TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

T. Saad, Ed.
R. Gandhi
Cisco Systems Inc
X. Liu
Ericsson
V. Beeram
Juniper Networks
H. Shah
Ciena
X. Chen
Huawei Technologies
R. Jones
Brocade
B. Wen
Comcast
July 06, 2015

A YANG Data Model for Traffic Engineering Tunnels and Interfaces
draft-saad-teas-yang-te-02

Abstract

This document defines a YANG data model for the configuration and management of Traffic Engineering (TE) interfaces and tunnels. The model defines generic data that is reusable across multiple data and control plane protocols.

The data model covers the configuration, operational state, remote procedural calls, and event notifications data for TE 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 7, 2016.

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 defines a YANG data model that can be used to configure and manage TE interfaces and P2P or P2MP TE tunnels. This data model restricts to TE generic data that is control and data plane agnostic. It is expected that other protocol and data plane specific modules (e.g. RSVP-TE [RFC3209]) will augment this TE 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 document describes the YANG data model for the TE generic and helper modules. It also describes the high-level relationship between these modules and to other external protocol modules. The current revision of the draft focuses on configuration and state data aspects of the model. It is expected that the future revisions will cover RPC, and notification aspects.

Also, the models that define technology specific extensions to the generic TE model (e.g. OTN [RFC4328] TE extensions), are expected to be addressed in separate documents.

1.4.1. 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 the read-write container, under the "state" sub-container, as shown in Figure 3

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

The first option allows for the reusing of the containers that hold configuration data (in the "config" sub-container), and by placing state data under the read-only "state" sub-container of the parent container. However, when adopting this approach for ephemeral or purely derived states (e.g. auto tunnels), and since in this case the state hangs off the read-write parent container, it will be possible to delete 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. Further discussions on this topic are expected to close on the best choice to adopt.

2. Data Model Overview

Although the basis of TE elements remain similar across different vendor implementations, however, the details of a TE model will usually vary across different vendor implementations. Also, implementations may vary in their support of the complete set of TE features. The TE YANG module defined in this document is an attempts to define a vendor agnostic model that will prescribe to IETF

standard terminology when different representation of data is possible.

The model is composed of common building blocks that are independent of specific data or control plane instantiations. It covers data representation for the configuration, state, remote procedural calls (RPCs), and event notifications.

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 statistical 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. Design Objectives

The goal of this document is to define a TE data model that can represent different TE vendor implementations, while adhering to standard terminology and behavior when resolving differences in implementations.

The following considerations with respect data organization are taken into account when defining the model:

- 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 PSC or switching technologies as defined in [RFC3473]) are expected to be grouped in a technology- specific types module, e.g. "ietf-te-psc-types.yang". It is expected that technology specific types will augment TE generic types as shown in Figure 1

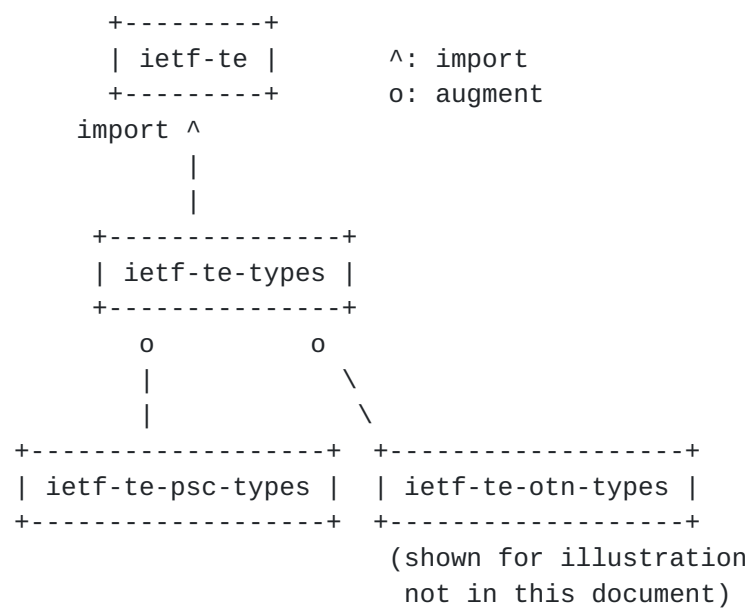

Figure 1: Relationship between generic and technology specific TE types modules

- TE generic module includes data elements that are control plane independent. Data elements specific to a control plane protocol (e.g. RSVP-TE [RFC3209]) are expected to be in a separate module that augments the TE generic module. It is also expected that data relevant to a specific instantiations of data plane technology will exist in a separate YANG module that augments the TE generic model, see Figure 2.

Figure 2: Relationship of TE module with other control plane protocol modules

- 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.

2.2. Optional Features

Optional features are features beyond the generic TE model, and hence, it is up to a vendor to decide whether or not to support of a particular feature on a particular device.

This module declares a number of TE functions as features (such as P2MP-TE, soft-preemption etc.). It is intended that vendors will extend this features list.

2.3. 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 (e.g. tunnel, interface or path). Vendors are expected to augment the above container(s) to provide the list of inheritance command for their implementations.

2.4. Vendor Configuration Models

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, auto-tunnels, etc.)

Data for the first two categories are contained under the respective "state" sub-container of the intended object (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 interfaces
            +-- 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>>
         .
         .
      +--ro tunnels-state
           <<ephemeral tunnels>>
rpcs:
   +---x globals-rpc
   +---x interfaces-rpc
   +---x tunnels-rpc
notifications:
   +---n globals-notif
   +---n interfaces-notif
   +---n tunnels-notif
```

Figure 3: TE 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
  - Reoptimization timers (periodic interval, LSP installation and cleanup)
- System-wide capabilities for TE state flooding
  - Periodic flooding interval
- System-wide capabilities that affect the originating, traversing and terminating LSPs. For example:
  - Path selection parameters (e.g. metric) at head-end LSR
  - Path protection parameters at head-end LSR
  - (Soft) preemption parameters
  - Fast reroute parameters

The approach described in [I-D.openconfig-netmod-opstate] is utilised 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)

```
module: ietf-te
   +--rw te!
      +--rw globals
      |  +--rw config
      |  +--ro state
      |  |  +--ro tunnels-counter?   uint32
```

```
|  |  +--ro lsps-counter?      uint32
|  +--rw named-admin-groups
|  |  +--rw config
|  |  |  +--rw named-admin-groups* [name]
|  |  |     +--rw name     string
|  |  |     +--rw group?   ietf-te-types:admin-groups
|  |  +--ro state
|  |     +--ro named-admin-groups* [name]
|  |        +--ro name     string
|  |        +--ro group?   ietf-te-types:admin-groups
|  +--rw named-srlgs
|  |  +--rw config
|  |  |  +--rw named-srlgs* [name]
|  |  |     +--rw name     string
|  |  |     +--rw group?   ietf-te-types:srlg
|  |  +--ro state
|  |     +--ro named-srlgs* [name]
|  |        +--ro name     string
|  |        +--ro group?   ietf-te-types:srlg
|  +--rw named-explicit-paths
|  |  +--rw config
|  |  |  +--rw named-explicit-paths* [name]
|  |  |     +--rw name                       string
|  |  |     +--rw explicit-route-objects* [index]
|  |  |        +--rw index                    uint8
|  |  |        +--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 explicit-route-usage?   identityref
|  |  +--ro state
|  |     +--ro named-explicit-paths* [name]
|  |        +--ro name                       string
|  |        +--ro explicit-route-objects* [index]
|  |           +--ro index                    uint8
|  |           +--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 explicit-route-usage?   identityref
|  +--rw named-path-constraints
|     +--rw config
|     |  +--rw named-path-constraints* [name]
|     |     +--rw name              string
|     |     +--rw path-selection
|     |        +--rw topology?                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
|     |        |  +--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*         srlg
|     |              +--:(named)
|     |                 +--rw constraints* [usage]
|     |                    +--rw usage         identityref
|     |                    +--rw constraint
|     |                       +--rw srlg-names* [name]
```

```
|     |                          +--rw name    string
|     +--ro state
|        +--ro named-path-constraints* [name]
|           +--ro name              string
|           +--ro path-selection
|              +--ro topology?        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
|              |  +--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*        srlg
|                    +--:(named)
|                       +--ro constraints* [usage]
|                          +--ro usage         identityref
|                          +--ro constraint
|                             +--ro srlg-names* [name]
|                                +--ro name    string
```

Figure 4: TE globals configuration and state tree

3.2. Interfaces Configuration and State Data

This branch of the data model covers configurations elements that control TE features behavior system-wide. Examples of such configuration data are:

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:

This branch of the model covers configuration and state data items, the corresponding applied state data, and possible derived state

pertaining to TE interfaces. Examples of tunnel configuration date for TE interfaces 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

```
module: ietf-te
   +--rw te!
      +--rw interfaces
      |  +--rw interface* [interface]
      |     +--rw interface                if:interface-ref
      |     +--rw config
      |     |  +--rw te-metric?   ietf-te-types:te-metric
      |     +--ro state
      |     |  +--ro te-metric? ietf-te-types:te-metric
      |     |  +--ro interface-advertisements_state
      |     |     +--ro flood-interval?           uint32
      |     |     +--ro last-flooded-time?        uint32
      |     |     +--ro next-flooded-time?        uint32
      |     |     +--ro last-flooded-trigger?     enumeration
      |     |     +--ro advertized-level-areas* [level-area]
      |     |        +--ro level-area    uint32
      |     +--rw te-admin-groups
      |     |  +--rw config
      |     |  |  +--rw (admin-group-type)?
      |     |  |     +--:(value-admin-groups)
      |     |  |     |  +--rw (value-admin-group-type)?
      |     |  |     |     +--:(value-admin-groups)
      |     |  |     |     |  +--rw admin-group?
      |     |  |     |     +--:(value-extended-admin-groups)
      |     |  |     |        +--rw extended-admin-group?
      |     |  |     +--:(named-admin-groups)
      |     |  |        +--rw named-admin-groups* [named-admin-group]
      |     |  |           +--rw named-admin-group    leafref
```

```
|     |  +--ro state
|     |     +--ro (admin-group-type)?
|     |        +--:(value-admin-groups)
|     |        |  +--ro (value-admin-group-type)?
|     |        |     +--:(value-admin-groups)
|     |        |     |  +--ro admin-group?
|     |        |     +--:(value-extended-admin-groups)
|     |        |        +--ro extended-admin-group?
|     |        +--:(named-admin-groups)
|     |           +--ro named-admin-groups* [named-admin-group]
|     |              +--ro named-admin-group    leafref
|     +--rw te-srlgs
|     |  +--rw config
|     |  |  +--rw (srlg-type)?
|     |  |     +--:(value-srlgs)
|     |  |     |  +--rw values* [value]
|     |  |     |     +--rw value    uint32
|     |  |     +--:(named-srlgs)
|     |  |        +--rw named-srlgs* [named-srlg]
|     |  |           +--rw named-srlg    leafref
|     |  +--ro state
|     |     +--ro (srlg-type)?
|     |        +--:(value-srlgs)
|     |        |  +--ro values* [value]
|     |        |     +--ro value    uint32
|     |        +--:(named-srlgs)
|     |           +--ro named-srlgs* [named-srlg]
|     |              +--ro named-srlg    leafref
|     +--rw te-switching-cap
|     |  +--rw config
|     |  |  +--rw switching-capabilities* [switching-capability]
|     |  |     +--rw switching-capability    identityref
|     |  |     +--rw encoding?               identityref
|     |  +--ro state
|     |     +--ro switching-capabilities* [switching-capability]
|     |        +--ro switching-capability    identityref
|     |        +--ro encoding?               identityref
|     +--rw te-flooding-parameters
|        +--rw config
|        |  +--rw thresholds
|        |     +--rw (type)?
|        |        +--:(equal-steps)
|        |        |  +--rw (equal-step-type)?
|        |        |     +--:(up-down-different-step)
|        |        |     |  +--rw up-step?      uint8
|        |        |     |  +--rw down-step?    uint8
|        |        |     +--:(up-down-same-step)
|        |        |        +--rw step?         uint8
```

```
|        |        +--:(unequal-steps)
|        |           +--rw up-steps* [value]
|        |           |  +--rw value    uint8
|        |           +--rw down-steps* [value]
|        |              +--rw value    uint8
|        +--ro state
|           +--ro thresholds
|              +--ro (type)?
|                 +--:(equal-steps)
|                 |  +--ro (equal-step-type)?
|                 |     +--:(up-down-different-step)
|                 |     |  +--ro up-step?      uint8
|                 |     |  +--ro down-step?    uint8
|                 |     +--:(up-down-same-step)
|                 |        +--ro step?         uint8
|                 +--:(unequal-steps)
|                    +--ro up-steps* [value]
|                    |  +--ro value    uint8
|                    +--ro down-steps* [value]
|                       +--ro value    uint8
```

Figure 5: TE interfaces configuration and state tree

The state corresponding to the TE interfaces applied configuration, protocol derived state, and stats and counters all fall under the interface attributes "state" sub-container as shown in Figure 6 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 6: 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 utilised for the inclusion of operational and statistical data as shown in Figure 7.

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

Figure 7: TE interface state tree

Examples of tunnel configuration date for TE tunnels:

- Name and type (e.g. P2P, P2MP) of tunnel

- Admin-state

- Primary and secondary paths

- Routing usage (auto-route announce, forwarding adjacency)

- Policy based routing (PBR) parameters

```
   +--rw te!
      +--rw tunnels
      |  +--rw tunnel* [name type]
      |     +--rw name          string
      |     +--rw type          identityref
      |     +--rw identifier?   uint16
      |     +--rw config
      |     |  +--rw description?           string
      |     |  +--rw admin-status?          identityref
      |     |  +--rw (routing-choice)?
      |     |  |  +--:(autoroute)
      |     |  |  |  +--rw autoroute-announce!
      |     |  |  |     +--rw routing-afs*       inet:ip-version
      |     |  |  |     +--rw (metric-type)?
      |     |  |  |        +--:(metric)
      |     |  |  |        |  +--rw metric?            uint32
      |     |  |  |        +--:(relative-metric)
      |     |  |  |        |  +--rw relative-metric?   int32
      |     |  |  |        +--:(absolute-metric)
      |     |  |  |           +--rw absolute-metric?   uint32
      |     |  |  +--:(forwarding-adjacency)
      |     |  |     +--rw forwarding-adjacency!
      |     |  |        +--rw holdtime?      uint32
      |     |  |        +--rw routing-afs*   inet:ip-version
      |     |  +--rw forwarding
      |     |  |  +--rw load-share?   uint32
      |     |  |  +--rw (policy-type)?
      |     |  |     +--:(class)
      |     |  |     |  +--rw class
      |     |  |     |     +--rw class?   uint8
      |     |  |     +--:(group)
      |     |  |        +--rw group
      |     |  |           +--rw classes*   uint8
      |     |  +--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 (path-type)?
      |     |     +--:(p2p)
      |     |     |  +--rw destination?            inet:ip-address
      |     |     |  +--rw primary-paths* [preference]
      |     |     |     +--rw preference            uint8
      |     |     |     +--rw tunnel-path-params
      |     |     |     |  +--rw path-named-constraint?   leafref
      |     |     |     |  +--rw path-selection
```

```
|     |     |     |  |  +--rw topology?         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
|     |     |     |  |  |  +--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*        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?      leafref
|     |     |     |  +--rw no-cspf?                 empty
|     |     |     |  +--rw lockdown?                empty
|     |     |     +--rw seondary-paths* [preference]
|     |     |        +--rw preference            uint8
|     |     |        +--rw tunnel-path-params
|     |     |           +--rw path-named-constraint?   leafref
|     |     |           +--rw path-selection
|     |     |              |  +--rw topology?        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
|     |     |              |  |  +--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* 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? leafref
|     |     |          +--rw no-cspf?                empty
|     |     |          +--rw lockdown?               empty
|     |     +--:(p2mp) {ietf-te-types:p2mp-te}?
|     |        +--rw p2mp-paths* [destination]
|     |           +--rw destination      inet:ip-address
|     |           +--rw primary-paths* [preference]
|     |              +--rw preference           uint8
|     |              +--rw tunnel-path-params
|     |              |  +--rw path-named-constraint?   leafref
|     |              |  +--rw path-selection
|     |              |  |  +--rw topology?        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
|     |              |  |  |  +--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*        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? leafref
|     |              |  +--rw no-cspf?                 empty
|     |              |  +--rw lockdown?                empty
|     |              +--rw seondary-paths* [preference]
|     |                 +--rw preference            uint8
|     |                 +--rw tunnel-path-params
|     |                    +--rw path-named-constraint?   leafref
|     |                    +--rw path-selection
|     |                    |  +--rw topology?        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
|     |                    |  |  +--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*        srlg
|     |                    |        +--:(named)
|     |                    |           +--rw constraints* [usage]
|     |                    |              +--rw usage identityref
|     |                    |              +--rw constraint
|     |                    |                 +--rw srlg-names*
|     |                    |                    +--rw name string
|     |                    +--rw (type)?
|     |                    |  +--:(dynamic)
|     |                    |  |  +--rw dynamic? empty
|     |                    |  +--:(explicit)
|     |                    |     +--rw explicit-path-name? leafref
|     |                    +--rw no-cspf?                 empty
|     |                    +--rw lockdown?                empty
|     +--ro state
|        +--ro description?            string
|        +--ro admin-status?           identityref
|        +--ro (routing-choice)?
|        |  +--:(autoroute)
|        |  |  +--ro autoroute-announce!
|        |  |     +--ro routing-afs*       inet:ip-version
|        |  |     +--ro (metric-type)?
|        |  |        +--:(metric)
|        |  |        |  +--ro metric?            uint32
|        |  |        +--:(relative-metric)
|        |  |        |  +--ro relative-metric?   int32
|        |  |        +--:(absolute-metric)
|        |  |           +--ro absolute-metric?   uint32
|        |  +--:(forwarding-adjacency)
|        |     +--ro forwarding-adjacency!
|        |        +--ro holdtime?      uint32
|        |        +--ro routing-afs*   inet:ip-version
|        +--ro forwarding
|        |  +--ro load-share?   uint32
|        |  +--ro (policy-type)?
|        |     +--:(class)
|        |     |  +--ro class
|        |     |     +--ro class?   uint8
|        |     +--:(group)
|        |        +--ro group
|        |           +--ro classes*   uint8
|        +--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 (path-type)?
|        |  +--:(p2p)
|        |  |  +--ro destination?            inet:ip-address
|        |  |  +--ro primary-paths* [preference]
|        |  |     +--ro preference            uint8
|        |  |     +--ro tunnel-path-params
|        |  |     |  +--ro path-named-constraint?   leafref
|        |  |     |  +--ro path-selection
|        |  |     |  |  +--ro topology?        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
|        |  |     |  |  |  +--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*        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?      leafref
|        |  |     |  +--ro no-cspf?                empty
|        |  |     |  +--ro lockdown?               empty
|        |  |     +--ro seondary-paths* [preference]
|        |  |        +--ro preference           uint8
|        |  |        +--ro tunnel-path-params
|        |  |           +--ro path-named-constraint?   leafref
|        |  |           +--ro path-selection
```

```
|        |  |           |  +--ro topology?        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
|        |  |           |  |  +--ro (style)?
|        |  |           |  |     +--:(values)
|        |  |           |  |     |  +--ro value?         uint32
|        |  |           |  |     |  +--ro mask?          uint32
|        |  |           |  |     +--:(named)
|        |  |           |  |        +--ro constraints* [usage]
|        |  |           |  |           +--ro usage identityref
|        |  |           |  |           +--ro constraint
|        |  |           |  |              +--ro affinity-names*
|        |  |           |  |                 +--ro name    string
|        |  |           |  +--ro tunnel-path-srlgs
|        |  |           |     +--ro (style)?
|        |  |           |        +--:(values)
|        |  |           |        |  +--ro usage? identityref
|        |  |           |        |  +--ro values*        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? leafref
|        |  |           +--ro no-cspf?                 empty
|        |  |           +--ro lockdown?                empty
|        |  +--:(p2mp) {ietf-te-types:p2mp-te}?
|        |     +--ro p2mp-paths* [destination]
|        |        +--ro destination      inet:ip-address
|        |        +--ro primary-paths* [preference]
|        |           +--ro preference            uint8
|        |           +--ro tunnel-path-params
|        |           |  +--ro path-named-constraint?   leafref
|        |           |  +--ro path-selection
|        |           |  |  +--ro topology?        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
       |        |           |  |  |  +--ro (style)?
       |        |           |  |  |     +--:(values)
       |        |           |  |  |     |  +--ro value?         uint32
       |        |           |  |  |     |  +--ro mask?          uint32
       |        |           |  |  |     +--:(named)
       |        |           |  |  |        +--ro constraints* [usage]
       |        |           |  |  |           +--ro usage identityref
       |        |           |  |  |           +--ro constraint
       |        |           |  |  |              +--ro affinity-names*
       |        |           |  |  |                 +--ro name    string
       |        |           |  |  +--ro tunnel-path-srlgs
       |        |           |  |     +--ro (style)?
       |        |           |  |        +--:(values)
       |        |           |  |        |  +--ro usage? identityref
       |        |           |  |        |  +--ro values*        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? leafref
       |        |           |  +--ro no-cspf?                 empty
       |        |           |  +--ro lockdown?                empty
       |        |           +--ro seondary-paths* [preference]
       |        |              +--ro preference            uint8
       |        |              +--ro tunnel-path-params
       |        |                 +--ro path-named-constraint?   leafref
       |        |                 +--ro path-selection
       |        |                 |  +--ro topology?        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
       |        |                 |  |  +--ro (style)?
       |        |                 |  |     +--:(values)
       |        |                 |  |     |  +--ro value?         uint32
       |        |                 |  |     |  +--ro mask?          uint32
       |        |                 |  |     +--:(named)
       |        |                 |  |        +--ro constraints* [usage]
```

```
|       |                |  |          +--ro usage identityref
|       |                |  |          +--ro constraint
|       |                |  |             +--ro affinity-names*
|       |                |  |                +--ro name string
|       |                |  +--ro tunnel-path-srlgs
|       |                |     +--ro (style)?
|       |                |        +--:(values)
|       |                |        |  +--ro usage? identityref
|       |                |        |  +--ro values*        srlg
|       |                |        +--:(named)
|       |                |           +--ro constraints* [usage]
|       |                |              +--ro usage identityref
|       |                |              +--ro constraint
|       |                |                 +--ro srlg-names*
|       |                |                    +--ro name string
|       |                |  +--ro (type)?
|       |                |  |  +--:(dynamic)
|       |                |  |  |  +--ro dynamic?            empty
|       |                |  |  +--:(explicit)
|       |                |  |     +--ro explicit-path-name? leafref
|       |                |  +--ro no-cspf?                 empty
|       |                |  +--ro lockdown?                empty
|       |  +--ro oper-status?            identityref
|       |  +--ro lsp* [source destination tunnel-id lsp-id]
|       |     +--ro source                leafref
|       |     +--ro destination           leafref
|       |     +--ro tunnel-id             leafref
|       |     +--ro lsp-id                leafref
|       |     +--ro extended-tunnel-id?   leafref
|       |     +--ro type?                 leafref
|  +--ro tunnels-state
|  |  +--ro tunnel* [name type]
|  |     +--ro name          string
|  |     +--ro type          identityref
|  |     +--ro identifier?   uint16
```

Figure 8: TE tunnels configuration and state tree

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) as shown in Figure 9.

In the model, the nodes holding LSPs data exist in a read-only list as shown below:

```
         +--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-status?    enumeration
               +--ro lsp-operational-status?   empty
               +--ro lsp-timers
               |  +--ro life-time?         uint32
               |  +--ro time-to-install?   uint32
               |  +--ro time-to-die?       uint32
               +--ro downstream-info
               |  +--ro nhop?                  inet:ip-address
               |  +--ro outgoing-interface?    if:interface-ref
               |  +--ro neighbor?              inet:ip-address
               |  +--ro label?                 uint32
               +--ro upstream-info
                  +--ro nhop?                  inet:ip-address
                  +--ro incoming-interface?    if:interface-ref
                  +--ro neighbor?              inet:ip-address
                  +--ro label?                 uint32
```

Figure 9: TE LSPs state tree

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

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

4. TE Generic and Helper YANG Modules

```
<CODE BEGINS>file "ietf-te-types@2015-07-06.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;
  }

  organization
    "IETF TEAS Working Group";

  contact "Fill me";

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

  revision 2015-07-06 {
    description "Latest revision of TE basic types";
    reference "RFC3209";
  }

  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 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";
}

/* 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 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";
    }

    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";
}

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

```
     /**
      * 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 {
           type 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 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
     **/

    /* TE interface flooding parameters */
    grouping interface-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";
              }
            }
          }
        }
      }

    grouping interface-te-flooding-parameters {
      description "Interface TE flooding properties.";
      container te-flooding-parameters {
        description
          "Parameters for interface flooding";
        container config {
          description
            "Configuration parameters for interface
            flooding";
          uses interface-te-flooding-parameters_config;
        }
        container state {
          config false;
          description
            "State parameters for interface flooding";
          uses interface-te-flooding-parameters_config;
        }
      }
    }
  }
  <CODE ENDS>
```

Figure 10: TE basic types YANG module

```
  <CODE BEGINS> file "ietf-te-psc-types@2015-07-06.yang"
  module ietf-te-psc-types {

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

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

    import ietf-inet-types { prefix inet; }

    organization
      "IETF TEAS Working Group";

    contact "Fill me";
```

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

     revision 2015-07-06 {
       description "Latest revision of TE MPLS/packet types";
       reference "RFC3209";
     }

     /* Describes egress LSP label allocation */
     typedef egress-label {
       type enumeration {
         enum "IPv4-EXPLICIT-NULL" {
           description
             "Use IPv4 explicit-NULL MPLS label at the
             egress";
         }
         enum "IPv6-EXPLICIT-NULL" {
           description
             "Use IPv6 explicit-NULL MPLS label at the
             egress";
         }
         enum "IMPLICIT-NULL" {
           description
             "Use implicit-NULL MPLS label at the egress";
         }
         enum "NON-NULL"{
           description
           "Use a non NULL MPLS label at the egress";
       }
       }
       description
         "Describes egress label allocation";
     }

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

     identity backup-facility {
       base backup-type;
       description
         "Use facility backup to protect LSPs traversing
         protected TE interface";
       reference
         "RFC49090: RSVP-TE Fast Reroute";
     }
```

```
    identity backup-detour {
      base backup-type;
      description
        "Use detour or 1-for-1 protection";
      reference
        "RFC49090: RSVP-TE Fast Reroute";
    }

    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-psc-constraints;
            description
              "Percentage values";
          }
        }
        case absolutes {
          container abs-values {
            uses bandwidth-psc-constraints;
            description
              "Absolute values";
          }
        }
      }
    }

    grouping bandwidth-psc-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-psc-constraints;
          }
        }
        case bc-model-mam {
          container bc-model-mam {
            uses bandwidth-psc-constraints;
            description
              "Maximum Allocation Model Bandwidth
              Constraints.";
          }
        }
        case bc-model-mar {
          container bc-model-mar {
            uses bandwidth-psc-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-psc-constraints {
      description "Bandwidth constraints.";
      container bandwidth-psc-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 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 packet/MPLS specific types YANG module

```
  <CODE BEGINS> file "ietf-te@2015-07-06.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 ietf-te-types;
    }

    /* Import TE packet specific types */
    import ietf-te-psc-types {
      prefix ietf-te-psc-types;
    }

    import ietf-interfaces {
      prefix if;
    }

    import ietf-inet-types {
      prefix inet;
    }

    organization
      "IETF TEAS Working Group";

    contact
      "Fill me";

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

```
     revision 2015-07-06 {
       description "Latest update to TE generic YANG module.";
       reference "TBD";
     }

     /**
      * TE interface 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 value-admin-groups {
               description
                 "Administrative group/Resource
                 class/Color.";
               leaf admin-group {
                 type ietf-te-types:admin-group;
                 description
                   "TE interface administrative group";
               }
             }
             case value-extended-admin-groups {
               if-feature ietf-te-types:extended-admin-groups;
               description
                 "Extended administrative group/Resource
                 class/Color.";
               leaf extended-admin-group {
                 type ietf-te-types:extended-admin-group;
                 description
                   "TE interface extended administrativei
                   group";
               }
             }
           }
         }
         case named-admin-groups {
           list named-admin-groups {
             if-feature ietf-te-types:extended-admin-groups;
             if-feature ietf-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/" +
                  "named-admin-groups/config/"+
                  "named-admin-groups/name";
              }
              description
                "A named admin-group entry";
            }
          }
        }
      }
    }

    grouping te-admin-groups {
      description "TE admin-group configuration grouping";
      container te-admin-groups {
        description
          "Configuration parameters for interface
          administrative groups";
        container config {
          description
            "Configuration parameters for interface
            administrative groups";
          uses te-admin-groups_config;
        }
        container state {
          config false;
          description
            "Configuration parameters for interface
            administrative groups";
          uses te-admin-groups_config;
        }
      }
    }

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

    grouping te-srlgs {
      description "TE SRLG configuration grouping";
      container te-srlgs {
        description "TBD";
        container config {
          description
            "Configuration parameters for interface SRLG
            groups";
          uses te-srlgs_config;
        }
        container state {
          config false;
          description
            "State parameters for interface SRLG groups";
          uses te-srlgs_config;
        }
      }
    }

    grouping te-metric_config {
      description "Interface TE metric grouping";
```

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

    grouping te-attributes {
      description "TE attributes configuration grouping";
      container config {
        description
          "Configuration parameters for interface TE
           attributes";
        uses te-metric_config;
      }
      container state {
        config false;
        description
          "State parameters for interface TE metric";
        uses te-metric_config;
        uses interface-advertisements_state;
      }
    }

    /* 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 ietf-te-types:switching-capabilities;
          }
          description
            "Switching Capability for this interface";
        }
        leaf encoding {
          type identityref {
            base ietf-te-types:lsp-encoding-types;
          }
          description
            "Encoding supported by this interface";
        }
      }
    }
```

```
    grouping te-switching-cap {
      description "TE interface switching capability grouping";
      container te-switching-cap {
        description
          "Interface switching capabilities container";
        container config {
          description
            "Configuration parameters for interface
            switching capabilities";
          uses te-switching-cap_config;
        }
        container state {
          config false;
          description
            "State parameters for interface switching
            capabilities";
          uses te-switching-cap_config;
        }
      }
    }

    grouping interface-advertisements_state {
      description
        "TE interface advertisements state grouping";
      container interface-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";
          }
        }
      }
    }
    /*** End of TE interface groupings ***/

    /**
     * TE tunnel generic groupings
     */

    /* TE tunnel path properties */
```

```
     grouping tunnel-path-params {
       description
         "Tunnel path properties grouping";
       container tunnel-path-params {
         description
           "Defines a TE tunnel path properties";
         leaf path-named-constraint {
           if-feature ietf-te-types:named-path-constraints;
           type leafref {
             path "/te/globals/named-path-constraints/config/"+
               "named-path-constraints/name";
           }
           description
             "Reference to a globally defined named path
             constraint set";
         }
         uses ietf-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 leafref {
                 path "/te/globals/named-explicit-paths/config/"+
                   "named-explicit-paths/name";
               }
               description
                 "Reference to a globally defined
                 explicit-path";
             }
           }
         }
         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.";
        }
      }
    }

    grouping tunnel-properties_config {
      description
        "Configuration parameters relating to TE tunnel";
      leaf description {
        type string;
        description
          "TE tunnel description.";
      }
      leaf admin-status {
        type identityref {
          base ietf-te-types:state-type;
        }
        default ietf-te-types:state-up;
        description "TE tunnel administrative state.";
      }
      uses ietf-te-psc-types:tunnel-routing-properties;
      uses ietf-te-psc-types:tunnel-forwarding-properties;
      uses ietf-te-types:tunnel-bidir-assoc-properties;
      choice path-type {
        description
          "Describes the path type";
        case p2p {
          leaf destination {
            type inet:ip-address;
            description
              "P2P tunnel destination address";
          }
          /* P2P list of path(s) */
          list primary-paths {
            key "preference";
            description
              "List of primary paths for this
              tunnel.";
            leaf preference {
              type uint8 {
                range "1..255";
              }
              description
                "Specifies a preference for
                this path. The lower the
                number higher the
                preference";
```

```
            }
            uses tunnel-path-params;
            list seondary-paths {
              key "preference";
              description
                "List of secondary paths for this
                tunnel.";
              leaf preference {
                type uint8 {
                  range "1..255";
                }
                description
                  "Specifies a preference for
                  this path. The lower the
                  number higher the
                  preference";
              }
              uses tunnel-path-params;
            }
          }
        }
        case p2mp {
          if-feature ietf-te-types:p2mp-te;
          list p2mp-paths {
            key "destination";
            description
              "List of destinations and their
              paths.";
            leaf destination {
              type inet:ip-address;
              description
                "P2MP destination leaf address";
            }
            list primary-paths {
              key "preference";
              description
                "List of primary paths";
              leaf preference {
                type uint8 {
                  range "1..255";
                }
                description
                  "Specifies a preference for
                  this path. The lower the
                  number higher the
                  preference";
              }
              uses tunnel-path-params;
```

```
              list seondary-paths {
                key "preference";
                description
                  "List of secondary paths";
                leaf preference {
                  type uint8 {
                    range "1..255";
                  }
                  description
                    "Specifies a preference
                    for this path. The lower
                      the number higher
                        the preference";
                }
                uses tunnel-path-params;
              }
            }
          }
        }
      }

    grouping tunnel-properties {
      description
        "Top level grouping for tunnel properties.";
      container config {
        description
          "Configuration parameters relating to
          tunnel properties";
        uses tunnel-properties_config;
      }
      container state {
        config false;
        description
          "State information associated with tunnel
          properties";
        uses tunnel-properties_config;
        uses tunnel-properties_state;
      }
    }

    grouping tunnel-properties_state {
      description
        "State parameters relating to TE tunnel";
      leaf oper-status {
        type identityref {
          base ietf-te-types:state-type;
        }
```

```
          description "TE tunnel operational state.";
        }
        list lsp {
          key "source destination tunnel-id lsp-id";
          description "List of LSPs associated with the tunnel.";

          leaf source {
            type leafref {
              path "/te/lsps-state/lsp/source";
            }
            description
              "Tunnel sender address extracted from
              SENDER_TEMPLATE  object";
            reference "RFC3209";
          }
          leaf destination {
            type leafref {
              path "/te/lsps-state/lsp/destination";
            }
            description
              "Tunnel endpoint address extracted from
              SESSION object";
            reference "RFC3209";
          }
          leaf tunnel-id {
            type leafref {
              path "/te/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 "/te/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 "/te/lsps-state/lsp/extended-tunnel-id";
```

```
          }
          description
            "Extended Tunnel ID of the LSP.";
          reference "RFC3209";
        }
        leaf type {
          type leafref {
            path "/te/lsps-state/lsp/type";
          }
          description "LSP type P2P or P2MP";
        }
      }
    }
    /*** End of TE tunnel groupings ***/

    /**
     * LSP related generic groupings
     */

    grouping lsp-properties_state {
      description
        "State parameters relating to LSP";
      leaf oper-status {
        type identityref {
          base ietf-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-status {
        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";
      }

      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 {
          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 {
        description
          "upstream information";

        leaf nhop { // phop?
          type inet:ip-address;
          description
            "upstream nexthop.";
        }

        leaf incoming-interface {
          type if:interface-ref;
          description
            "upstream interface.";
        }

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

        leaf label {
          type uint32;
          description
            "upstream label.";
        }
      }
    }
    /*** End of TE LSP groupings ***/

    /**
     * TE global generic groupings
     */

    grouping global-attributes_config {
      description
        "Top level grouping for global config data.";
    }

    grouping global-attributes_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";
      }
    }

    grouping global-attributes {
      description "TE Global attributes grouping";
      container config {
        description
          "Global configuration parameters";
        uses global-attributes_config;
      }
      container state {
        config false;
        description
          "Global configuration parameters";
        uses global-attributes_config;
        uses global-attributes_state;
      }
    }

    grouping named-admin-groups_config {
      description
        "Global named administrative groups configuration
        grouping";
      list named-admin-groups {
        if-feature ietf-te-types:extended-admin-groups;
        if-feature ietf-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 group {
          type ietf-te-types:admin-groups;
          description
            "An SRLG value";
        }
```

```
      }
    }

    grouping named-admin-groups {
      description
        "Named admin groups grouping";
      container named-admin-groups {
        description
          "Named admin groups container";
        container config {
          description
            "Configuration parameters for named admin
            groups";
          uses named-admin-groups_config;
        }
        container state {
          config false;
          description
            "State parameters for named admin groups";
          uses named-admin-groups_config;
        }
      }
    }

    grouping named-srlgs_config {
      description
        "Global named SRLGs configuration
        grouping";
      list named-srlgs {
        if-feature ietf-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 ietf-te-types:srlg;
          description "An SRLG value";
        }
      }
    }

    grouping named-srlgs {
      description
```

```
        "Global named SRLGs grouping";
      container named-srlgs {
        description
          "Named SRLGs container";
        container config {
          description
            "Configuration parameters for named SRLG groups";
          uses named-srlgs_config;
        }
        container state {
          config false;
          description
            "State parameters for named SRLG groups";
          uses named-srlgs_config;
        }
      }
    }

    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";
          }
          uses ietf-te-types:explicit-route-subobject;
          leaf explicit-route-usage {
            type identityref {
              base ietf-te-types:route-usage-type;
            }
```

```
            description
              "An IP hop action.";
          }
        }
      }
    }

    grouping named-explicit-paths {
      description
        "Global named explicit path grouping";
      container named-explicit-paths {
        description
          "Nmaed explicit paths container";
        container config {
          description
            "Configuration parameters for named explicit
            paths";
          uses named-explicit-paths_config;
        }
        container state {
          config false;
          description
            "State parameters for named explicit paths";
          uses named-explicit-paths_config;
        }
      }
    }

    grouping named-path-constraints_config {
      description
        "Global named path constraints configuration
        grouping";
      list named-path-constraints {
        if-feature ietf-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 ietf-te-types:tunnel-path-selection;
      }
    }

    grouping named-path-constraints {
```

```
      description
        "Global named path constraints grouping";
      container named-path-constraints {
        description
          "Nmaed explicit paths container";
        container config {
          description
            "Configuration parameters for named explicit
            paths";
          uses named-path-constraints_config;
        }
        container state {
          config false;
          description
            "State parameters for named explicit paths";
          uses named-path-constraints_config;
        }
      }
    }
    /*** End of TE global groupings ***/

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

      /* TE Global Configuration Data */
      container globals {
        description
          "Configuration data for Global System-wide
          Traffic Engineering.";
        uses global-attributes;
        uses named-admin-groups;
        uses named-srlgs;
        uses named-explicit-paths;
        uses named-path-constraints;
      }

      /* TE Interface Configuration Data */
      container interfaces {
        description
          "Configuration data model for TE interfaces.";
        list interface {
          key "interface";
          description "TE interfaces.";
```

```
          leaf interface {
            type if:interface-ref;
            description
              "TE interface name.";
          }
          /* TE interface parameters */
          uses te-attributes;
          uses te-admin-groups;
          uses te-srlgs;
          uses te-switching-cap;
          /* TE interface flooding parameters */
          uses ietf-te-types:interface-te-flooding-parameters;
        }
      }

      /* TE Tunnel Configuration Data */
      container tunnels {
        description
          "Configuration, operational, notification and RPC
          data model for TE tunnels.";

        list tunnel {
          key "name type";
          unique "identifier";
          description "TE tunnel.";
          leaf name {
            type string;
            description "TE tunnel name.";
          }
          leaf type {
            type identityref {
              base ietf-te-types:tunnel-type;
            }
            description "TE tunnel type.";
          }
          leaf identifier {
            type uint16;
            description
              "TE tunnel Identifier.";
          }
          uses tunnel-properties;
        }
      }

      /* TE Tunnel Ephemeral State Data (TBD) */
      container tunnels-state {
        config "false";
        description
```

```
          "Derived state corresponding to ephemeral tunnels";

        list tunnel {
          key "name type";
          unique "identifier";
          description "TE tunnel.";
          leaf name {
            type string;
            description "TE tunnel name.";
          }
          leaf type {
            type identityref {
              base ietf-te-types:tunnel-type;
            }
            description "TE tunnel type.";
          }
          leaf identifier {
            type uint16;
            description
              "TE tunnel Identifier.";
          }

        }
      }

      /* TE LSPs State Data */
      container lsps-state {
        config "false";
        description "MPLS-TE LSP 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 ietf-te-types:tunnel-type;
            }
            description "The LSP type P2P or P2MP";
          }
          uses lsp-properties_state;
        }
      }
    }

    /* 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 Interfaces Notification Data */
    notification interfaces-notif {
      description
        "Notification messages for TE interfaces.";
    }

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

Figure 12: TE generic 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-types XML: N/A, the requested URI is an XML namespace.

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

name: ietf-te-psc-types namespace: urn:ietf:params:xml:ns:yang:ietf-te-psc-types prefix: ietf-te-psc-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/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 Lou Berger for reviewing and providing valuable feedback on this document.

8. References

8.1. Normative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-19 (work in progress), May 2015.

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

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

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

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

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

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

8.2. Informative References

[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-00 (work in progress), March 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-00 (work in progress), March 2015.

[RFC3473] Berger, L., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[RFC4328] Papadimitriou, D., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Extensions for G.709 Optical Transport Networks Control", RFC 4328, January 2006.

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

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