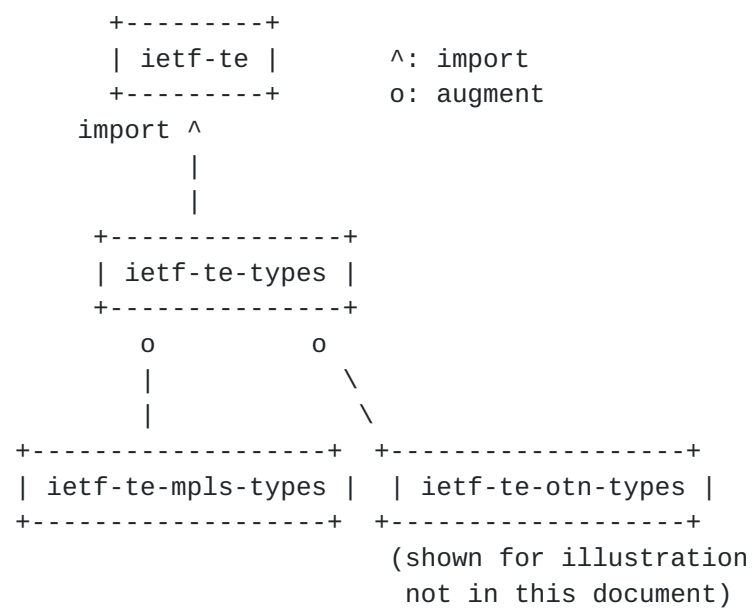

Figure 2: Relationship between generic and technology specific TE types modules

2.2. Design Considerations

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

- reusable data elements are grouped into separate TE types module(s) that can be readily imported by other modules whenever needed

- reusable TE data types that are data plane independent are grouped in the TE generic types module "ietf-te-types.yang"

- reusable TE data elements that are data plane specific (e.g. packet MPLS or switching technologies as defined in [RFC3473]) are expected to be grouped in a technology- specific types module, e.g. "ietf-te-mpls-types.yang". It is expected that technology specific types will augment TE generic types as shown in Figure 2

- The TE generic model contains device independent data and can be used to model data off a device (e.g. on a controller). The TE data that is device-specific are grouped in a separate module as shown in Figure 1.

- In general, little information in the model is designated as "mandatory", to allow freedom to vendors to adapt the data model to their specific product implementation.

2.3. Optional Features

Optional features that are beyond the base TE model and are left to the specific vendor to decide support using vendor model augmentation and/or using feature checks.

This model declares a number of TE functions as features (such as P2MP-TE, soft-preemption etc.).

2.4. Configuration Inheritance

The defined data model supports configuration inheritance for tunnels, paths, and interfaces. Data elements defined in the main container (e.g. that encompasses the list of tunnels, interfaces, or paths) are assumed to apply equally to all elements of the list, unless overridden explicitly for a certain element of a list (e.g. a tunnel, interface or path).

2.5. Routing Instance Support

There two main popular types of routing protocol configuration that vendors may support:

- protocol centric - all the protocol related configuration is contained within the protocol itself. Configuration belonging to multiple instances of the protocol running in different routing-instances (e.g. VRFs) are contained under the default routing instance [I-D.ietf-netmod-routing-cfg]:

- VRF centric - all the protocol related configuration for a routing- instance is contained within this routing-instance.

On-going discussions within the IETF community have converged on adopting the VRF centric approach. The proposed model in this document adheres to this conclusion.

3. TE Generic Model Organization

This model covers configuration, state, RPC, and notifications data pertaining to TE global parameters, interfaces, and tunnels parameters.

The container "te" is the top level container in this data model. The presence of this container is expected to enable TE function system wide.

The approach described in [I-D.openconfig-netmod-opstate] allows for modeling the intended and respective applied and derived state. The

TE state data in this model falls into one of the following categories:

- State corresponding to applied configuration
- State corresponding to derived state, counters, stats, etc.
- State corresponding to ephemeral data (e.g. LSPs, etc.)

Data for the first two categories are contained under the respective "state" sub-container of the intended (e.g. tunnel). The last category falls under a separate - e.g. lsps-state- container that contains the attributes of a purely derived state data (e.g. ephemeral objects) that are not associated with any configuration as shown in Figure 3.

```
module: ietf-te
   +--rw te!
      +--rw globals
            +-- rw config
               <<intended configuration>>
            .
            +-- ro state
               <<applied configuration>>
               <<derived state associated with the tunnel>>
         .
         .

      +--rw tunnels
            +-- rw config
               <<intended configuration>>
            .
            +-- ro state
               <<applied configuration>>
               <<derived state associated with the tunnel>>
         .
         .
rpcs:
   +---x globals-rpc
   +---x tunnels-rpc
notifications:
   +---n globals-notif
   +---n tunnels-notif
```

Figure 3: TE generic highlevel model view

3.1. Global Configuration and State Data

This branch of the data model covers configurations that control TE features behavior system-wide, and its respective state. Examples of such configuration data are:

- Table of named SRLG mappings
- Table of named (extended) administrative groups mappings
- Table of named explicit paths to be referenced by TE tunnels
- Table of named path-constraints sets
- Auto-bandwidth global parameters
- TE diff-serve TE-class maps
- System-wide capabilities for LSP reoptimization (included in the TE device model)
  - Reoptimization timers (periodic interval, LSP installation and cleanup)
- System-wide capabilities for TE state flooding (included in the TE device model)
  - Periodic flooding interval
- Global capabilities that affect the originating, traversing and terminating LSPs. For example:
  - Path selection parameters (e.g. metric to optimize, etc.)
  - Path or segment protection parameters

The approach described in [I-D.openconfig-netmod-opstate] is utilized to include the global state data under the global "state" sub-container as shown in Figure 3.

Examples of such states are:

- Global statistics (signaling, admission, preemption, flooding)
- Global counters (number of tunnels/LSPs/interfaces)

3.2. Interfaces Configuration and State Data

This branch of the model covers configuration and state data items, corresponding to TE interfaces that are present on a specific device. A new module is introduced that holds the TE device specific properties.

Examples of TE interface properties are:

- Maximum reservable bandwidth, bandwidth constraints (BC)
- Flooding parameters
  - Flooding intervals and threshold values
- Fast reroute backup tunnel properties (such as static, auto-tunnel)
- interface attributes
  - (Extended) administrative groups
  - SRLG values
  - TE metric value

The state corresponding to the TE interfaces applied configuration, protocol derived state, and stats and counters all fall under the interface "state" sub-container as shown in Figure 4 below:

```
module: ietf-te
   +--rw te!
      +--rw interfaces
         .
         +-- rw te-attributes
             +-- rw config
                <<intended configuration>>
             .
             +-- ro state
                <<applied configuration>>
                <<derived state associated with the TE interface>>
```

Figure 4: TE interface state

This covers state data for TE interfaces such as:

- 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

3.3. Tunnels Configuration and State Data

This branch of the model covers intended, and corresponding applied configuration for tunnels. As well, it holds possible derived state pertaining to TE tunnels.

The approach described in [I-D.openconfig-netmod-opstate] is utilized for the inclusion of operational and statistical data as shown in Figure 5.

```
module: ietf-te
   +--rw te!
      +--rw tunnels
         .
         +-- rw config
            <<intended configuration>>
         .
         +-- ro state
            <<applied configuration>>
            <<derived state associated with the tunnel>>
```

Figure 5: TE interface state tree

Examples of tunnel configuration date for TE tunnels:

- Name and type (e.g. P2P, P2MP) of the TE tunnel
- Admin-state
- Set of primary and corresponding secondary paths
- Routing usage (auto-route announce, forwarding adjacency)
- Policy based routing (PBR) parameters

3.4. TE LSPs State Data

TE LSPs are derived state data that is usually instantiated via signaling protocols. TE LSPs exists on routers as ingress (starting point of LSP), transit (mid-point of LSP), or egress (termination point of the LSP). TE LSPs are distinguished by the 5 tuple, and LSP type (P2P or P2MP). In the model, the nodes holding LSPs data exist in the read-only lsps-state list as show in Figure 6.

3.5. Global RPC Data

This branch of the model covers system-wide RPC execution data to trigger actions and optionally expect responses. Examples of such TE commands are to:

- Clear global TE statistics of various features

3.6. Interface RPC Data

This collection of data in the model defines TE interface RPC execution commands. Examples of these are to:

- Clear TE statistics for all or for individual TE interfaces

- Trigger immediate flooding for one or all TE interfaces

3.7. Tunnel RPC Data

This branch of the model covers TE tunnel RPC execution data to trigger actions and optionally expect responses. Examples of such TE commands are:

- Clear statistics for all or for individual tunnels

- Trigger the tear and setup of existing tunnels or LSPs.

3.8. Global Notifications Data

This branch of the model covers system-wide notifications data. The node notifies the registered events to the server using the defined notification messages. Example of such global TE events are:

- Backup tunnel FRR active and not-active state transition events

3.9. Interfaces Notifications Data

This branch of the model covers TE interfaces related notifications data. The TE interface configuration is used for specific events registration. Notifications are sent for registered events to the server. Example events for TE interfaces are:

- Interface creation and deletion
- Interface state transitions
- (Soft) preemption triggers
- Fast reroute activation

3.10. Tunnel Notification Data

This branch of the model covers TE tunnels related notifications data. The TE tunnels configuration is used for specific events registration. Notifications are sent for registered events to the server. Example events for TE tunnels are:

- Tunnel creation and deletion events
- Tunnel state up/down changes
- Tunnel state reoptimization changes

Figure Figure 6 below shows the tree diagram of the YANG model defined in modules: ietf-te.yang, ietf-te-device.yang, ietf-te-mpls.yang, and ietf-te-sr.yang.

```
module: ietf-te
   +--rw te!
      +--rw globals
      |  +--rw config
      |  |  +--rw named-admin-groups* [name]
      {te-types:extended-admin-groups,
      te-types:named-extended-admin-groups}?
      |  |  |  +--rw name            string
      |  |  |  +--rw bit-position?   uint32
      |  |  +--rw named-srlgs* [name] {te-types:named-srlg-groups}?
      |  |  |  +--rw name     string
      |  |  |  +--rw group?   te-types:srlg
      |  |  +--rw named-explicit-paths* [name]
      |  |  |  +--rw name                      string
      |  |  |  +--rw explicit-route-objects* [index]
      |  |  |     +--rw index                   uint8
```

```
|  |  |     +--rw explicit-route-usage?   identityref
|  |  |     +--rw (type)?
|  |  |        +--:(ipv4-address)
|  |  |        |  +--rw v4-address? inet:ipv4-address
|  |  |        |  +--rw v4-prefix-length?       uint8
|  |  |        |  +--rw v4-loose?               boolean
|  |  |        +--:(ipv6-address)
|  |  |        |  +--rw v6-address? inet:ipv6-address
|  |  |        |  +--rw v6-prefix-length?       uint8
|  |  |        |  +--rw v6-loose?               boolean
|  |  |        +--:(as-number)
|  |  |        |  +--rw as-number?              uint16
|  |  |        +--:(unnumbered-link)
|  |  |        |  +--rw router-id?              inet:ip-address
|  |  |        |  +--rw interface-id?           uint32
|  |  |        +--:(label)
|  |  |           +--rw value?                 uint32
|  |  +--rw named-constraints* [name]
{te-types:named-path-constraints}?
|  |  |  +--rw name                               string
|  |  |  +--rw path-selection
|  |  |  |  +--rw topology-id? te-types:te-topology-id
|  |  |  |  +--rw cost-limit?               uint32
|  |  |  |  +--rw hop-limit?                uint8
|  |  |  |  +--rw metric-type?              identityref
|  |  |  |  +--rw tiebreaker-type?          identityref
|  |  |  |  +--rw ignore-overload?          boolean
|  |  |  |  +--rw tunnel-path-affinities
{named-path-affinities}?
|  |  |  |  |  +--rw (style)?
|  |  |  |  |     +--:(values)
|  |  |  |  |     |  +--rw value?         uint32
|  |  |  |  |     |  +--rw mask?          uint32
|  |  |  |  |     +--:(named)
|  |  |  |  |        +--rw constraints* [usage]
|  |  |  |  |           +--rw usage         identityref
|  |  |  |  |           +--rw constraint
|  |  |  |  |              +--rw affinity-names* [name]
|  |  |  |  |                 +--rw name    string
|  |  |  |  +--rw tunnel-path-srlgs
|  |  |  |     +--rw (style)?
|  |  |  |        +--:(values)
|  |  |  |        |  +--rw usage?         identityref
|  |  |  |        |  +--rw values*        te-types:srlg
|  |  |  |        +--:(named)
|  |  |  |           +--rw constraints* [usage]
|  |  |  |              +--rw usage         identityref
|  |  |  |              +--rw constraint
```

```
|  |  |  |                +--rw srlg-names* [name]
|  |  |  |                   +--rw name    string
|  |  |  +--rw te-sr-mpls:adjacency-protection?   identityref
|  |  +--rw te-dev:lsp-install-interval?        uint32
|  |  +--rw te-dev:lsp-cleanup-interval?        uint32
|  |  +--rw te-dev:lsp-invalidation-interval?   uint32
|  +--ro state
|     +--ro named-admin-groups* [name]
{te-types:extended-admin-groups,
te-types:named-extended-admin-groups}?
|     |  +--ro name            string
|     |  +--ro bit-position?   uint32
|     +--ro named-srlgs* [name] {te-types:named-srlg-groups}?
|     |  +--ro name     string
|     |  +--ro group?   te-types:srlg
|     +--ro named-explicit-paths* [name]
|     |  +--ro name                      string
|     |  +--ro explicit-route-objects* [index]
|     |     +--ro index                   uint8
|     |     +--ro explicit-route-usage?   identityref
|     |     +--ro (type)?
|     |        +--:(ipv4-address)
|     |        |  +--ro v4-address? inet:ipv4-address
|     |        |  +--ro v4-prefix-length?       uint8
|     |        |  +--ro v4-loose?               boolean
|     |        +--:(ipv6-address)
|     |        |  +--ro v6-address? inet:ipv6-address
|     |        |  +--ro v6-prefix-length?       uint8
|     |        |  +--ro v6-loose?               boolean
|     |        +--:(as-number)
|     |        |  +--ro as-number?              uint16
|     |        +--:(unnumbered-link)
|     |        |  +--ro router-id?              inet:ip-address
|     |        |  +--ro interface-id?           uint32
|     |        +--:(label)
|     |           +--ro value?                 uint32
|     +--ro named-constraints* [name]
{te-types:named-path-constraints}?
|     |  +--ro name                              string
|     |  +--ro path-selection
|     |  |  +--ro topology-id? te-types:te-topology-id
|     |  |  +--ro cost-limit?               uint32
|     |  |  +--ro hop-limit?                uint8
|     |  |  +--ro metric-type?              identityref
|     |  |  +--ro tiebreaker-type?          identityref
|     |  |  +--ro ignore-overload?          boolean
|     |  |  +--ro tunnel-path-affinities
{named-path-affinities}?
```

```
|     |  |  |  +--ro (style)?
|     |  |  |     +--:(values)
|     |  |  |     |  +--ro value?         uint32
|     |  |  |     |  +--ro mask?          uint32
|     |  |  |     +--:(named)
|     |  |  |        +--ro constraints* [usage]
|     |  |  |           +--ro usage         identityref
|     |  |  |           +--ro constraint
|     |  |  |              +--ro affinity-names* [name]
|     |  |  |                 +--ro name    string
|     |  |  +--ro tunnel-path-srlgs
|     |  |     +--ro (style)?
|     |  |        +--:(values)
|     |  |        |  +--ro usage?         identityref
|     |  |        |  +--ro values*        te-types:srlg
|     |  |        +--:(named)
|     |  |           +--ro constraints* [usage]
|     |  |              +--ro usage         identityref
|     |  |              +--ro constraint
|     |  |                 +--ro srlg-names* [name]
|     |  |                    +--ro name    string
|     |  +--ro te-sr-mpls:adjacency-protection?   identityref
|     +--ro te-dev:lsp-install-interval?        uint32
|     +--ro te-dev:lsp-cleanup-interval?        uint32
|     +--ro te-dev:lsp-invalidation-interval?   uint32
|     +--ro te-dev:tunnels-counter?             uint32
|     +--ro te-dev:lsps-counter?                uint32
+--rw tunnels
|  +--rw tunnel* [name type]
|     +--rw name             -> ../config/name
|     +--rw type             -> ../config/type
|     +--rw identifier?      -> ../config/identifier
|     +--rw config
|     |  +--rw name?                              string
|     |  +--rw type?                              identityref
|     |  +--rw identifier?                        uint16
|     |  +--rw description?                       string
|     |  +--rw lsp-priority-setup?                uint8
|     |  +--rw lsp-priority-hold?                 uint8
|     |  +--rw lsp-protection-type?               identityref
|     |  +--rw admin-status?                      identityref
|     |  +--rw source?             inet:ip-address
|     |  +--rw destination?        inet:ip-address
|     |  +--rw src-tp-id?                         binary
|     |  +--rw dst-tp-id?                         binary
|     |  +--rw hierarchical-link-id
|     |  |  +--rw local-te-node-id?     te-types:te-node-id
|     |  |  +--rw local-te-link-tp-id?  te-types:te-tp-id
```

```
|     |  |  +--rw remote-te-node-id?    te-types:te-node-id
|     |  |  +--rw te-topology-id?       te-types:te-topology-id
|     |  +--rw bidirectional
|     |  |  +--rw association
|     |  |     +--rw id?              uint16
|     |  |     +--rw source?          inet:ip-address
|     |  |     +--rw global-source?   inet:ip-address
|     |  |     +--rw type?            identityref
|     |  |     +--rw provisioing?     identityref
|     |  +--rw te-dev:lsp-install-interval?        uint32
|     |  +--rw te-dev:lsp-cleanup-interval?        uint32
|     |  +--rw te-dev:lsp-invalidation-interval?   uint32
|     |  +--rw (te-mpls:routing-choice)?
|     |  |  +--:(te-mpls:autoroute)
|     |  |  |  +--rw te-mpls:autoroute-announce!
|     |  |  |     +--rw te-mpls:routing-afs*   inet:ip-version
|     |  |  |     +--rw (te-mpls:metric-type)?
|     |  |  |        +--:(te-mpls:metric)
|     |  |  |        |  +--rw te-mpls:metric?            uint32
|     |  |  |        +--:(te-mpls:relative-metric)
|     |  |  |        |  +--rw te-mpls:relative-metric?   int32
|     |  |  |        +--:(te-mpls:absolute-metric)
|     |  |  |           +--rw te-mpls:absolute-metric?   uint32
|     |  |  +--:(te-mpls:forwarding-adjacency)
|     |  |     +--rw te-mpls:forwarding-adjacency!
|     |  |        +--rw te-mpls:holdtime?      uint32
|     |  |        +--rw te-mpls:routing-afs*   inet:ip-version
|     |  +--rw te-mpls:forwarding
|     |     +--rw te-mpls:binding-label?   mpls:mpls-label
|     |     +--rw te-mpls:load-share?      uint32
|     |     +--rw (te-mpls:policy-type)?
|     |        +--:(te-mpls:class)
|     |        |  +--rw te-mpls:class
|     |        |     +--rw te-mpls:class?   uint8
|     |        +--:(te-mpls:group)
|     |           +--rw te-mpls:group
|     |              +--rw te-mpls:classes*   uint8
|     +--ro state
|     |  +--ro name?                              string
|     |  +--ro type?                              identityref
|     |  +--ro identifier?                        uint16
|     |  +--ro description?                       string
|     |  +--ro lsp-priority-setup?                uint8
|     |  +--ro lsp-priority-hold?                 uint8
|     |  +--ro lsp-protection-type?               identityref
|     |  +--ro admin-status?                      identityref
|     |  +--ro source?      inet:ip-address
|     |  +--ro destination? inet:ip-address
```

```
|     |  +--ro src-tp-id?                          binary
|     |  +--ro dst-tp-id?                          binary
|     |  +--ro hierarchical-link-id
|     |  |  +--ro local-te-node-id?     te-types:te-node-id
|     |  |  +--ro local-te-link-tp-id?  te-types:te-tp-id
|     |  |  +--ro remote-te-node-id?    te-types:te-node-id
|     |  |  +--ro te-topology-id?       te-types:te-topology-id
|     |  +--ro bidirectional
|     |  |  +--ro association
|     |  |     +--ro id?              uint16
|     |  |     +--ro source?          inet:ip-address
|     |  |     +--ro global-source?   inet:ip-address
|     |  |     +--ro type?            identityref
|     |  |     +--ro provisioing?     identityref
|     |  +--ro oper-status?                        identityref
|     |  +--ro te-dev:lsp-install-interval?        uint32
|     |  +--ro te-dev:lsp-cleanup-interval?        uint32
|     |  +--ro te-dev:lsp-invalidation-interval?   uint32
|     |  +--ro (te-mpls:routing-choice)?
|     |  |  +--:(te-mpls:autoroute)
|     |  |  |  +--ro te-mpls:autoroute-announce!
|     |  |  |     +--ro te-mpls:routing-afs*       inet:ip-version
|     |  |  |     +--ro (te-mpls:metric-type)?
|     |  |  |        +--:(te-mpls:metric)
|     |  |  |        |  +--ro te-mpls:metric?            uint32
|     |  |  |        +--:(te-mpls:relative-metric)
|     |  |  |        |  +--ro te-mpls:relative-metric?   int32
|     |  |  |        +--:(te-mpls:absolute-metric)
|     |  |  |           +--ro te-mpls:absolute-metric?   uint32
|     |  |  +--:(te-mpls:forwarding-adjacency)
|     |  |     +--ro te-mpls:forwarding-adjacency!
|     |  |        +--ro te-mpls:holdtime?      uint32
|     |  |        +--ro te-mpls:routing-afs*   inet:ip-version
|     |  +--ro te-mpls:forwarding
|     |     +--ro te-mpls:binding-label?   mpls:mpls-label
|     |     +--ro te-mpls:load-share?      uint32
|     |     +--ro (te-mpls:policy-type)?
|     |        +--:(te-mpls:class)
|     |        |  +--ro te-mpls:class
|     |        |     +--ro te-mpls:class?   uint8
|     |        +--:(te-mpls:group)
|     |           +--ro te-mpls:group
|     |              +--ro te-mpls:classes*   uint8
|     +--rw primary-paths* [name]
|        +--rw name         -> ../config/name
|        +--rw preference?  -> ../config/preference
|        +--rw config
|        |  +--rw name?                    string
```

```
|        |  +--rw preference?              uint8
|        |  +--rw path-named-constraint?
string {te-types:named-path-constraints}?
|        |  +--rw path-selection
|        |  |  +--rw topology-id? te-types:te-topology-id
|        |  |  +--rw cost-limit?              uint32
|        |  |  +--rw hop-limit?               uint8
|        |  |  +--rw metric-type?             identityref
|        |  |  +--rw tiebreaker-type?         identityref
|        |  |  +--rw ignore-overload?         boolean
|        |  |  +--rw tunnel-path-affinities
{named-path-affinities}?
|        |  |  |  +--rw (style)?
|        |  |  |     +--:(values)
|        |  |  |     |  +--rw value?         uint32
|        |  |  |     |  +--rw mask?          uint32
|        |  |  |     +--:(named)
|        |  |  |        +--rw constraints* [usage]
|        |  |  |           +--rw usage         identityref
|        |  |  |           +--rw constraint
|        |  |  |              +--rw affinity-names* [name]
|        |  |  |                 +--rw name    string
|        |  |  +--rw tunnel-path-srlgs
|        |  |     +--rw (style)?
|        |  |        +--:(values)
|        |  |        |  +--rw usage?         identityref
|        |  |        |  +--rw values*        te-types:srlg
|        |  |        +--:(named)
|        |  |           +--rw constraints* [usage]
|        |  |              +--rw usage         identityref
|        |  |              +--rw constraint
|        |  |                 +--rw srlg-names* [name]
|        |  |                    +--rw name    string
|        |  +--rw (type)?
|        |  |  +--:(dynamic)
|        |  |  |  +--rw dynamic?                 empty
|        |  |  +--:(explicit)
|        |  |     +--rw explicit-path-name?      string
|        |  |     +--rw explicit-route-objects* [index]
|        |  |        +--rw index                  uint8
|        |  |        +--rw explicit-route-usage?  identityref
|        |  |        +--rw (type)?
|        |  |           +--:(ipv4-address)
|        |  |           |  +--rw v4-address?    inet:ipv4-address
|        |  |           |  +--rw v4-prefix-length?      uint8
|        |  |           |  +--rw v4-loose?              boolean
|        |  |           +--:(ipv6-address)
|        |  |           |  +--rw v6-address?    inet:ipv6-address
```

```
       |        |  |          |  +--rw v6-prefix-length?       uint8
       |        |  |          |  +--rw v6-loose?               boolean
       |        |  |          +--:(as-number)
       |        |  |          |  +--rw as-number?              uint16
       |        |  |          +--:(unnumbered-link)
       |        |  |          |  +--rw router-id?     inet:ip-address
       |        |  |          |  +--rw interface-id?           uint32
       |        |  |          +--:(label)
       |        |  |             +--rw value?                  uint32
       |        |  +--rw no-cspf?                 empty
       |        |  +--rw lockdown?                empty
       |        +--ro state
       |        |  +--ro path-named-constraint?    string
      {te-types:named-path-constraints}?
       |        |  +--ro path-selection
       |        |  |  +--ro topology-id? te-types:te-topology-id
       |        |  |  +--ro cost-limit?              uint32
       |        |  |  +--ro hop-limit?               uint8
       |        |  |  +--ro metric-type?             identityref
       |        |  |  +--ro tiebreaker-type?         identityref
       |        |  |  +--ro ignore-overload?         boolean
       |        |  |  +--ro tunnel-path-affinities
      {named-path-affinities}?
       |        |  |  |  +--ro (style)?
       |        |  |  |     +--:(values)
       |        |  |  |     |  +--ro value?         uint32
       |        |  |  |     |  +--ro mask?          uint32
       |        |  |  |     +--:(named)
       |        |  |  |        +--ro constraints* [usage]
       |        |  |  |           +--ro usage         identityref
       |        |  |  |           +--ro constraint
       |        |  |  |              +--ro affinity-names* [name]
       |        |  |  |                 +--ro name    string
       |        |  |  +--ro tunnel-path-srlgs
       |        |  |     +--ro (style)?
       |        |  |        +--:(values)
       |        |  |        |  +--ro usage?         identityref
       |        |  |        |  +--ro values*        te-types:srlg
       |        |  |        +--:(named)
       |        |  |           +--ro constraints* [usage]
       |        |  |              +--ro usage         identityref
       |        |  |              +--ro constraint
       |        |  |                 +--ro srlg-names* [name]
       |        |  |                    +--ro name    string
       |        |  +--ro (type)?
       |        |  |  +--:(dynamic)
       |        |  |  |  +--ro dynamic?                 empty
       |        |  |  +--:(explicit)
```

```
|        |  |     +--ro explicit-path-name?       string
|        |  |     +--ro explicit-route-objects* [index]
|        |  |        +--ro index                    uint8
|        |  |        +--ro explicit-route-usage?   identityref
|        |  |        +--ro (type)?
|        |  |           +--:(ipv4-address)
|        |  |           |  +--ro v4-address?   inet:ipv4-address
|        |  |           |  +--ro v4-prefix-length?       uint8
|        |  |           |  +--ro v4-loose?               boolean
|        |  |           +--:(ipv6-address)
|        |  |           |  +--ro v6-address?   inet:ipv6-address
|        |  |           |  +--ro v6-prefix-length?       uint8
|        |  |           |  +--ro v6-loose?               boolean
|        |  |           +--:(as-number)
|        |  |           |  +--ro as-number?              uint16
|        |  |           +--:(unnumbered-link)
|        |  |           |  +--ro router-id?    inet:ip-address
|        |  |           |  +--ro interface-id?           uint32
|        |  |           +--:(label)
|        |  |              +--ro value?                  uint32
|        |  +--ro no-cspf?                  empty
|        |  +--ro lockdown?                 empty
|        |  +--ro lsp*
[source destination tunnel-id lsp-id extended-tunnel-id type]
|        |     +--ro source
-> ../../../../../../lsps-state/lsp/source
|        |     +--ro destination
-> ../../../../../../lsps-state/lsp/destination
|        |     +--ro tunnel-id
-> ../../../../../../lsps-state/lsp/tunnel-id
|        |     +--ro lsp-id
-> ../../../../../../lsps-state/lsp/lsp-id
|        |     +--ro extended-tunnel-id
-> ../../../../../../lsps-state/lsp/extended-tunnel-id
|        |     +--ro type
-> ../../../../../../lsps-state/lsp/type
|        +--rw secondary-paths* [name]
|        |  +--rw name        -> ../config/name
|        |  +--rw preference? -> ../config/preference
|        |  +--rw config
|        |  |  +--rw name?                    string
|        |  |  +--rw preference?              uint8
|        |  |  +--rw path-named-constraint?   string
{te-types:named-path-constraints}?
|        |  |  +--rw path-selection
|        |  |  |  +--rw topology-id?
te-types:te-topology-id
|        |  |  |  +--rw cost-limit?              uint32
```

```
      |        |  |  |  +--rw hop-limit?               uint8
      |        |  |  |  +--rw metric-type?             identityref
      |        |  |  |  +--rw tiebreaker-type?         identityref
      |        |  |  |  +--rw ignore-overload?         boolean
      |        |  |  |  +--rw tunnel-path-affinities
      {named-path-affinities}?
      |        |  |  |  |  +--rw (style)?
      |        |  |  |  |     +--:(values)
      |        |  |  |  |     |  +--rw value?         uint32
      |        |  |  |  |     |  +--rw mask?          uint32
      |        |  |  |  |     +--:(named)
      |        |  |  |  |        +--rw constraints* [usage]
      |        |  |  |  |           +--rw usage         identityref
      |        |  |  |  |           +--rw constraint
      |        |  |  |  |              +--rw affinity-names* [name]
      |        |  |  |  |                 +--rw name    string
      |        |  |  |  +--rw tunnel-path-srlgs
      |        |  |  |     +--rw (style)?
      |        |  |  |        +--:(values)
      |        |  |  |        |  +--rw usage?         identityref
      |        |  |  |        |  +--rw values*        te-types:srlg
      |        |  |  |        +--:(named)
      |        |  |  |           +--rw constraints* [usage]
      |        |  |  |              +--rw usage         identityref
      |        |  |  |              +--rw constraint
      |        |  |  |                 +--rw srlg-names* [name]
      |        |  |  |                    +--rw name    string
      |        |  |  +--rw (type)?
      |        |  |  |  +--:(dynamic)
      |        |  |  |  |  +--rw dynamic?                empty
      |        |  |  |  +--:(explicit)
      |        |  |  |     +--rw explicit-path-name?       string
      |        |  |  |     +--rw explicit-route-objects* [index]
      |        |  |  |        +--rw index                   uint8
      |        |  |  |        +--rw explicit-route-usage?   identityref
      |        |  |  |        +--rw (type)?
      |        |  |  |           +--:(ipv4-address)
      |        |  |  |           |  +--rw v4-address?  inet:ipv4-address
      |        |  |  |           |  +--rw v4-prefix-length?       uint8
      |        |  |  |           |  +--rw v4-loose?    boolean
      |        |  |  |           +--:(ipv6-address)
      |        |  |  |           |  +--rw v6-address?  inet:ipv6-address
      |        |  |  |           |  +--rw v6-prefix-length?       uint8
      |        |  |  |           |  +--rw v6-loose?    boolean
      |        |  |  |           +--:(as-number)
      |        |  |  |           |  +--rw as-number?             uint16
      |        |  |  |           +--:(unnumbered-link)
      |        |  |  |           |  +--rw router-id?   inet:ip-address
```

```
|        |  |  |           |  +--rw interface-id?          uint32
|        |  |  |           +--:(label)
|        |  |  |              +--rw value?                uint32
|        |  |  +--rw no-cspf?                  empty
|        |  |  +--rw lockdown?                 empty
|        |  +--ro state
|        |  |  +--ro name?                     string
|        |  |  +--ro preference?               uint8
|        |  |  +--ro path-named-constraint?    string
{te-types:named-path-constraints}?
|        |  |  +--ro path-selection
|        |  |  |  +--ro topology-id? te-types:te-topology-id
|        |  |  |  +--ro cost-limit?              uint32
|        |  |  |  +--ro hop-limit?               uint8
|        |  |  |  +--ro metric-type?             identityref
|        |  |  |  +--ro tiebreaker-type?         identityref
|        |  |  |  +--ro ignore-overload?         boolean
|        |  |  |  +--ro tunnel-path-affinities
{named-path-affinities}?
|        |  |  |  |  +--ro (style)?
|        |  |  |  |     +--:(values)
|        |  |  |  |     |  +--ro value?         uint32
|        |  |  |  |     |  +--ro mask?          uint32
|        |  |  |  |     +--:(named)
|        |  |  |  |        +--ro constraints* [usage]
|        |  |  |  |           +--ro usage         identityref
|        |  |  |  |           +--ro constraint
|        |  |  |  |              +--ro affinity-names* [name]
|        |  |  |  |                 +--ro name    string
|        |  |  |  +--ro tunnel-path-srlgs
|        |  |  |     +--ro (style)?
|        |  |  |        +--:(values)
|        |  |  |        |  +--ro usage?         identityref
|        |  |  |        |  +--ro values*        te-types:srlg
|        |  |  |        +--:(named)
|        |  |  |           +--ro constraints* [usage]
|        |  |  |              +--ro usage         identityref
|        |  |  |              +--ro constraint
|        |  |  |                 +--ro srlg-names* [name]
|        |  |  |                    +--ro name    string
|        |  |  +--ro (type)?
|        |  |  |  +--:(dynamic)
|        |  |  |  |  +--ro dynamic?                 empty
|        |  |  |  +--:(explicit)
|        |  |  |     +--ro explicit-path-name?       string
|        |  |  |     +--ro explicit-route-objects* [index]
|        |  |  |        +--ro index                   uint8
|        |  |  |        +--ro explicit-route-usage?   identityref
```

```
      |        |  |  |        +--ro (type)?
      |        |  |  |           +--:(ipv4-address)
      |        |  |  |           |  +--ro v4-address?  inet:ipv4-address
      |        |  |  |           |  +--ro v4-prefix-length?       uint8
      |        |  |  |           |  +--ro v4-loose?    boolean
      |        |  |  |           +--:(ipv6-address)
      |        |  |  |           |  +--ro v6-address?  inet:ipv6-address
      |        |  |  |           |  +--ro v6-prefix-length?       uint8
      |        |  |  |           |  +--ro v6-loose?    boolean
      |        |  |  |           +--:(as-number)
      |        |  |  |           |  +--ro as-number?              uint16
      |        |  |  |           +--:(unnumbered-link)
      |        |  |  |           |  +--ro router-id?   inet:ip-address
      |        |  |  |           |  +--ro interface-id?           uint32
      |        |  |  |           +--:(label)
      |        |  |  |              +--ro value?                  uint32
      |        |  |  +--ro no-cspf?                  empty
      |        |  |  +--ro lockdown?                 empty
      |        |  |  +--ro lsp* [source]
      |        |  |     +--ro source
      -> ../../../../../../../../lsps-state/lsp/source
      |        |  |     +--ro destination?
      -> ../../../../../../../../lsps-state/lsp/destination
      |        |  |     +--ro tunnel-id?
      -> ../../../../../../../../lsps-state/lsp/tunnel-id
      |        |  |     +--ro lsp-id?
      -> ../../../../../../../../lsps-state/lsp/lsp-id
      |        |  |     +--ro extended-tunnel-id?
      -> ../../../../../../../../lsps-state/lsp/extended-tunnel-id
      |        |  |     +--ro type?
      -> ../../../../../../../../lsps-state/lsp/type
      |        |  +--rw te-sr-mpls:path-signaling-type?   identityref
      |        +--rw te-sr-mpls:path-signaling-type?   identityref
      +--ro lsps-state
      |  +--ro lsp*
      [source destination tunnel-id lsp-id extended-tunnel-id type]
      |     +--ro source                   inet:ip-address
      |     +--ro destination              inet:ip-address
      |     +--ro tunnel-id                uint16
      |     +--ro lsp-id                   uint16
      |     +--ro extended-tunnel-id       inet:ip-address
      |     +--ro type                     identityref
      |     +--ro oper-status?             identityref
      |     +--ro origin-type?             enumeration
      |     +--ro lsp-resource-status?     enumeration
      |     +--ro lsp-protection-role?     enumeration
      |     +--ro lsp-operational-status?  empty
      |     +--ro lsp-record-route
```

```
|     |  +--ro record-route-subobjects* [subobject-index]
|     |     +--ro subobject-index     uint32
|     |     +--ro (type)?
|     |        +--:(ipv4-address)
|     |        |  +--ro v4-address?         inet:ipv4-address
|     |        |  +--ro v4-prefix-length?   uint8
|     |        |  +--ro v4-flags?           uint8
|     |        +--:(ipv6-address)
|     |        |  +--ro v6-address?         inet:ipv6-address
|     |        |  +--ro v6-prefix-length?   uint8
|     |        |  +--ro v6-flags?           uint8
|     |        +--:(unnumbered-link)
|     |        |  +--ro router-id?          inet:ip-address
|     |        |  +--ro interface-id?       uint32
|     |        +--:(label)
|     |           +--ro value?             uint32
|     |           +--ro flags?             uint8
|     +--ro te-dev:lsp-timers
|     |  +--ro te-dev:life-time?         uint32
|     |  +--ro te-dev:time-to-install?   uint32
|     |  +--ro te-dev:time-to-die?       uint32
|     +--ro te-dev:downstream-info
|     |  +--ro te-dev:nhop?                 inet:ip-address
|     |  +--ro te-dev:outgoing-interface?   if:interface-ref
|     |  +--ro te-dev:neighbor?             inet:ip-address
|     |  +--ro te-dev:label?                uint32
|     +--ro te-dev:upstream-info
|        +--ro te-dev:phop?       inet:ip-address
|        +--ro te-dev:neighbor?   inet:ip-address
|        +--ro te-dev:label?      uint32
+--rw te-dev:interfaces
   +--rw te-dev:config
   |  +--rw te-dev:thresholds
   |     +--rw (te-dev:type)?
   |        +--:(te-dev:equal-steps)
   |        |  +--rw (te-dev:equal-step-type)?
   |        |     +--:(te-dev:up-down-different-step)
   |        |     |  +--rw te-dev:up-step?      uint8
   |        |     |  +--rw te-dev:down-step?    uint8
   |        |     +--:(te-dev:up-down-same-step)
   |        |        +--rw te-dev:step?         uint8
   |        +--:(te-dev:unequal-steps)
   |           +--rw te-dev:up-steps* [value]
   |           |  +--rw te-dev:value    uint8
   |           +--rw te-dev:down-steps* [value]
   |              +--rw te-dev:value    uint8
   +--ro te-dev:state
   |  +--ro te-dev:thresholds
```

```
|     +--ro (te-dev:type)?
|        +--:(te-dev:equal-steps)
|        |  +--ro (te-dev:equal-step-type)?
|        |     +--:(te-dev:up-down-different-step)
|        |     |  +--ro te-dev:up-step?      uint8
|        |     |  +--ro te-dev:down-step?    uint8
|        |     +--:(te-dev:up-down-same-step)
|        |        +--ro te-dev:step?         uint8
|        +--:(te-dev:unequal-steps)
|           +--ro te-dev:up-steps* [value]
|           |  +--ro te-dev:value    uint8
|           +--ro te-dev:down-steps* [value]
|              +--ro te-dev:value    uint8
+--rw te-dev:interface* [interface]
   +--rw te-dev:interface    if:interface-ref
   +--rw te-dev:config
   |  +--rw te-dev:te-metric?               te-types:te-metric
   |  +--rw (te-dev:admin-group-type)?
   |  |  +--:(te-dev:value-admin-groups)
   |  |  |  +--rw (te-dev:value-admin-group-type)?
   |  |  |     +--:(te-dev:admin-groups)
   |  |  |     |  +--rw te-dev:admin-group?te-types:admin-group
   |  |  |     +--:(te-dev:extended-admin-groups)
   {te-types:extended-admin-groups}?
   |  |  |        +--rw te-dev:extended-admin-group?
   te-types:extended-admin-group
   |  |  +--:(te-dev:named-admin-groups)
   |  |     +--rw te-dev:named-admin-groups*[named-admin-group]
   {te-types:extended-admin-groups,
   te-types:named-extended-admin-groups}?
   |  |        +--rw te-dev:named-admin-group
   -> ../../../../../te:globals/config/named-admin-groups/name
   |  +--rw (te-dev:srlg-type)?
   |  |  +--:(te-dev:value-srlgs)
   |  |  |  +--rw te-dev:values* [value]
   |  |  |     +--rw te-dev:value    uint32
   |  |  +--:(te-dev:named-srlgs)
   |  |     +--rw te-dev:named-srlgs* [named-srlg]
   {te-types:named-srlg-groups}?
   |  |        +--rw te-dev:named-srlg
   -> ../../../../../te:globals/config/named-srlgs/name
   |  +--rw te-dev:switching-capabilities*
   [switching-capability]
   |  |  +--rw te-dev:switching-capability    identityref
   |  |  +--rw te-dev:encoding?               identityref
   |  +--rw te-dev:thresholds
   |     +--rw (te-dev:type)?
   |        +--:(te-dev:equal-steps)
```

```
|       |  +--rw (te-dev:equal-step-type)?
|       |     +--:(te-dev:up-down-different-step)
|       |     |  +--rw te-dev:up-step?      uint8
|       |     |  +--rw te-dev:down-step?    uint8
|       |     +--:(te-dev:up-down-same-step)
|       |        +--rw te-dev:step?         uint8
|       +--:(te-dev:unequal-steps)
|          +--rw te-dev:up-steps* [value]
|          |  +--rw te-dev:value    uint8
|          +--rw te-dev:down-steps* [value]
|             +--rw te-dev:value    uint8
+--ro te-dev:state
   +--ro te-dev:te-metric? te-types:te-metric
   +--ro (te-dev:admin-group-type)?
   |  +--:(te-dev:value-admin-groups)
   |  |  +--ro (te-dev:value-admin-group-type)?
   |  |     +--:(te-dev:admin-groups)
   |  |     |  +--ro te-dev:admin-group?te-types:admin-group
   |  |     +--:(te-dev:extended-admin-groups)
   {te-types:extended-admin-groups}?
   |  |        +--ro te-dev:extended-admin-group?
   te-types:extended-admin-group
   |  +--:(te-dev:named-admin-groups)
   |     +--ro te-dev:named-admin-groups*[named-admin-group]
   {te-types:extended-admin-groups,
   te-types:named-extended-admin-groups}?
   |        +--ro te-dev:named-admin-group
   -> ../../../../../te:globals/config/named-admin-groups/na
   +--ro (te-dev:srlg-type)?
   |  +--:(te-dev:value-srlgs)
   |  |  +--ro te-dev:values* [value]
   |  |     +--ro te-dev:value    uint32
   |  +--:(te-dev:named-srlgs)
   |     +--ro te-dev:named-srlgs* [named-srlg]
   {te-types:named-srlg-groups}?
   |        +--ro te-dev:named-srlg
   -> ../../../../../te:globals/config/named-srlgs/name
   +--ro te-dev:switching-capabilities*[switching-capability
   |  +--ro te-dev:switching-capability    identityref
   |  +--ro te-dev:encoding?               identityref
   +--ro te-dev:thresholds
   |  +--ro (te-dev:type)?
   |     +--:(te-dev:equal-steps)
   |     |  +--ro (te-dev:equal-step-type)?
   |     |     +--:(te-dev:up-down-different-step)
   |     |     |  +--ro te-dev:up-step?      uint8
   |     |     |  +--ro te-dev:down-step?    uint8
   |     |     +--:(te-dev:up-down-same-step)
```

```
               |     |        +--ro te-dev:step?         uint8
               |     +--:(te-dev:unequal-steps)
               |        +--ro te-dev:up-steps* [value]
               |        |  +--ro te-dev:value    uint8
               |        +--ro te-dev:down-steps* [value]
               |           +--ro te-dev:value    uint8
               +--ro te-dev:te-advertisements_state
                  +--ro te-dev:flood-interval?           uint32
                  +--ro te-dev:last-flooded-time?        uint32
                  +--ro te-dev:next-flooded-time?        uint32
                  +--ro te-dev:last-flooded-trigger?     enumeration
                  +--ro te-dev:advertized-level-areas* [level-area]
                     +--ro te-dev:level-area    uint32
rpcs:
   +---x globals-rpc
   +---x interfaces-rpc
   +---x tunnels-rpc
notifications:
   +---n globals-notif
   +---n tunnels-notif
module: ietf-te-device
rpcs:
   +---x interfaces-rpc
notifications:
   +---n interfaces-notif
```

Figure 6: TE globals configuration and state tree

4. TE Generic and Helper YANG Modules

```
   <CODE BEGINS>file "ietf-te-types@2016-07-05.yang"
   module ietf-te-types {

     namespace "urn:ietf:params:xml:ns:yang:ietf-te-types";

     /* Replace with IANA when assigned */
     prefix "te-types";

     import ietf-inet-types {
       prefix inet;
     }

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

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

```
        Working Group";

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

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

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

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

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

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

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

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

         Editor:   Xia Chen
                   <mailto:jescia.chenxia@huawei.com>

         Editor:   Raqib Jones
                   <mailto:raqib@Brocade.com>

         Editor:   Bin Wen
                   <mailto:Bin_Wen@cable.comcast.com>";

      description
        "This module contains a collection of generally
        useful TE specific YANG data type defintions.";

      revision "2016-07-05" {
        description "Latest revision of TE basic types";
        reference "RFC3209";
      }

      /*
       * Identities
       */
```

```
identity tunnel-type {
  description
    "Base identity from which specific tunnel types are
    derived.";
}

identity tunnel-p2p {
  base tunnel-type;
  description
    "TE point-to-point tunnel type.";
}

identity tunnel-p2mp {
  base tunnel-type;
  description
    "TE point-to-multipoint tunnel type.";
}

identity state-type {
  description
    "Base identity for TE states";
}

identity state-up {
  base state-type;
  description
    "State up";
}

identity state-down {
  base state-type;
  description
    "State down";
}

identity path-invalidation-action-type {
  description
    "Base identity for TE path invalidation action types";
}

identity tunnel-invalidation-action-drop-type {
  base path-invalidation-action-type;
  description
    "TE path invalidation action drop";
}

identity tunnel-invalidation-action-drop-tear {
  base path-invalidation-action-type;
```

```
      description
        "TE path invalidation action tear";
    }

    identity lsp-prot-type {
      description
        "Base identity from which LSP protection types are
        derived.";
    }

    identity lsp-prot-unprotected {
      base lsp-prot-type;
      description
        "LSP protection 'Unprotected'";
      reference "RFC4872";
    }

    identity lsp-prot-reroute-extra {
      base lsp-prot-type;
      description
        "LSP protection '(Full) Rerouting'";
      reference "RFC4872";
    }

    identity lsp-prot-reroute {
      base lsp-prot-type;
      description
        "LSP protection 'Rerouting without Extra-Traffic'";
      reference "RFC4872";
    }

    identity lsp-prot-1-for-n {
      base lsp-prot-type;
      description
        "LSP protection '1:N Protection with Extra-Traffic'";
      reference "RFC4872";
    }

    identity lsp-prot-unidir-1-to-1 {
      base lsp-prot-type;
      description
        "LSP protection '1+1 Unidirectional Protection'";
      reference "RFC4872";
    }

    identity lsp-prot-bidir-1-to-1 {
      base lsp-prot-type;
      description
```

```
        "LSP protection '1+1 Bidirectional Protection'";
      reference "RFC4872";
    }

    identity switching-capabilities {
      description
        "Base identity for interface switching capabilities";
    }

    identity switching-psc1 {
      base switching-capabilities;
      description
        "Packet-Switch Capable-1 (PSC-1)";
    }

    identity switching-evpl {
      base switching-capabilities;
      description
        "Ethernet Virtual Private Line (EVPL)";
    }

    identity switching-l2sc {
      base switching-capabilities;
      description
        "Layer-2 Switch Capable (L2SC)";
    }

    identity switching-tdm {
      base switching-capabilities;
      description
        "Time-Division-Multiplex Capable (TDM)";
    }

    identity switching-otn {
      base switching-capabilities;
      description
        "OTN-TDM capable";
    }

    identity switching-dcsc {
      base switching-capabilities;
      description
        "Data Channel Switching Capable (DCSC)";
    }

    identity switching-lsc {
      base switching-capabilities;
      description
```

```
        "Lambda-Switch Capable (LSC)";
    }

    identity switching-fsc {
      base switching-capabilities;
      description
        "Fiber-Switch Capable (FSC)";
    }

    identity lsp-encoding-types {
      description
        "Base identity for encoding types";
    }

    identity lsp-encoding-packet {
      base lsp-encoding-types;
      description
        "Packet LSP encoding";
    }

    identity lsp-encoding-ethernet {
      base lsp-encoding-types;
      description
        "Ethernet LSP encoding";
    }

    identity lsp-encoding-pdh {
      base lsp-encoding-types;
      description
        "ANSI/ETSI LSP encoding";
    }

    identity lsp-encoding-sdh {
      base lsp-encoding-types;
      description
        "SDH ITU-T G.707 / SONET ANSI T1.105 LSP encoding";
    }

    identity lsp-encoding-digital-wrapper {
      base lsp-encoding-types;
      description
        "Digital Wrapper LSP encoding";
    }

    identity lsp-encoding-lambda {
      base lsp-encoding-types;
      description
        "Lambda (photonic) LSP encoding";
```

```
    }

    identity lsp-encoding-fiber {
      base lsp-encoding-types;
      description
        "Fiber LSP encoding";
    }

    identity lsp-encoding-fiber-channel {
      base lsp-encoding-types;
      description
        "FiberChannel LSP encoding";
    }

    identity lsp-encoding-oduk {
      base lsp-encoding-types;
      description
        "G.709 ODUk (Digital Path)LSP encoding";
    }

    identity lsp-encoding-optical-channel {
      base lsp-encoding-types;
      description
        "Line (e.g., 8B/10B) LSP encoding";
    }

    identity lsp-encoding-line {
      base lsp-encoding-types;
      description
        "Line (e.g., 8B/10B) LSP encoding";
    }

    identity path-signaling-type {
      description
        "Base identity from which specific path signaling
         types are derived.";
    }

    identity path-signaling-rsvpte {
      base tunnel-type;
      description
        "RSVP-TE path signaling type";
    }

    identity path-signaling-sr {
      base tunnel-type;
      description
        "Segment-routing path signaling type";
```

```
    }

    /* TE basic features */
    feature p2mp-te {
      description
        "Indicates support for P2MP-TE";
    }

    feature frr-te {
      description
        "Indicates support for TE FastReroute (FRR)";
    }

    feature extended-admin-groups {
      description
        "Indicates support for TE link extended admin
        groups.";
    }

    feature named-path-affinities {
      description
        "Indicates support for named path affinities";
    }

    feature named-extended-admin-groups {
      description
        "Indicates support for named extended admin groups";
    }

    feature named-srlg-groups {
      description
        "Indicates support for named SRLG groups";
    }

    feature named-path-constraints {
      description
        "Indicates support for named path constraints";
    }

    grouping explicit-route-subobject {
      description
        "The explicit route subobject grouping";
      choice type {
        description
          "The explicit route subobject type";
        case ipv4-address {
          description
            "IPv4 address explicit route subobject";
```

```
            leaf v4-address {
              type inet:ipv4-address;
              description
                "An IPv4 address.  This address is
                treated as a prefix based on the
                prefix length value below. Bits beyond
                the prefix are ignored on receipt and
                SHOULD be set to zero on transmission.";
            }
            leaf v4-prefix-length {
              type uint8;
              description
                "Length in bits of the IPv4 prefix";
            }
            leaf v4-loose {
              type boolean;
              description
                "Describes whether the object is loose
                if set, or otherwise strict";
            }
          }
          case ipv6-address {
            description
              "IPv6 address Explicit Route Object";
            leaf v6-address {
              type inet:ipv6-address;
              description
                "An IPv6 address.  This address is
                treated as a prefix based on the
                prefix length value below.  Bits
                beyond the prefix are ignored on
                receipt and SHOULD be set to zero
                on transmission.";
            }
            leaf v6-prefix-length {
              type uint8;
              description
                "Length in bits of the IPv4 prefix";
            }
            leaf v6-loose {
              type boolean;
              description
                "Describes whether the object is loose
                if set, or otherwise strict";
            }
          }
          case as-number {
            leaf as-number {
```

```
            type uint16;
            description "AS number";
          }
          description
            "Autonomous System explicit route subobject";
        }
        case unnumbered-link {
          leaf router-id {
            type inet:ip-address;
            description
              "A router-id address";
          }
          leaf interface-id {
            type uint32;
            description "The interface identifier";
          }
          description
            "Unnumbered link explicit route subobject";
          reference
            "RFC3477: Signalling Unnumbered Links in
            RSVP-TE";
        }
        case label {
          leaf value {
            type uint32;
            description "the label value";
          }
          description
            "The Label ERO subobject";
        }
        /* AS domain sequence..? */
      }
    }

    grouping record-route-subobject {
      description
        "The record route subobject grouping";
      choice type {
        description
          "The record route subobject type";
        case ipv4-address {
          leaf v4-address {
            type inet:ipv4-address;
            description
              "An IPv4 address.  This address is
              treated as a prefix based on the prefix
              length value below. Bits beyond the
              prefix are ignored on receipt and
```

```
              SHOULD be set to zero on transmission.";
          }
          leaf v4-prefix-length {
            type uint8;
            description
              "Length in bits of the IPv4 prefix";
          }
          leaf v4-flags {
            type uint8;
            description
              "IPv4 address sub-object flags";
            reference "RFC3209";
          }
        }
        case ipv6-address {
          leaf v6-address {
            type inet:ipv6-address;
            description
              "An IPv6 address.  This address is
              treated as a prefix based on the
              prefix length value below. Bits
              beyond the prefix are ignored on
              receipt and SHOULD be set to zero
              on transmission.";
          }
          leaf v6-prefix-length {
            type uint8;
            description
              "Length in bits of the IPv4 prefix";
          }
          leaf v6-flags {
            type uint8;
            description
              "IPv6 address sub-object flags";
            reference "RFC3209";
          }
        }
        case unnumbered-link {
          leaf router-id {
            type inet:ip-address;
            description
              "A router-id address";
          }
          leaf interface-id {
            type uint32;
            description "The interface identifier";
          }
          description
```

```
            "Unnumbered link record route subobject";
          reference
            "RFC3477: Signalling Unnumbered Links in
             RSVP-TE";
        }
        case label {
          leaf value {
            type uint32;
            description "the label value";
          }
          leaf flags {
            type uint8;
            description
              "Label sub-object flags";
            reference "RFC3209";
          }
          description
            "The Label ERO subobject";
        }
      }
    }

    identity route-usage-type {
      description
        "Base identity for route usage";
    }

    identity route-include-ero {
      base route-usage-type;
      description
        "Include ERO from route";
    }

    identity route-exclude-ero {
      base route-usage-type;
      description
        "Exclude ERO from route";
    }

    identity route-exclude-srlg {
      base route-usage-type;
      description
        "Exclude SRLG from route";
    }

    identity path-metric-type {
      description
        "Base identity for path metric type";
```

```
    }

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

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

    identity path-tiebreaker-type {
      description
        "Base identity for path tie-breaker type";
    }

    identity path-tiebreaker-minfill {
      base path-tiebreaker-type;
      description
        "Min-Fill LSP path placement";
    }

    identity path-tiebreaker-maxfill {
      base path-tiebreaker-type;
      description
        "Max-Fill LSP path placement";
    }

    identity path-tiebreaker-randoom {
      base path-tiebreaker-type;
      description
        "Random LSP path placement";
    }

    identity bidir-provisioning-mode {
      description
        "Base identity for bidirectional provisioning
        mode.";
    }

    identity bidir-provisioning-single-sided {
      base bidir-provisioning-mode;
      description
        "Single-sided bidirectional provioning mode";
    }
```

```
    identity bidir-provisioning-double-sided {
      base bidir-provisioning-mode;
      description
        "Double-sided bidirectional provioning mode";
    }

    identity bidir-association-type {
      description
        "Base identity for bidirectional association type";
    }

    identity bidir-assoc-corouted {
      base bidir-association-type;
      description
        "Co-routed bidirectional association type";
    }

    identity bidir-assoc-non-corouted {
      base bidir-association-type;
      description
        "Non co-routed bidirectional association type";
    }

    identity resource-affinities-type {
      description
        "Base identity for resource affinities";
    }

    identity resource-aff-include-all {
      base resource-affinities-type;
      description
        "The set of attribute filters associated with a
        tunnel all of which must be present for a link
        to be acceptable";
    }

    identity resource-aff-include-any {
      base resource-affinities-type;
      description
        "The set of attribute filters associated with a
        tunnel any of which must be present for a link
        to be acceptable";
    }

    identity resource-aff-exclude-any {
      base resource-affinities-type;
      description
        "The set of attribute filters associated with a
```

```
        tunnel any of which renders a link unacceptable";
    }

    identity te-optimization-criterion {
      description
        "Base identity for TE optimization criterion.";
      reference
        "RFC3272: Overview and Principles of Internet Traffic
         Engineering.";
    }

    identity not-optimized {
      base te-optimization-criterion;
      description "Optimization is not applied.";
    }

    identity cost {
      base te-optimization-criterion;
      description "Optimized on cost.";
    }

    identity delay {
      base te-optimization-criterion;
      description "Optimized on delay.";
    }

    /*
     * Typedefs
     */
    typedef performance-metric-normality {
      type enumeration {
        enum "unknown" {
          value 0;
          description
            "Unknown.";
        }
        enum "normal" {
          value 1;
          description
            "Normal.";
        }
        enum "abnormal" {
          value 2;
          description
            "Abnormal. The anomalous bit is set.";
        }
      }
      description
```

```
        "Indicates whether a performance metric is normal, abnormal, or
         unknown.";
      reference
        "RFC7471: OSPF Traffic Engineering (TE) Metric Extensions.
         RFC7810: IS-IS Traffic Engineering (TE) Metric Extensions.
         RFC7823: Performance-Based Path Selection for Explicitly
         Routed Label Switched Paths (LSPs) Using TE Metric
         Extensions";
    }

    typedef te-admin-status {
      type enumeration {
        enum up {
          description
            "Enabled.";
        }
        enum down {
          description
            "Disabled.";
        }
        enum testing {
          description
            "In some test mode.";
        }
        enum preparing-maintenance {
          description
            "Resource is disabled in the control plane to prepare for
             graceful shutdown for maintenance purposes.";
          reference
            "RFC5817: Graceful Shutdown in MPLS and Generalized MPLS
             Traffic Engineering Networks";
        }
        enum maintenance {
          description
            "Resource is disabled in the data plane for maintenance
             purposes.";
        }
      }
      description
        "Defines a type representing the administrative status of
         a TE resource.";
    }

    typedef te-global-id {
      type uint32;
      description
        "An identifier to uniquely identify an operator, which can be
         either a provider or a client.
```

```
          The definition of this type is taken from RFC6370 and RFC5003.
          This attribute type is used solely to provide a globally
          unique context for TE topologies.";
      }

      typedef te-link-access-type {
        type enumeration {
          enum point-to-point {
            description
              "The link is point-to-point.";
          }
          enum multi-access {
            description
              "The link is multi-access, including broacast and NBMA.";
          }
        }
        description
          "Defines a type representing the access type of a TE link.";
        reference
          "RFC3630: Traffic Engineering (TE) Extensions to OSPF
           Version 2.";
      }

      typedef te-node-id {
        type yang:dotted-quad;
        description
          "An identifier for a node in a topology.
           The identifier is represented as 32-bit unsigned integer in
           the dotted-quad notation.
           This attribute is mapped to Router ID in
           RFC3630, RFC5329, RFC5305, and RFC6119.";
      }

      typedef te-oper-status {
        type enumeration {
          enum up {
            description
            "Operational up.";
          }
          enum down {
            description
            "Operational down.";
          }
          enum testing {
            description
            "In some test mode.";
          }
          enum unknown {
```

```
          description
          "Status cannot be determined for some reason.";
        }
        enum preparing-maintenance {
          description
            "Resource is disabled in the control plane to prepare for
             graceful shutdown for maintenance purposes.";
          reference
            "RFC5817: Graceful Shutdown in MPLS and Generalized MPLS
             Traffic Engineering Networks";
        }
        enum maintenance {
          description
            "Resource is disabled in the data plane for maintenance
             purposes.";
        }
      }
      description
        "Defines a type representing the operational status of
         a TE resource.";
    }

    typedef te-recovery-status {
      type enumeration {
        enum normal {
          description
            "Both the recovery and working spans are fully
             allocated and active, data traffic is being
             transported over (or selected from) the working
             span, and no trigger events are reported.";
        }
        enum recovery-started {
          description
            "The recovery action has been started, but not completed.";
        }
        enum recovery-succeeded {
          description
            "The recovery action has succeeded. The working span has
             reported a failure/degrade condition and the user traffic
             is being transported (or selected) on the recovery span.";
        }
        enum recovery-failed {
          description
            "The recovery action has failed.";
        }
        enum reversion-started {
          description
            "The reversion has started.";
```

```
        }
        enum reversion-failed {
          description
            "The reversion has failed.";
        }
        enum recovery-unavailable {
          description
            "The recovery is unavailable -- either as a result of an
             operator Lockout command or a failure condition detected
             on the recovery span.";
        }
        enum recovery-admin {
          description
            "The operator has issued a command switching the user
             traffic to the recovery span.";
        }
        enum wait-to-restore {
          description
            "The recovery domain is recovering from a failuer/degrade
             condition on the working span that is being controlled by
             the Wait-to-Restore (WTR) timer.";
        }
      }
      description
        "Defines the status of a recovery action.";
      reference
        "RFC4427: Recovery (Protection and Restoration) Terminology
         for Generalized Multi-Protocol Label Switching (GMPLS).
         RFC6378: MPLS Transport Profile (MPLS-TP) Linear Protection";
    }

    typedef te-template-name {
      type string {
        pattern '/?([a-zA-Z0-9\-_.]+)(/[a-zA-Z0-9\-_.]+)*';
      }
      description
        "A type for the name of a TE node template or TE link
         template.";
    }

    typedef te-topology-event-type {
      type enumeration {
        enum "add" {
          value 0;
          description
            "A TE node or te-link has been added.";
        }
        enum "remove" {
```

```
          value 1;
          description
            "A TE node or te-link has been removed.";
        }
        enum "update" {
          value 2;
          description
            "A TE node or te-link has been updated.";
        }
      }
      description "TE  Event type for notifications";
    } // te-topology-event-type

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

    typedef te-tp-id {
      type union {
        type uint32;          // Unnumbered
        type inet:ip-address; // IPv4 or IPv6 address
      }
      description
        "An identifier for a TE link endpoint on a node.
         This attribute is mapped to local or remote link identifier in
         RFC3630 and RFC5305.";
    }

    typedef generalized-label {
      type binary;
      description
        "Generalized label. Nodes sending and receiving the
         Generalized Label know what kinds of link they are
         using, the Generalized Label does not identify its
         type.  Instead, nodes are expected to know from the
         context what type of label to expect.";
        reference "rfc3471: section 3.2";
    }

    typedef admin-group {
      type binary {
        length 32;
      }
      description
```

```
        "Administrative group/Resource class/Color.";
    }

    typedef extended-admin-group {
      type binary;
      description
        "Extended administrative group/Resource class/Color.";
    }

    typedef admin-groups {
      type union {
        type admin-group;
        type extended-admin-group;
      }
      description "TE administrative group derived type";
    }

    typedef srlg {
      type uint32;
      description "SRLG type";
    }

    identity path-computation-srlg-type {
      description
        "Base identity for SRLG path computation";
    }

    identity srlg-ignore {
      base path-computation-srlg-type;
      description
        "Ignores SRLGs in path computation";
    }

    identity srlg-strict {
      base path-computation-srlg-type;
      description
        "Include strict SRLG check in path computation";
    }

    identity srlg-preferred {
      base path-computation-srlg-type;
      description
        "Include preferred SRLG check in path computation";
    }

    identity srlg-weighted {
      base path-computation-srlg-type;
      description
```

```
        "Include weighted SRLG check in path computation";
    }

    typedef te-metric {
      type uint32;
      description
        "TE link metric";
    }

    /**
     * TE tunnel generic groupings
     **/

    /* Tunnel path selection parameters */
    grouping tunnel-path-selection {
      description
        "Tunnel path selection properties grouping";
      container path-selection {
        description
          "Tunnel path selection properties container";
        leaf topology-id {
          type te-types:te-topology-id;
          description
            "The tunnel path is computed using the specific
            topology identified by this identifier";
        }
        leaf cost-limit {
          type uint32 {
            range "1..4294967295";
          }
          description
            "The tunnel path cost limit.";
        }
        leaf hop-limit {
          type uint8 {
            range "1..255";
          }
          description
            "The tunnel path hop limit.";
        }
        leaf metric-type {
          type identityref {
            base path-metric-type;
          }
          default path-metric-te;
          description
            "The tunnel path metric type.";
        }
```

```
        leaf tiebreaker-type {
          type identityref {
            base path-tiebreaker-type;
          }
          default path-tiebreaker-maxfill;
          description
            "The tunnel path computation tie breakers.";
        }
        leaf ignore-overload {
          type boolean;
          description
            "The tunnel path can traverse overloaded node.";
        }
        uses tunnel-path-affinities;
        uses tunnel-path-srlgs;
      }
    }

    grouping tunnel-path-affinities {
      description
        "Path affinities grouping";
      container tunnel-path-affinities {
        if-feature named-path-affinities;
        description
          "Path affinities container";
        choice style {
          description
            "Path affinities representation style";
          case values {
            leaf value {
              type uint32 {
                range "0..4294967295";
              }
              description
                "Affinity value";
            }
            leaf mask {
              type uint32 {
                range "0..4294967295";
              }
              description
                "Affinity mask";
            }
          }
          case named {
            list constraints {
              key "usage";
              leaf usage {
```

```
                type identityref {
                  base resource-affinities-type;
                }
                description "Affinities usage";
              }
              container constraint {
                description
                  "Container for named affinities";
                list affinity-names {
                  key "name";
                  leaf name {
                    type string;
                    description
                      "Affinity name";
                  }
                  description
                    "List of named affinities";
                }
              }
              description
                "List of named affinity constraints";
            }
          }
        }
      }
    }

    grouping tunnel-path-srlgs {
      description
        "Path SRLG properties grouping";
      container tunnel-path-srlgs {
        description
          "Path SRLG properties container";
        choice style {
          description
            "Type of SRLG representation";
          case values {
            leaf usage {
              type identityref {
                base route-exclude-srlg;
              }
              description "SRLG usage";
            }
            leaf-list values {
              type te-types:srlg;
              description "SRLG value";
            }
          }
```

```
          case named {
            list constraints {
              key "usage";
              leaf usage {
                type identityref {
                  base route-exclude-srlg;
                }
                description "SRLG usage";
              }
              container constraint {
                description
                  "Container for named SRLG list";
                list srlg-names {
                  key "name";
                  leaf name {
                    type string;
                    description
                      "The SRLG name";
                  }
                  description
                    "List named SRLGs";
                }
              }
              description
                "List of named SRLG constraints";
            }
          }
        }
      }
    }

    grouping tunnel-bidir-assoc-properties {
      description
        "TE tunnel associated bidirectional properties
        grouping";
      container bidirectional {
        description
          "TE tunnel associated bidirectional attributes.";
        container association {
          description
            "Tunnel bidirectional association properties";
          leaf id {
            type uint16;
            description
              "The TE tunnel association identifier.";
          }
          leaf source {
            type inet:ip-address;
```

```
            description
              "The TE tunnel association source.";
          }
          leaf global-source {
            type inet:ip-address;
            description
              "The TE tunnel association global
              source.";
          }
          leaf type {
            type identityref {
              base bidir-association-type;
            }
            default bidir-assoc-non-corouted;
            description
              "The TE tunnel association type.";
          }
          leaf provisioing {
            type identityref {
              base bidir-provisioning-mode;
            }
            description
              "Describes the provisioning model of the
              associated bidirectional LSP";
            reference
              "draft-ietf-teas-mpls-tp-rsvpte-ext-
              associated-lsp, section-3.2";
          }
        }
      }
    }
    /*** End of TE tunnel groupings ***/

    /**
     * TE interface generic groupings
     **/
  }
  <CODE ENDS>
```

Figure 7: TE basic types YANG module

```
<CODE BEGINS> file "ietf-te@2016-07-05.yang"
module ietf-te {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te";

  /* Replace with IANA when assigned */
  prefix "te";
```

```
  /* Import TE generic types */
  import ietf-te-types {
    prefix te-types;
  }

  import ietf-inet-types {
    prefix inet;
  }

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

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

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

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

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

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

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

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

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

     Editor:   Xia Chen
               <mailto:jescia.chenxia@huawei.com>

     Editor:   Raqib Jones
               <mailto:raqib@Brocade.com>

     Editor:   Bin Wen
               <mailto:Bin_Wen@cable.comcast.com>";

  description
```

```
    "YANG data module for TE configuration,
    state, RPC and notifications.";

  revision "2016-07-05" {
    description "Latest update to TE generic YANG module.";
    reference "TBD";
  }

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

  /**
   * TE tunnel generic groupings
   */

  grouping p2p-secondary-path-params {
    description
      "tunnel path properties.";
    container config {
      description
        "Configuration parameters relating to
        tunnel properties";
      uses path-properties_config;
      uses path-params_config;
    }
    container state {
      config false;
      description
        "State information associated with tunnel
        properties";
      uses path-properties_config;
      uses path-params_config;
      uses p2p-secondary-path-params_state;
    }
  }

  grouping p2p-primary-path-params {
    description
      "TE tunnel primary path properties grouping";
    container config {
      description
        "Configuration parameters relating to
```

```
        tunnel properties";
      uses path-properties_config;
      uses path-params_config;
    }
    container state {
      config false;
      description
        "State information associated with tunnel
        properties";
      uses path-params_config;
      uses p2p-primary-path-params_state;
    }
  }

  grouping p2p-primary-path-params_state {
    description "TE primary path state parameters";
    list lsp {
      key
        "source destination tunnel-id lsp-id "+
        "extended-tunnel-id type";
      description "List of LSPs associated with the tunnel.";

      leaf source {
        type leafref {
          path "../../../../../../lsps-state/lsp/source";
        }
        description
          "Tunnel sender address extracted from
          SENDER_TEMPLATE  object";
        reference "RFC3209";
      }
      leaf destination {
        type leafref {
          path "../../../../../../lsps-state/lsp/destination";
        }
        description
          "Tunnel endpoint address extracted from
          SESSION object";
        reference "RFC3209";
      }
      leaf tunnel-id {
        type leafref {
          path "../../../../../../lsps-state/lsp/tunnel-id";
        }
        description
          "Tunnel identifier used in the SESSION
          that remains constant over the life
          of the tunnel.";
```

```
        reference "RFC3209";
      }
      leaf lsp-id {
        type leafref {
          path "../../../../../../lsps-state/lsp/lsp-id";
        }
        description
          "Identifier used in the SENDER_TEMPLATE
          and the FILTER_SPEC that can be changed
          to allow a sender to share resources with
          itself.";
        reference "RFC3209";
      }
      leaf extended-tunnel-id {
        type leafref {
          path "../../../../../../lsps-state/lsp/extended-tunnel-id";
        }
        description
          "Extended Tunnel ID of the LSP.";
        reference "RFC3209";
      }
      leaf type {
        type leafref {
          path "../../../../../../lsps-state/lsp/type";
        }
        description "LSP type P2P or P2MP";
      }
    }
  }

  grouping p2p-secondary-path-params_state {
    description "TE secondary path state parameters";
    list lsp {
      key "source";
      description "List of LSPs associated with the tunnel.";

      leaf source {
        type leafref {
          path "../../../../../../../lsps-state/lsp/source";
        }
        description
          "Tunnel sender address extracted from
          SENDER_TEMPLATE  object";
        reference "RFC3209";
      }
      leaf destination {
        type leafref {
          path "../../../../../../../lsps-state/lsp/destination";
```

```
        }
        description
          "Tunnel endpoint address extracted from
          SESSION object";
        reference "RFC3209";
      }
      leaf tunnel-id {
        type leafref {
          path "../../../../../../../lsps-state/lsp/tunnel-id";
        }
        description
          "Tunnel identifier used in the SESSION
          that remains constant over the life
          of the tunnel.";
        reference "RFC3209";
      }
      leaf lsp-id {
        type leafref {
          path "../../../../../../../lsps-state/lsp/lsp-id";
        }
        description
          "Identifier used in the SENDER_TEMPLATE
          and the FILTER_SPEC that can be changed
          to allow a sender to share resources with
          itself.";
        reference "RFC3209";
      }
      leaf extended-tunnel-id {
        type leafref {
          path "../../../../../../../lsps-state/lsp/extended-tunnel-id";
        }
        description
          "Extended Tunnel ID of the LSP.";
        reference "RFC3209";
      }
      leaf type {
        type leafref {
          path "../../../../../../../lsps-state/lsp/type";
        }
        description "LSP type P2P or P2MP";
      }
    }
  }

  grouping path-params_config {
    description
      "TE tunnel path parameters configuration grouping";
    leaf path-named-constraint {
```

```
      if-feature te-types:named-path-constraints;
      type string;
      description
        "Reference to a globally defined named path
        constraint set";
    }
    uses te-types:tunnel-path-selection;
    choice type {
      description
        "Describes the path type";
      case dynamic {
        leaf dynamic {
          type empty;
          description
            "A CSPF dynamically computed path";
        }
      }
      case explicit {
        leaf explicit-path-name {
          type string;
          description
            "The explicit-path name";
        }

        list explicit-route-objects {
          key "index";
          description
            "List of explicit route objects";
          leaf index {
            type uint8 {
              range "0..255";
            }
            description
              "Index of this explicit route object";
          }
          leaf explicit-route-usage {
            type identityref {
              base te-types:route-usage-type;
            }
            description "An explicit-route hop action.";
          }
          uses te-types:explicit-route-subobject;
        }
      }
    }
    leaf no-cspf {
      type empty;
      description
```

```
        "Indicates no CSPF is to be attempted on this
        path.";
    }
    leaf lockdown {
      type empty;
      description
        "Indicates no reoptimization to be attempted for
        this path.";
    }
  }

  /* TE tunnel configuration data */
  grouping tunnel-params_config {
    description
      "Configuration parameters relating to TE tunnel";
    leaf name {
      type string;
      description "TE tunnel name.";
    }
    leaf type {
      type identityref {
        base te-types:tunnel-type;
      }
      description "TE tunnel type.";
    }
    leaf identifier {
      type uint16;
      description
        "TE tunnel Identifier.";
    }
    leaf description {
      type string;
      description
        "Textual description for this TE tunnel";
    }
    leaf lsp-priority-setup {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP setup priority";
    }
    leaf lsp-priority-hold {
      type uint8 {
        range "0..7";
      }
      description
        "TE LSP hold priority";
```

```
    }
    leaf lsp-protection-type {
      type identityref {
        base te-types:lsp-prot-type;
      }
      description "LSP protection type.";
    }
    leaf admin-status {
      type identityref {
        base te-types:state-type;
      }
      default te-types:state-up;
      description "TE tunnel administrative state.";
    }
    leaf source {
      type inet:ip-address;
      description
        "TE tunnel source address.";
    }
    leaf destination {
      /* Add when check */
      type inet:ip-address;
      description
        "P2P tunnel destination address";
    }
    leaf src-tp-id {
      type binary;
      description
        "TE tunnel source termination point identifier.";
    }
    leaf dst-tp-id {
      /* Add when check */
      type binary;
      description
        "TE tunnel destination termination point identifier.";
    }
    container hierarchical-link-id {
        description
          "Identifies a hierarchical link (in server layer)
           that this tunnel is associated with.";
        leaf local-te-node-id {
          type te-types:te-node-id;
          description
            "Local TE node identifier";
        }
        leaf local-te-link-tp-id {
          type te-types:te-tp-id;
          description
```

```
            "Local TE link termination point identifier";
        }
        leaf remote-te-node-id {
          type te-types:te-node-id;
          description
            "Remote TE node identifier";
        }
        leaf te-topology-id {
          type te-types:te-topology-id;
          description
            "It is presumed that a datastore will contain many
             topologies. To distinguish between topologies it is
             vital to have UNIQUE topology identifiers.";
        }
    }
    uses te-types:tunnel-bidir-assoc-properties;
  }

  grouping tunnel-params_state {
    description
      "State parameters relating to TE tunnel";
    leaf oper-status {
      type identityref {
        base te-types:state-type;
      }
      description "TE tunnel operational state.";
    }
  }

  grouping path-properties_config {
    description "TE path properties grouping";
    leaf name {
      type string;
      description "TE path name";
    }
    leaf preference {
      type uint8 {
        range "1..255";
      }
      description
        "Specifies a preference for this path. The lower the
        number higher the preference";
    }
  }

  /* TE tunnel configuration/state grouping */
  grouping tunnel-properties {
    description
```

```
      "Top level grouping for tunnel properties.";
    container config {
      description
        "Configuration parameters relating to
         tunnel properties";
      uses tunnel-params_config;
    }
    container state {
      config false;
      description
        "State information associated with tunnel
         properties";
      uses tunnel-params_config;
      uses tunnel-params_state;
    }
    list primary-paths {
      key "name";
      description
        "List of primary paths for this tunnel.";
      leaf name {
        type leafref {
          path "../config/name";
        }
        description "TE path name";
      }
      leaf preference {
        type leafref {
          path "../config/preference";
        }
        description
          "Specifies a preference for this path. The lower the
           number higher the preference";
      }
      uses p2p-primary-path-params;
      list secondary-paths {
        key "name";
        description
          "List of secondary paths for this tunnel.";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "TE path name";
        }
        leaf preference {
          type leafref {
            path "../config/preference";
          }
```

```
          description
            "Specifies a preference for this path. The lower the
            number higher the preference";
        }
        uses p2p-secondary-path-params;
      }
    }
  }
  /*** End of TE tunnel groupings ***/

  /**
   * LSP related generic groupings
   */
  grouping lsp-record-route-information_state {
    description "recorded route information grouping";
    container lsp-record-route {
      description "RSVP recorded route object information";
      list record-route-subobjects {
        when "../origin-type != 'ingress'" {
          description "Applicable on non-ingress LSPs only";
        }
        key "subobject-index";
        description "";
        leaf subobject-index {
          type uint32;
          description "RRO subobject index";
        }
        uses te-types:record-route-subobject;
      }
    }
  }

  grouping lsp-properties_state {
    description
      "State parameters relating to LSP";
    leaf oper-status {
      type identityref {
        base te-types:state-type;
      }
      description "LSP operational state.";
    }

    leaf origin-type {
      type enumeration {
        enum ingress {
          description
            "Origin ingress";
        }
```

```
        enum egress {
          description
            "Origin egress";
        }
        enum transit {
          description
            "transit";
        }
      }
      description
        "Origin type of LSP relative to the location
        of the local switch in the path.";
    }

    leaf lsp-resource-status {
      type enumeration {
        enum primary {
          description
            "A primary LSP is a fully established LSP for
             which the resource allocation has been committed
             at the data plane";
        }
        enum secondary {
          description
            "A secondary LSP is an LSP that has been provisioned
             in the control plane only; e.g. resource allocation
             has not been committed at the data plane";
        }
      }
      description "LSP resource allocation type";
      reference "rfc4872, section 4.2.1";
    }

    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-operational-status {
      type empty;
      description
        "This bit is set when a protecting LSP is carrying the normal
         traffic after protection switching";
    }
  }
  /*** End of TE LSP groupings ***/

  /**
   * TE global generic groupings
   */

  /* Global named admin-groups configuration data */
  grouping named-admin-groups_config {
    description
      "Global named administrative groups configuration
      grouping";
    list named-admin-groups {
      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";
      }
    }
  }

  /* Global named admin-srlgs configuration data */
  grouping named-srlgs_config {
    description
      "Global named SRLGs configuration
      grouping";
```

```
    list named-srlgs {
      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 group {
        type te-types:srlg;
        description "An SRLG value";
      }
    }
  }

  /* Global named explicit-paths configuration data */
  grouping named-explicit-paths_config {
    description
      "Global explicit path configuration
      grouping";
    list named-explicit-paths {
      key "name";
      description
        "A list of explicit paths";
      leaf name {
        type string;
        description
          "A string name that uniquely identifies an
          explicit path";
      }
      list explicit-route-objects {
        key "index";
        description
          "List of explicit route objects";
        leaf index {
          type uint8 {
            range "0..255";
          }
          description
            "Index of this explicit route object";
        }
        leaf explicit-route-usage {
          type identityref {
            base te-types:route-usage-type;
          }
```

```
          description "An explicit-route hop action.";
        }
        uses te-types:explicit-route-subobject;
      }
    }
  }

  /* Global named paths constraints configuration data */
  grouping named-path-constraints_config {
    description
      "Global named path constraints configuration
      grouping";
    list named-constraints {
      if-feature te-types:named-path-constraints;
      key "name";
      description
        "A list of named path constraints";
      leaf name {
        type string;
        description
          "A string name that uniquely identifies a
          path constraint set";
      }
      uses te-types:tunnel-path-selection;
    }
  }

  /* TE globals container data */
  grouping globals-grouping {
    description
      "Globals TE system-wide configuration data grouping";
    container globals {
      description
        "Globals TE system-wide configuration data container";
      container config {
        description
          "Configuration parameters for system-wide
           parameters";
        uses named-admin-groups_config;
        uses named-srlgs_config;
        uses named-explicit-paths_config;
        uses named-path-constraints_config;
      }
      container state {
        config false;
        description
          "State for system-wide parameters";
        uses named-admin-groups_config;
```

```
        uses named-srlgs_config;
        uses named-explicit-paths_config;
        uses named-path-constraints_config;
      }
    }
  }

  /* TE tunnels container data */
  grouping tunnels-grouping {
    description
      "Tunnels TE configuration data grouping";
    container tunnels {
      description
        "Tunnels TE configuration data container";

      list tunnel {
        key "name type";
        unique "identifier";
        description "TE tunnel.";
        leaf name {
          type leafref {
            path "../config/name";
          }
          description "TE tunnel name.";
        }
        leaf type {
          type leafref {
            path "../config/type";
          }
          description "TE tunnel type.";
        }
        leaf identifier {
          type leafref {
            path "../config/identifier";
          }
          description
            "TE tunnel Identifier.";
        }
        uses tunnel-properties;
      }
    }
  }

  /* TE LSPs ephemeral state container data */
  grouping lsps-state-grouping {
    description
      "LSPs state operational data grouping";
    container lsps-state {
```

```
      config "false";
      description "LSPs operational state data.";

      list lsp {
        key
          "source destination tunnel-id lsp-id "+
          "extended-tunnel-id type";
        description
          "List of LSPs associated with the tunnel.";
        leaf source {
          type inet:ip-address;
          description
            "Tunnel sender address extracted from
            SENDER_TEMPLATE  object";
          reference "RFC3209";
        }
        leaf destination {
          type inet:ip-address;
          description
            "Tunnel endpoint address extracted from
            SESSION object";
          reference "RFC3209";
        }
        leaf tunnel-id {
          type uint16;
          description
            "Tunnel identifier used in the SESSION
            that remains constant over the life
            of the tunnel.";
          reference "RFC3209";
        }
        leaf lsp-id {
          type uint16;
          description
            "Identifier used in the SENDER_TEMPLATE
            and the FILTER_SPEC that can be changed
            to allow a sender to share resources with
            itself.";
          reference "RFC3209";
        }
        leaf extended-tunnel-id {
          type inet:ip-address;
           description
            "Extended Tunnel ID of the LSP.";
          reference "RFC3209";
        }
        leaf type {
          type identityref {
```

```
            base te-types:tunnel-type;
          }
          description "The LSP type P2P or P2MP";
        }
        uses lsp-properties_state;
        uses lsp-record-route-information_state;
      }
    }
  }
  /*** End of TE global groupings ***/

  /**
   * TE configurations container
   */
  container te {
    presence "Enable TE feature.";
    description
       "TE global container.";

    /* TE Global Configuration Data */
    uses globals-grouping;

    /* TE Tunnel Configuration Data */
    uses tunnels-grouping;

    /* TE LSPs State Data */
    uses lsps-state-grouping;
  }

  /* TE Global RPCs/execution Data */
  rpc globals-rpc {
    description
      "Execution data for TE global.";
  }

  /* TE interfaces RPCs/execution Data */
  rpc interfaces-rpc {
    description
      "Execution data for TE interfaces.";
  }

  /* TE Tunnel RPCs/execution Data */
  rpc tunnels-rpc {
    description
      "TE tunnels RPC nodes";
  }

  /* TE Global Notification Data */
```

```
  notification globals-notif {
    description
      "Notification messages for Global TE.";
  }

  /* TE Tunnel Notification Data */
  notification tunnels-notif {
    description
      "Notification messages for TE tunnels.";
  }
}
<CODE ENDS>
```

Figure 8: TE generic YANG module

```
   <CODE BEGINS> file "ietf-te-device@2016-07-05.yang"
   module ietf-te-device {

     namespace "urn:ietf:params:xml:ns:yang:ietf-te-device";

     /* Replace with IANA when assigned */
     prefix "te-dev";

     /* Import TE generic types */
     import ietf-te {
       prefix te;
     }

     /* Import TE generic types */
     import ietf-te-types {
       prefix te-types;
     }

     import ietf-interfaces {
       prefix if;
     }

     import ietf-inet-types {
       prefix inet;
     }

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

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

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

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

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

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

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

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

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

        Editor:   Xia Chen
                  <mailto:jescia.chenxia@huawei.com>

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

    description
      "YANG data module for TE device configurations,
      state, RPC and notifications.";

    revision "2016-07-05" {
      description "Latest update to TE device YANG module.";
      reference "TBD";
    }


    /**
     * TE LSP device state grouping
     */
    grouping lsps-device_state {
      description "TE LSP device state grouping";
      container lsp-timers {
        when "../origin-type = 'ingress'" {
          description "Applicable to ingress LSPs only";
```

```
        }
        description "Ingress LSP timers";
        leaf life-time {
          type uint32;
          units seconds;
          description
            "lsp life time";
        }

        leaf time-to-install {
          type uint32;
          units seconds;
          description
            "lsp installation delay time";
        }

        leaf time-to-die {
          type uint32;
          units seconds;
          description
            "lsp expire delay time";
        }
      }

      container downstream-info {
        when "../../origin-type != 'egress'" {
          description "Applicable to ingress LSPs only";
        }
        description
          "downstream information";

        leaf nhop {
          type inet:ip-address;
          description
            "downstream nexthop.";
        }

        leaf outgoing-interface {
          type if:interface-ref;
          description
            "downstream interface.";
        }

        leaf neighbor {
          type inet:ip-address;
          description
            "downstream neighbor.";
        }
```

```
        leaf label {
          type uint32;
          description
            "downstream label.";
        }
      }

      container upstream-info {
        when "../../origin-type != 'ingress'" {
          description "Applicable to non-ingress LSPs only";
        }
        description
          "upstream information";

        leaf phop {
          type inet:ip-address;
          description
            "upstream nexthop or previous-hop.";
        }

        leaf neighbor {
          type inet:ip-address;
          description
            "upstream neighbor.";
        }

        leaf label {
          type uint32;
          description
            "upstream label.";
        }
      }
    }

    /**
     * Device general groupings.
     */
    grouping tunnel-device_config {
      description "Device TE tunnel configs";
      leaf path-invalidation-action {
        type identityref {
          base te-types:path-invalidation-action-type;
        }
        description "Tunnel path invalidtion action";
      }
    }

    grouping lsp-device-timers_config {
```

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

    /**
     * TE global device generic groupings
     */

    /* Global device specific configuration data */
    grouping globals-device_config {
      description "Device globals configs";
      uses lsp-device-timers_config;
    }

    /* Global device specific state data */
    grouping globals-device_state {
      description
        "Top level grouping for global state data.";
      leaf tunnels-counter {
        type uint32;
        description "Tunnels count";
      }
      leaf lsps-counter {
        type uint32;
        description "Tunnels count";
      }
    }

    /* TE interface container data */
    grouping interfaces-grouping {
      description
```

```
        "Interface TE configuration data grouping";
      container interfaces {
        description
          "Configuration data model for TE interfaces.";
        uses te-all-attributes;
        list interface {
          key "interface";
          description "TE interfaces.";
          leaf interface {
            type if:interface-ref;
            description
              "TE interface name.";
          }
          /* TE interface parameters */
          uses te-attributes;
        }
      }
    }

    /**
     * TE interface device generic groupings
     */
    grouping te-admin-groups_config {
      description
        "TE interface affinities grouping";
      choice admin-group-type {
        description
          "TE interface administrative groups
          representation type";
        case value-admin-groups {
          choice value-admin-group-type {
            description "choice of admin-groups";
            case admin-groups {
              description
                "Administrative group/Resource
                class/Color.";
              leaf admin-group {
                type te-types:admin-group;
                description
                  "TE interface administrative group";
              }
            }
            case extended-admin-groups {
              if-feature te-types:extended-admin-groups;
              description
                "Extended administrative group/Resource
                class/Color.";
              leaf extended-admin-group {
```

```
                type te-types:extended-admin-group;
                description
                  "TE interface extended administrativei
                  group";
              }
            }
          }
        }
        case named-admin-groups {
          list named-admin-groups {
            if-feature te-types:extended-admin-groups;
            if-feature te-types:named-extended-admin-groups;
            key named-admin-group;
            description
              "A list of named admin-group entries";
            leaf named-admin-group {
              type leafref {
                path "../../../../../te:globals/te:config/" +
                  "te:named-admin-groups/te:name";
              }
              description "A named admin-group entry";
            }
          }
        }
      }
    }

    /* TE interface SRLGs */
    grouping te-srlgs_config {
      description "TE interface SRLG grouping";
      choice srlg-type {
        description "Choice of SRLG configuration";
        case value-srlgs {
          list values {
            key "value";
            description "List of SRLG values that
              this link is part of.";
            leaf value {
              type uint32 {
                range "0..4294967295";
              }
              description
                "Value of the SRLG";
            }
          }
        }
        case named-srlgs {
          list named-srlgs {
```

```
            if-feature te-types:named-srlg-groups;
            key named-srlg;
            description
              "A list of named SRLG entries";
            leaf named-srlg {
              type leafref {
                path "../../../../../te:globals/te:config/" +
                  "te:named-srlgs/te:name";
              }
              description
                "A named SRLG entry";
            }
          }
        }
      }
    }

    /* TE interface flooding parameters */
    grouping te-flooding-parameters_config {
      description "Interface TE flooding properties.";
      container thresholds {
        description "Flooding threshold values in percentages.";
        choice type {
          description
            "Describes the flooding threshold step method";
          case equal-steps {
            choice equal-step-type {
              description
                "Describes whether up and down equal step
                size are same or different";
              case up-down-different-step {
                leaf up-step {
                  type uint8 {
                    range "0..100";
                  }
                  description
                    "Set single percentage threshold
                    for increasing resource
                      allocation";
                }
                leaf down-step {
                  type uint8 {
                    range "0..100";
                  }
                  description
                    "Set single percentage threshold
                    for decreasing resource
                      allocation";
```

```
                }
              }
              case up-down-same-step {
                leaf step {
                  type uint8 {
                    range "0..100";
                  }
                  description
                    "Set single percentage threshold
                    for increasing and decreasing
                      resource allocation";
                }
              }
            }
          }
          case unequal-steps {
            list up-steps {
              key "value";
              description
                "Set nultuple percentage thresholds for
                increasing resource allocation";
              leaf value {
                type uint8 {
                  range "0..100";
                }
                description
                  "Percentage value";
              }
            }
            list down-steps {
              key "value";
              description
                "Set nultuple percentage thresholds for
                decreasing resource allocation";
              leaf value {
                type uint8 {
                  range "0..100";
                }
                description
                  "Percentage value";
              }
            }
          }
        }
      }
    }

    /* TE interface metric */
```

```
    grouping te-metric_config {
      description "Interface TE metric grouping";
      leaf te-metric {
        type te-types:te-metric;
        description "Interface TE metric.";
      }
    }

    /* TE interface switching capabilities */
    grouping te-switching-cap_config {
      description
        "TE interface switching capabilities";
      list switching-capabilities {
        key "switching-capability";
        description
          "List of interface capabilities for this interface";
        leaf switching-capability {
          type identityref {
            base te-types:switching-capabilities;
          }
          description
            "Switching Capability for this interface";
        }
        leaf encoding {
          type identityref {
            base te-types:lsp-encoding-types;
          }
          description
            "Encoding supported by this interface";
        }
      }
    }

    grouping te-advertisements_state {
      description
        "TE interface advertisements state grouping";
      container te-advertisements_state {
        description
          "TE interface advertisements state container";
        leaf flood-interval {
          type uint32;
          description
            "The periodic flooding interval";
        }
        leaf last-flooded-time {
          type uint32;
          units seconds;
          description
```

```
            "Time elapsed since last flooding in seconds";
          }
          leaf next-flooded-time {
            type uint32;
            units seconds;
            description
              "Time remained for next flooding in seconds";
          }
          leaf last-flooded-trigger {
            type enumeration {
              enum link-up {
                description "Link-up flooding trigger";
              }
              enum link-down {
                description "Link-up flooding trigger";
              }
              enum threshold-up {
                description
                  "Bandwidth reservation up threshold";
              }
              enum threshold-down {
                description
                  "Bandwidth reservation down threshold";
              }
              enum bandwidth-change {
                description "Banwidth capacity change";
              }
              enum user-initiated {
                description "Initiated by user";
              }
              enum srlg-change {
                description "SRLG property change";
              }
              enum periodic-timer {
                description "Periodic timer expired";
              }
            }
            description "Trigger for the last flood";
          }
          list advertized-level-areas {
            key level-area;
            description
              "List of areas the TE interface is advertised
              in";
            leaf level-area {
              type uint32;
              description
                "The IGP area or level where the TE
```

```
              interface state is advertised in";
          }
        }
      }
    }

    /* TE interface attributes grouping */
    grouping te-attributes {
      description "TE attributes configuration grouping";
      container config {
        description
          "Configuration parameters for interface TE
           attributes";
        uses te-metric_config;
        uses te-admin-groups_config;
        uses te-srlgs_config;
        uses te-switching-cap_config;
        uses te-flooding-parameters_config;
      }
      container state {
        config false;
        description
          "State parameters for interface TE metric";
        uses te-metric_config;
        uses te-admin-groups_config;
        uses te-srlgs_config;
        uses te-switching-cap_config;
        uses te-flooding-parameters_config;
        uses te-advertisements_state;
      }
    }

    grouping te-all-attributes {
      description
        "TE attributes configuration grouping for all
         interfaces";
      container config {
        description
          "Configuration parameters for all interface TE
           attributes";
        uses te-flooding-parameters_config;
      }
      container state {
        config false;
        description
          "State parameters for all interface TE metric";
        uses te-flooding-parameters_config;
      }
```

```
    }
    /*** End of TE interfaces device groupings ***/


    /**
     * TE device augmentations
     */
    augment "/te:te" {
      description "TE global container.";
      /* TE Interface Configuration Data */
      uses interfaces-grouping;
    }

    /* TE globals device augmentation */
    augment "/te:te/te:globals/te:config" {
      description
        "Global TE device specific configuration parameters";
      uses globals-device_config;
    }
    augment "/te:te/te:globals/te:state" {
      description
        "Global TE device specific state parameters";
      uses globals-device_config;
      uses globals-device_state;
    }

    /* TE tunnels device configuration augmentation */
    augment "/te:te/te:tunnels/te:tunnel/te:config" {
      description
        "Tunnel device dependent augmentation";
      uses lsp-device-timers_config;
    }
    augment "/te:te/te:tunnels/te:tunnel/te:state" {
      description
        "Tunnel device dependent augmentation";
      uses lsp-device-timers_config;
    }

    /* TE LSPs device state augmentation */
    augment "/te:te/te:lsps-state/te:lsp" {
      description
        "LSP device dependent augmentation";
      uses lsps-device_state;
    }

    /* TE interfaces RPCs/execution Data */
    rpc interfaces-rpc {
      description
```

```
        "Execution data for TE interfaces.";
    }

    /* TE Interfaces Notification Data */
    notification interfaces-notif {
      description
        "Notification messages for TE interfaces.";
    }
  }
  <CODE ENDS>
```

Figure 9: TE MPLS specific types YANG module

```
  <CODE BEGINS> file "ietf-te-mpls@2016-07-05.yang"
  module ietf-te-mpls {

    namespace "urn:ietf:params:xml:ns:yang:ietf-te-mpls";

    /* Replace with IANA when assigned */
    prefix "te-mpls";

    /* Import TE generic types */
    import ietf-te {
      prefix te;
    }

    /* Import MPLS TE specific types */
    import ietf-te-mpls-types {
      prefix te-mpls-type;
    }

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

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

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

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

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

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

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

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

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

Editor: Xia Chen
<mailto:jescia.chenxia@huawei.com>

Editor: Raqib Jones
<mailto:raqib@Brocade.com>

Editor: Bin Wen
<mailto:Bin_Wen@cable.comcast.com>";

```
  description
    "YANG data module for MPLS TE configurations,
    state, RPC and notifications.";

  revision "2016-07-05" {
    description "Latest update to MPLS TE YANG module.";
    reference "TBD";
  }

  /* MPLS TE tunnel properties*/
  grouping tunnel-mpls-properties_config {
    description "MPLS TE tunnel properties";
    uses te-mpls-type:tunnel-routing-properties;
    uses te-mpls-type:tunnel-forwarding-properties;
  }

  /*** End of MPLS TE tunnel configuration/state */


  /**
   * MPLS TE augmentations
   */

  /* MPLS TE tunnel augmentations */
  augment "/te:te/te:tunnels/te:tunnel/te:config" {
    description "MPLS TE tunnel config augmentations";
    uses tunnel-mpls-properties_config;
```

```
    }
    augment "/te:te/te:tunnels/te:tunnel/te:state" {
      description "MPLS TE tunnel state augmentations";
      uses tunnel-mpls-properties_config;
    }

    /* MPLS TE LSPs augmentations */
  }
  <CODE ENDS>
```

Figure 10: TE MPLS YANG module

```
  <CODE BEGINS> file "ietf-te-mpls-types@2016-07-05.yang"
  module ietf-te-mpls-types {

    namespace "urn:ietf:params:xml:ns:yang:ietf-te-mpls-types";

    /* Replace with IANA when assigned */
    prefix "te-mpls-types";

    import ietf-inet-types {
      prefix inet;
    }

    import ietf-mpls {
      prefix mpls;
    }

    organization
      "IETF TEAS Working Group";

    contact "Fill me";

    description
      "This module contains a collection of generally
      useful TE specific YANG data type defintions.";

    revision "2016-07-05" {
      description "Latest revision of TE MPLS types";
      reference "RFC3209";
    }

    identity backup-protection-type {
      description
        "Base identity for backup protection type";
    }

    identity backup-protection-link {
```

```
      base backup-protection-type;
      description
        "backup provides link protection only";
    }

    identity backup-protection-node-link {
      base backup-protection-type;
      description
        "backup offers node (preferred) or link protection";
    }

    identity bc-model-type {
      description
        "Base identity for Diffserv-TE bandwidth constraint
        model type";
    }

    identity bc-model-rdm {
      base bc-model-type;
      description
        "Russian Doll bandwidth constraint model type.";
    }

    identity bc-model-mam {
      base bc-model-type;
      description
        "Maximum Allocation bandwidth constraint
        model type.";
    }

    identity bc-model-mar {
      base bc-model-type;
      description
        "Maximum Allocation with Reservation
        bandwidth constraint model type.";
    }

    grouping bandwidth-constraint-values {
      description
        "Packet bandwidth contraints values";
      choice value-type {
        description
          "Value representation";
        case percentages {
          container perc-values {
            uses bandwidth-mpls-constraints;
            description
              "Percentage values";
```

```
          }
        }
        case absolutes {
          container abs-values {
            uses bandwidth-mpls-constraints;
            description
              "Absolute values";
          }
        }
      }
    }

    grouping bandwidth-mpls-reservable {
      description
        "Packet reservable bandwidth";
      choice bandwidth-value {
        description "Reservable bandwidth configuraiton choice";
        case absolute {
          leaf absolute-value {
            type uint32;
            description "Absolute value of the bandwidth";
          }
        }
        case precentage {
          leaf percent-value {
            type uint32 {
              range "0..4294967295";
            }
            description "Percentage reservable bandwidth";
          }
          description
            "The maximum reservable bandwidth on the
            interface";
        }
      }
      choice bc-model-type {
        description
          "Reservable bandwidth percentage capacity
          values.";
        case bc-model-rdm {
          container bc-model-rdm {
            description
              "Russian Doll Model Bandwidth Constraints.";
            uses bandwidth-mpls-constraints;
          }
        }
        case bc-model-mam {
          container bc-model-mam {
```

```
            uses bandwidth-mpls-constraints;
            description
              "Maximum Allocation Model Bandwidth
              Constraints.";
          }
        }
        case bc-model-mar {
          container bc-model-mar {
            uses bandwidth-mpls-constraints;
            description
              "Maximum Allocation with Reservation Model
              Bandwidth Constraints.";
          }
        }
      }
    }

    typedef bfd-type {
      type enumeration {
        enum classical {
          description "BFD classical session type.";
        }
        enum seamless {
          description "BFD seamless session type.";
        }
      }
      default "classical";
      description
        "Type of BFD session";
    }

    typedef bfd-encap-mode-type {
      type enumeration {
        enum gal {
          description
            "BFD with GAL mode";
        }
        enum ip {
          description
            "BFD with IP mode";
        }
      }
      default ip;
      description
        "Possible BFD transport modes when running over TE
        LSPs.";
    }
```

```
     grouping bandwidth-mpls-constraints {
       description "Bandwidth constraints.";
       container bandwidth-mpls-constraints {
         description
           "Holds the bandwidth contraints properties";
         leaf maximum-reservable {
           type uint32 {
             range "0..4294967295";
           }
           description
             "The maximum reservable bandwidth on the
             interface";
         }
         leaf-list bc-value {
           type uint32 {
             range "0..4294967295";
           }
           max-elements 8;
           description
             "The bandwidth contraint type";
         }
       }
     }

     grouping tunnel-forwarding-properties {
       description "Properties for using tunnel in forwarding.";
       container forwarding {
         description
           "Tunnel forwarding properties container";
         leaf binding-label {
           type mpls:mpls-label;
           description "MPLS tunnel binding label";
         }
         leaf load-share {
           type uint32 {
             range "1..4294967295";
           }
           description "ECMP tunnel forwarding
             load-share factor.";
         }
         choice policy-type {
           description
             "Tunnel policy type";
           container class {
             description
               "Tunnel forwarding per class properties";
             leaf class {
               type uint8 {
```

```
                range "1..7";
              }
              description
                "The class associated with this tunnel";
            }
          }
          container group {
            description
              "Tunnel frowarding per group properties";
            leaf-list classes {
              type uint8 {
                range "1..7";
              }
              description
                "The forwarding class";
            }
          }
        }
      }
    }

    grouping tunnel-routing-properties {
      description
        "TE tunnel routing properties";
      choice routing-choice {
        description
          "Announces the tunnel to IGP as either
          autoroute or forwarding adjacency.";
        case autoroute {
          container autoroute-announce {
            presence "Enable autoroute announce.";
            description
              "Announce the TE tunnel as autoroute to
              IGP for use as IGP shortcut.";
            leaf-list routing-afs {
              type inet:ip-version;
              description
                "Address families";
            }
            choice metric-type {
              description
                "Type of metric to use when announcing
                the tunnel as shortcut";
              leaf metric {
                type uint32 {
                  range "1..2147483647";
                }
                description
```

```
                  "Describes the metric to use when
                  announcing the tunnel as shortcut";
              }
              leaf relative-metric {
                type int32 {
                  range "-10..10";
                }
                description
                  "Relative TE metric to use when
                  announcing the tunnel as shortcut";
              }
              leaf absolute-metric {
                type uint32 {
                  range "1..2147483647";
                }
                description
                  "Absolute TE metric to use when
                  announcing the tunnel as shortcut";
              }
            }
          }
        }
        case forwarding-adjacency {
          container forwarding-adjacency {
            presence "Enable forwarding adjacency
              on the tunnel.";
            description
              "Announce the TE tunnel
              as forwarding adjacency.";
            leaf holdtime {
              type uint32 {
                range "0..4294967295";
              }
              description
                "Holdtime in seconds after
                tunnel becomes UP.";
            }
            leaf-list routing-afs {
              type inet:ip-version;
              description
                "Address families";
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

Figure 11: TE MPLS types YANG module

```
<CODE BEGINS> file "ietf-te-sr-mpls@2016-07-05.yang"
module ietf-te-sr-mpls {

  namespace "urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls";

  /* Replace with IANA when assigned */
  prefix "te-sr-mpls";

  /* Import TE generic types */
  import ietf-te {
    prefix te;
  }

  /* Import TE generic types */
  import ietf-te-types {
    prefix te-types;
  }

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

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

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

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

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

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

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

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

     Editor:   Xufeng Liu
               <mailto:xufeng.liu@ericsson.com>
```

```
        Editor:   Xia Chen
                  <mailto:jescia.chenxia@huawei.com>

        Editor:   Raqib Jones
                  <mailto:raqib@Brocade.com>

        Editor:   Bin Wen
                  <mailto:Bin_Wen@cable.comcast.com>";

    description
      "YANG data module for MPLS TE configurations,
      state, RPC and notifications.";

    revision "2016-07-05" {
      description "Latest update to MPLS TE YANG module.";
      reference "TBD";
    }

    identity sr-adjacency-protection-type {
      description
         "The Adj-SID base protection types";
    }

    identity sr-adjacency-protection-type-protected {
      base sr-adjacency-protection-type;
      description
         "The Adj-SID is eligible for protection (e.g.: using
          IPFRR or MPLS-FRR)";
    }

    identity sr-adjacency-protection-type-unprotected {
      base sr-adjacency-protection-type;
      description
         "The Adj-SID is not eligible for protection (e.g.: using
          IPFRR or MPLS-FRR)";
    }

    /* MPLS SR-TE tunnel properties*/
    grouping tunnel-sr-mpls-properties_config {
      description "MPLS TE SR tunnel properties";
      leaf path-signaling-type {
        type identityref {
          base te-types:path-signaling-type;
        }
        description "TE tunnel path signaling type";
      }
    }
```

```
    grouping named-sr-path-constraints_config {
      description "foo";
      leaf adjacency-protection {
        type identityref {
          base sr-adjacency-protection-type;
        }
        description "foo";
      }

    }

    /*** End of MPLS SR-TE tunnel configuration/state */


    /**
     * MPLS TE augmentations
     */

    /* MPLS TE global augmentations */
    augment "/te:te/te:globals/te:config/te:named-constraints" {
      description "foo";
      uses named-sr-path-constraints_config;
    }

    augment "/te:te/te:globals/te:state/te:named-constraints" {
      description "foo";
      uses named-sr-path-constraints_config;
    }

    /* MPLS TE tunnel augmentations */
    augment "/te:te/te:tunnels/te:tunnel/te:primary-paths" {
      description "MPLS TE tunnel config augmentations";
      uses tunnel-sr-mpls-properties_config;
    }
    augment "/te:te/te:tunnels/te:tunnel/te:primary-paths/" +
            "te:secondary-paths" {
      description "MPLS TE tunnel state augmentations";
      uses tunnel-sr-mpls-properties_config;
    }

    /* MPLS TE LSPs augmentations */
  }
  <CODE ENDS>
```

Figure 12: SR TE MPLS YANG module

5. IANA Considerations

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

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

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

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

URI: urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls XML: N/A, the requested URI is an XML namespace.

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

URI: urn:ietf:params:xml:ns:yang:ietf-te-mpls-types XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-te namespace: urn:ietf:params:xml:ns:yang:ietf-te prefix: ietf-te reference: RFC3209

name: ietf-te-device namespace: urn:ietf:params:xml:ns:yang:ietf-te prefix: ietf-te-device reference: RFC3209

name: ietf-te-mpls namespace: urn:ietf:params:xml:ns:yang:ietf-te-mpls prefix: ietf-te-mpls reference: RFC3209

name: ietf-te-sr-mpls namespace: urn:ietf:params:xml:ns:yang:ietf-te-sr-mpls prefix: ietf-te-sr-mpls

name: ietf-te-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-types prefix: ietf-te-types reference: RFC3209

name: ietf-te-mpls-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-mpls-types prefix: ietf-te-mpls-types reference: RFC3209

6. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF

users to a pre-configured subset of all available NETCONF protocol operations and content.

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

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

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

"/te/lsps-state": This list specifies the state derived LSPs. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

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

7. Acknowledgement

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

The authors would also like to thank Loa Andersson, Lou Berger, Aihua Guo, Dhruv Dhody, Anurag Sharma, and Xian Zhang for their comments and providing valuable feedback on this document.

8. References

8.1. Normative References

[I-D.bjorklund-netmod-structural-mount]
Bjorklund, M., "YANG Structural Mount", draft-bjorklund-netmod-structural-mount-02 (work in progress), February 2016.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-22 (work in progress), July 2016.

[I-D.ietf-teas-yang-rsvp]
Beeram, V., Saad, T., Gandhi, R., Liu, X., Shah, H., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-03 (work in progress), March 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc3473>.

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

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://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, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

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

8.2. Informative References

[I-D.clemm-netmod-mount]
Clemm, A., Medved, J., and E. Voit, "Mounting YANG-Defined Information from Remote Datastores", draft-clemm-netmod-mount-04 (work in progress), March 2016.

[I-D.openconfig-mpls-consolidated-model]
George, J., Fang, L., eric.osborne@level3.com, e., and R. Shakir, "MPLS / TE Model for Service Provider Networks", draft-openconfig-mpls-consolidated-model-02 (work in progress), October 2015.

[I-D.openconfig-netmod-opstate]
Shakir, R., Shaikh, A., and M. Hines, "Consistent Modeling of Operational State Data in YANG", draft-openconfig-netmod-opstate-01 (work in progress), July 2015.

Authors' Addresses

Tarek Saad (editor)
Cisco Systems Inc

Email: tsaad@cisco.com

Rakesh Gandhi
Cisco Systems Inc

Email: rgandhi@cisco.com

Xufeng Liu
Ericsson

Email: xufeng.liu@ericsson.com

Vishnu Pavan Beeram
Juniper Networks

Email: vbeeram@juniper.net

Himanshu Shah
Ciena

Email: hshah@ciena.com

Igor Bryskin
Huawei Technologies

Email: Igor.Bryskin@huawei.com

Xia Chen
Huawei Technologies

Email: jescia.chenxia@huawei.com

Raqib Jones
Brocade

Email: raqib@Brocade.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com