TEAS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 23, 2015

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

March 22, 2015

A YANG Data Model for Traffic Engineering Tunnels and Interfaces
draft-saad-teas-yang-te-01

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 September 23, 2015.

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) 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 is the initial version of the TE basic and its helper modules. It also describes the high-level relationship of these modules to other external protocol modules. The current revision of the TE basic model focuses on defining configuration data. However, future revisions are expected to cover state, RPC, and notification data.

It is also expected that models for technology specific extensions to the basic TE model (e.g. OTN [RFC4328] TE extension) , if needed, will likely be published in separate drafts.

2. Data Model Overview

Although the basis of TE elements remain similar across different vendor implementations, the detailed 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 has the 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.

2.1. Design Objectives

The goal of this document is to define a TE data model that can represent such different 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 basic types module "ietf-te-types.yang"

- reusable TE data elements that are data plane specific (e.g. packet PSC or switching techologies as defined in [RFC3473]) are expected to be grouped in a technology- specific types module. It is expected that technology specific types will augment TE basic types as shown in Figure 1

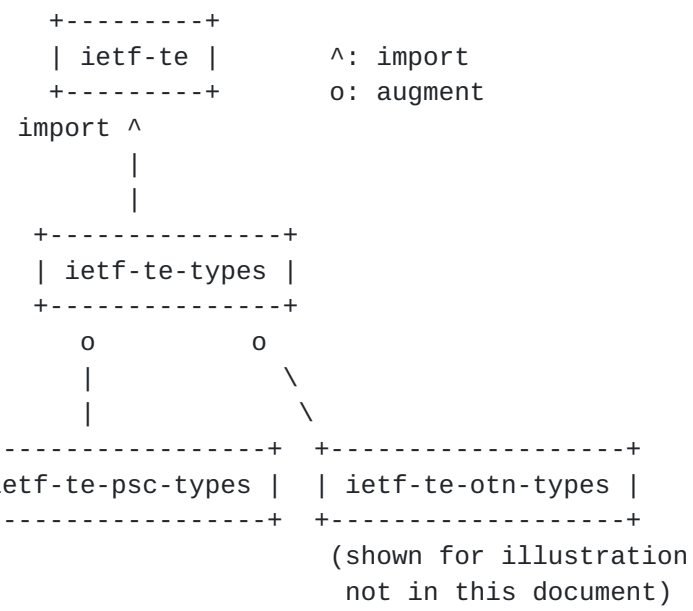

Figure 1: Relationship between generic and technology specific TE types modules

o TE basic module includes data elements that are control plane independent. Data elements specific to a control plane protocol (e.g. RSVP-TE) are expected to be in a separate module that augments the TE basic 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 basic 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 basic TE model, and hence, it is up to a vendor to decide whether 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.

Currently, there is on-going discussion with respect to the approach to take at IETF. Options being considered are:

- standard models to support both and vendors to pick the style that best suit them

- standard models to support both and requires vendors to support both styles

- standard models to pick one of the two style and vendors to support at least that

The model proposed in this document will adhere to the final outcome of those discussions.

3. Model Organization

This model defines configuration, state, execution, and notifications data for TE globals, interfaces, and tunnels properties.

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.

```
module: ietf-te
   +--rw te!
      +--rw globals
         .
         .
      +--rw interfaces
         .
         .
      +--rw tunnels
         .
         .
      +--ro global-state
         .
         .
      +--ro interfaces-state
         .
         .
      +--ro tunnels-state
         .
         .
rpcs:
   +---x globals-rpc
   +---x interfaces-rpc
   +---x tunnels-rpc
notifications:
   +---n globals-notif
   +---n interfaces-notif
   +---n tunnels-notif
```

3.1. TE Configuration Data

Following subsections provide overview of parts of the model pertaining to configuration data.

3.1.1. Global Configuration Data

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

- Auto-bandwidth parameters: control and manage auto-bandwidth specific system-wide properties
- 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
- 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
- Point-to-Multipoint (P2MP) TE parameters

```
module: ietf-te
   +--rw te!
   |  +--rw globals
   |  |  +--rw interface-named-admin-groups* [name]
   {ietf-te-types:extended-admin-groups,
   ietf-te-types:named-extended-admin-groups}?
   |  |  |  +--rw name     string
   |  |  |  +--rw group?   ietf-te-types:admin-groups
   |  |  +--rw interface-named-srlgs* [name]
   {ietf-te-types:named-srlg-groups}?
   |  |  |  +--rw name     string
   |  |  |  +--rw group?   ietf-te-types:srlg
   |  |  +--rw explicit-paths* [name]
   |  |  |  +--rw name                       string
   |  |  |  +--rw explicit-route-objects* [index]
   |  |  |     +--rw index                     uint8
   |  |  |     +--rw explicit-route-object
   |  |  |     |  +--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 label_value?        uint32
|  |  |     +--rw explicit-route-usage?    identityref
|  |  +--rw path-named-constraints* [name]
{ietf-te-types:named-path-constraints}?
|  |     +--rw name                string
|  |     +--rw path-constraints
|  |        +--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 path-affinities
{ietf-te-types:named-path-affinities}?
|  |           |  +--rw (style)?
|  |           |     +--:(values)
|  |           |     |  +--rw value?         uint32
|  |           |     |  +--rw mask?          uint32
|  |           |     +--:(named)
|  |           |        +--rw constraints* [usage]
|  |           |           +--rw usage         identityref
|  |           |           +--rw constraint
|  |           |              +--rw affinity-names* [name]
|  |           |                 +--rw name    string
|  |           +--rw path-srlgs
|  |              +--rw (style)?
|  |                 +--:(values)
|  |                 |  +--rw usage?         constraint-usage-type
|  |                 |  +--rw values*        srlg
|  |                 +--:(named)
|  |                    +--rw constraints* [usage]
|  |                       +--rw usage         constraint-usage-type
|  |                       +--rw constraint
|  |                          +--rw srlg-names* [name]
|  |                             +--rw name    string
```

3.1.2. Interface Configuration Data

This branch of the data model covers configurations elements relevant to TE interfaces.

Examples of such configuration items are TE interface's:

- 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 named-admin-groups* [named-admin-group]
   {ietf-te-types:extended-admin-groups,
   ietf-te-types:named-extended-admin-groups}?
   |  |     |  +--rw named-admin-group    leafref
   |  |     +--rw named-srlgs* [named-srlg]
   {ietf-te-types:named-srlg-groups}?
   |  |     |  +--rw named-srlg    leafref
   |  |     +--rw switching-capabilities* [switching-capability]
   |  |     |  +--rw capability?   identityref
   |  |     |  +--rw encoding?     identityref
   |  |     +--rw te-metric?                ietf-te-types:te-metric
   |  |     +--rw affinities
   |  |     |  +--rw (type)?
   |  |     |     +--:(admin-groups)
   |  |     |     |  +--rw admin-group?            admin-group
   |  |     |     +--:(extended-admin-groups) {extended-admin-groups}?
   |  |     |        +--rw extended-admin-group?   extended-admin-group
   |  |     +--rw srlgs
   |  |     |  +--rw (type)?
```

```
|  |     |     +--:(srlg-name)
|  |     |     |  +--rw names* [name]
|  |     |     |     +--rw name    string
|  |     |     +--:(srlg-value)
|  |     |        +--rw values* [value]
|  |     |           +--rw value    uint32
|  |     +--rw (bc-model-type)?
|  |     |  +--:(bc-model-rdm)
|  |     |  |  +--rw bc-model-rdm
|  |     |  |     +--rw bandwidth-psc-constraints
|  |     |  |        +--rw maximum-reservable?   uint32
|  |     |  |        +--rw bc-value*             uint32
|  |     |  +--:(bc-model-mam)
|  |     |  |  +--rw bc-model-mam
|  |     |  |     +--rw bandwidth-psc-constraints
|  |     |  |        +--rw maximum-reservable?   uint32
|  |     |  |        +--rw bc-value*             uint32
|  |     |  +--:(bc-model-mar)
|  |     |     +--rw bc-model-mar
|  |     |        +--rw bandwidth-psc-constraints
|  |     |           +--rw maximum-reservable?   uint32
|  |     |           +--rw bc-value*             uint32
|  |     +--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
|  |     +--rw fast-reroute-backups {ietf-te-types:frr-te}?
|  |        +--rw backup-capacity
|  |        |  +--rw capacity?   uint32
|  |        |  +--rw type?       uint32
|  |        +--rw (type)?
|  |           +--:(static-tunnel)
|  |           |  +--rw name?              leafref
|  |           |  +--rw static-backups* [backup-tunnel]
|  |           |     +--rw backup-tunnel    leafref
|  |           +--:(auto-tunnel)
|  |              +--rw auto-backup!
|  |                 +--rw method?            identityref
```

```
    |  |                +--rw protection?         identityref
    |  |                +--rw path-computation?   identityref
```

3.1.3. Tunnel Configuration Data

This branch of the model covers configuration items relevant TE tunnels. Examples of such configuration items are TE tunnel's:

- Name
- Admin-state
- Type (such as P2P, P2MP)
- Primary and secondary paths
- Routing usage (auto-route announce, forwarding adjacency)
- Policy based routing (PBR) parameters

```
module: ietf-te
   +--rw te!
   |  +--rw tunnels
   |     +--rw tunnel* [name type]
   |        +--rw name                    string
   |        +--rw type                    identityref
   |        +--rw identifier?             uint16
   |        +--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 path-properties
   |           |     |  +--rw path-named-constraint?   leafref
   {ietf-te-types:named-path-constraints}?
   |           |     |  +--rw path-constraints
   |           |     |  |  +--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 path-affinities
   {ietf-te-types:named-path-affinities}?
   |           |     |  |     |  +--rw (style)?
   |           |     |  |     |     +--:(values)
   |           |     |  |     |     |  +--rw value?         uint32
   |           |     |  |     |     |  +--rw mask?          uint32
   |           |     |  |     |     +--:(named)
   |           |     |  |     |        +--rw constraints* [usage]
   |           |     |  |     |           +--rw usage identityref
   |           |     |  |     |           +--rw constraint
   |           |     |  |     |             +--rw affinity-names* [name]
   |           |     |  |     |                +--rw name    string
   |           |     |  |     +--rw path-srlgs
   |           |     |  |        +--rw (style)?
   |           |     |  |           +--:(values)
   |           |     |  |           |  +--rw usage? const-usage-type
   |           |     |  |           |  +--rw values* srlg
   |           |     |  |           +--:(named)
   |           |     |  |             +--rw constraints* [usage]
   |           |     |  |              +--rw usage constraint-usage-type
```

```
   |           |     |  |               +--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 path-properties
   |           |           +--rw path-named-constraint?   leafref
   {ietf-te-types:named-path-constraints}?
   |           |           +--rw path-constraints
   |           |           |  +--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 path-affinities
   {ietf-te-types:named-path-affinities}?
   |           |           |     |  +--rw (style)?
   |           |           |     |     +--:(values)
   |           |           |     |     |  +--rw value?         uint32
   |           |           |     |     |  +--rw mask?          uint32
   |           |           |     |     +--:(named)
   |           |           |     |        +--rw constraints* [usage]
   |           |           |     |         +--rw usage identityref
   |           |           |     |         +--rw constraint
   |           |           |     |           +--rw affinity-names* [name]
   |           |           |     |               +--rw name string
   |           |           |     +--rw path-srlgs
   |           |           |        +--rw (style)?
   |           |           |           +--:(values)
   |           |           |           | +--rw usage? const-usage-type
   |           |           |           | +--rw values* srlg
   |           |           |           +--:(named)
   |           |           |              +--rw constraints* [usage]
   |           |           |                +--rw usage const-usage-type
   |           |           |                +--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 path-properties
   |                    |  +--rw path-named-constraint?   leafref
   {ietf-te-types:named-path-constraints}?
   |                    |  +--rw path-constraints
   |                    |  |  +--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 path-affinities
   {ietf-te-types:named-path-affinities}?
   |                    |  |     |  +--rw (style)?
   |                    |  |     |     +--:(values)
   |                    |  |     |     |  +--rw value?         uint32
   |                    |  |     |     |  +--rw mask?          uint32
   |                    |  |     |     +--:(named)
   |                    |  |     |        +--rw constraints* [usage]
   |                    |  |     |         +--rw usage identityref
   |                    |  |     |         +--rw constraint
   |                    |  |     |          +--rw affinity-names* [name]
   |                    |  |     |                +--rw name    string
   |                    |  |     +--rw path-srlgs
   |                    |  |        +--rw (style)?
   |                    |  |           +--:(values)
   |                    |  |           |  +--rw usage? const-usage-type
   |                    |  |           |  +--rw values*        srlg
   |                    |  |           +--:(named)
   |                    |  |              +--rw constraints* [usage]
   |                    |  |                +--rw usage const-usage-type
   |                    |  |                +--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 path-properties
   |                          +--rw path-named-constraint?   leafref
   {ietf-te-types:named-path-constraints}?
   |                          +--rw path-constraints
   |                          |  +--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 path-affinities
   {ietf-te-types:named-path-affinities}?
   |                          |     |  +--rw (style)?
   |                          |     |    +--:(values)
   |                          |     |    |  +--rw value?         uint32
   |                          |     |    |  +--rw mask?          uint32
   |                          |     |    +--:(named)
   |                          |     |     +--rw constraints* [usage]
   |                          |     |      +--rw usage identityref
   |                          |     |      +--rw constraint
   |                          |     |       +--rw affinity-names* [name]
   |                          |     |            +--rw name    string
   |                          |     +--rw path-srlgs
   |                          |        +--rw (style)?
   |                          |          +--:(values)
   |                          |          | +--rw usage? const-usage-type
   |                          |          | +--rw values*        srlg
   |                          |          +--:(named)
   |                          |            +--rw constraints* [usage]
   |                          |             +--rw usage const-usage-type
   |                          |              +--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
```

3.2. TE State Data

Following subsections provide overview of the parts of the model that hold state data.

3.2.1. Global State Data

This branch of the model covers system-wide state for various TE features. Examples of such states are:

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

3.2.2. Interface State Data

This branch of the model covers state data for for TE interfaces. Examples of such states are TE interface's:

- State information
  - UP/Down: when and reason
- 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

3.2.3. Tunnel State Data

This module defines operational state data for TE tunnels. Examples of such tunnel states are:

- Tunnel creation information
- State information

    * Up/Down: when and reason

  o Traffic counters

  o History of events

3.3. TE RPC data

The execution model facilitates issuing commands to a router and optionally returning responses.

3.3.1. 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:

  o Clear global TE statistics of various features

3.3.2. Interface RPC Data

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

  o Clear TE statistics for all or for individual links

  o Trigger immediate flooding for all TE interfaces

3.3.3. 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:

  o Clear statistics for all or for individual tunnels

3.4. TE Notification Data

Following subsections provide overview of parts of the model pertaining to notification data.

3.4.1. Global Notifications Data

This branch of the model covers system-wide notifications data. The global TE events notification model uses configuration data for registration. 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

3.4.2. 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:

- Link creation and deletion
- Link state transition
- (Soft) preemption trigger
- Fast reroute activation

3.4.3. 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 YANG Module

```
<CODE BEGINS>file "ietf-te-types@2015-03-22.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 2015-03-22 {
            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-object {
        description
            "Explicit Route Object grouping";
        container explicit-route-object {
            description
                "An explicit route describes as a list of groups of
                nodes along the explicit route.";
            reference "RFC3209: Extensions to RSVP for LSP Tunnels";

            choice type {
                description
                    "The Explicit Route Object type";
                case ipv4-address {
                    description
                        "IPv4 address Explicit Route Object";
                    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 Object";
                    }
                    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 Object";
                        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-object {
        description
            "The record route object grouping";
        container record-route-object {
            description
                "Describes a record route object";
            choice type {
                description
                    "The record route object type";
                case ipv4-address {
                    leaf 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 prefix-length {
                        type uint8;
                        description
                            "Length in bits of the IPv4 prefix";
                    }
                }
                case ipv6-address {
                    leaf 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 prefix-length {
                        type uint8;
                        description
                            "Length in bits of the IPv4 prefix";
                    }
                }
                case label {
                    leaf value {
                        type uint32;
                        description "the label value";
                    }
                    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 uint32;
        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.";
    }

    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 path-affinities;
                uses path-srlgs;
            }
        }

        grouping path-affinities {
            description
                "Path affinities grouping";
            container 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 path-srlgs {
        description
            "Path SRLG properties grouping";
        container 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";
                }
            }
        }
    }

    /* TE interface attribute properties */
    grouping interface-switching-cap {
        description
            "TE interface switching capabilities";
        list switching-capabilities {
            key "switching-capability";
            description
                "List of interface capabilities for this interface";
```

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

    grouping interface-affinities {
        description
            "TE interface affinities grouping";
        container affinities {
            description
                "TE interface affinities container";
            choice type {
                description
                    "TE interface administrative groups
                     representation type";
                case admin-groups {
                    description
                        "Administrative group/Resource
                         class/Color.";
                    leaf admin-group {
                        type admin-group;
                        description
                            "TE interface administrative group";
                    }
                }
                case extended-admin-groups {
                    if-feature extended-admin-groups;
                    description
                        "Extended administrative group/Resource
                         class/Color.";
                    leaf extended-admin-group {
                        type extended-admin-group;
                        description
                            "TE interface extended administrativei
                             group";
                    }
```

```
                }
            }
        }
    }

    grouping interface-srlgs {
        description "TE interface SRLG grouping";
        container srlgs {
            description "TE interface SRLG container";
            choice type {
                description
                    "SRLG representation type";
                case srlg-name {
                    list names {
                        key "name";
                        description "List of SRLG names that
                            this link is part of.";
                        leaf name {
                             type string;
                             description
                                "Name associated with the SRLG";
                        }
                    }
                }
                case srlg-value {
                    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";
                        }
                    }
                }
            }
        }
    }
}
<CODE ENDS>
```

Figure 3: TE basic types YANG module

```
<CODE BEGINS> file "ietf-te-psc-types@2015-03-22.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 2015-03-22 {
           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 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 4: TE packet/MPLS specific types YANG module

```
    <CODE BEGINS> file "ietf-te@2015-03-22.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 2015-03-22 {
            description
                "Initial revision.";
            reference "TBD";
        }
```

```
    grouping interface-attributes {
        description "Interface TE properties grouping.";
        leaf te-metric {
            type ietf-te-types:te-metric;
            description "Interface TE link metric.";
        }
        uses ietf-te-types:interface-affinities;
        uses ietf-te-types:interface-srlgs;
        uses ietf-te-psc-types:bandwidth-psc-reservable;
    }

    /* TE interface flooding parameters */
    grouping flooding-parameters {
        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 auto-backup {
        description
            "Auto-tunnel backup properties grouping.";
        container auto-backup {
            presence "Enable auto-tunnel backup feature.";
            description "Container for auto-backup features";
            leaf method {
```

```
                type identityref {
                    base ietf-te-psc-types:backup-type;
                }
                description
                    "Describes whether facility backup or 1-for-1
                    backup should be used";
            }
            leaf protection {
                type identityref {
                    base ietf-te-psc-types:backup-protection-type;
                }
                default
                    ietf-te-psc-types:backup-protection-node-link;
                description
                    "Describes whether the backup should offer
                    protection against link, node, or either";
            }
            leaf path-computation {
                type identityref {
                    base ietf-te-types:path-computation-srlg-type;
                }
                description
                    "FRR backup computation type";
            }
        }
    }

    grouping fast-reroute-backups {
        description
            "FRR backup tunnels";
        container fast-reroute-backups {
            if-feature ietf-te-types:frr-te;
            description
                "FRR backup tunnel container";
            container backup-capacity {
                description
                    "Limits the aggregate amount of primary
                     protected LSP bandwidth that this backup
                     tunnel may protect";
                leaf capacity {
                    type uint32;
                    description
                        "Maximum bandwidth this facility backup
                        is allowed to protect";
                }
                leaf type {
                    type uint32;
                    description
```

```
                        "Type of primary LSP bandwidth that the
                         backup is allowed to protect.";
                    }
                }
                choice type {
                    description
                        "FRR backup tunnel type";
                    case static-tunnel {
                        leaf name {
                            type leafref {
                                path "/te/tunnels/tunnel/name";
                            }
                            description
                                "Static FRR backup tunnel name";
                        }
                        list static-backups {
                            key "backup-tunnel";
                            description
                                "List of static backup tunnels that
                                protect the TE interface.";
                            leaf backup-tunnel {
                                type leafref {
                                    path "/te/tunnels/tunnel
                                    [name = current()/../../name]/type";
                                }
                                description "FRR Backup tunnel";
                            }
                        }
                    }
                    case auto-tunnel {
                        uses auto-backup;
                    }
                }
            }
        }

        grouping path-constraints {
            description
                "Grouping of possible TE path constraints";
            container path-constraints {
                description
                    "Path contraints container";
                uses ietf-te-types:tunnel-path-selection;
            }
        }

        grouping tunnel-path-properties {
            description
```

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

```
        }
      }

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

        /*** End of Groupings ***/
        container globals {
          description
            "Configuration data model for Global System-wide
             Traffic Engineering.";

          list interface-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";
            }
          }

          list interface-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";
```

```
                    }
                }

                list 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-object;
                        leaf explicit-route-usage {
                            type identityref {
                                base ietf-te-types:route-usage-type;
                            }
                            description
                                "An IP hop action.";
                        }
                    }
                }

                list path-named-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 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.";
                }

                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/" +
                                "interface-named-admin-groups/name";
                        }
                        description
                            "A named admin-group entry";
                    }
                }

                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/globals/" +
                                "interface-named-srlgs/name";
                        }
                        description
                            "A named SRLG entry";
                    }
                }
```

```
                uses ietf-te-types:interface-switching-cap;
                uses interface-attributes;
                /* Link IGP flooding properties */
                uses flooding-parameters;
                uses fast-reroute-backups;
            }
        }

        /* 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.";
                }
                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-properties;
                                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-properties;
                                }
                            }
                        }
                        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-properties;
                            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-properties;
                            }
                        }
                    }
                }
            }
        }
    }
    /* MPLS-TE Global Operational Data */
    container global-state {
```

```
            config "false";
            description
                "State for global TE data";
        }

        /* TE Interfaces State Data */
        container interface-state {
            config "false";
            description
                "Operational data model for TE interfaces.";
        }

        /* TE Tunnel State Data */
        container tunnels-state {
            config "false";
            description "MPLS-TE tunnel operational state data.";
        }
    }

    /* 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 5: 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-17 (work in progress), March 2015.

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

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

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

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

[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

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 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.

[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