q

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

J. George
Google
L. Fang
Microsoft
E. Osborne
Level 3
R. Shakir
Jive Communications
October 19, 2015

MPLS / TE Model for Service Provider Networks
draft-openconfig-mpls-consolidated-model-02

Abstract

This document defines a framework for a YANG data model for configuring and managing label switched paths, including the signaling protocols, traffic engineering, and operational aspects based on carrier and content provider operational requirements.

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 April 21, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document describes a YANG [RFC6020] data model for MPLS and traffic engineering, covering label switched path (LSP) configuration, as well as signaling protocol configuration. The model is intended to be vendor-neutral, in order to allow operators to manage MPLS in heterogeneous environments with physical or virtual devices (routers, switches, servers, etc.) supplied by multiple vendors. The model is also intended to be readily mapped to existing implementations, to facilitate support from as large a set of routing hardware and software vendors as possible.

1.1. Goals and approach

The focus area of the model in this revision, is to set forth a framework for MPLS, with hooks into which information specific to various signaling-protocols can be added. The framework is built around functionality from a network operator perspective rather than

a signaling protocol-centric approach. For example, a traffic-engineered LSP will have configuration relating to its path computation method, regardless of whether it is signaled with RSVP-TE or with segment routing. Thus, rather than creating separate per-signaling protocol models and trying to stitch them under a common umbrella, this framework focuses on functionality, and adds signaling protocol-specific information under it where applicable.

This model does not aim to be feature complete (i.e., cover all possible aspects or features of MPLS). Rather its development is driven by examination of actual production configurations in use across a number of operator network deployments.

Configuration items that are deemed to be widely available in existing major implementations are included in the model. Those configuration items that are only available from a single implementation are omitted from the model with the expectation they will be available in companion modules that augment the current model. This allows clarity in identifying data that is part of the vendor-neutral model.

An important aspect of the model is the representation of operational state data. This draft takes the approach described in [I-D.openconfig-netmod-opstate] and models configuration and operational state together. Thus, rather than building a separate tree of operational state, the operational state and configuration data are located in parallel containers at the leaves of the data model. This approach allows easy reuse of groupings across models, as well as making it easier to correlate configuration and state.

The consolidated MPLS model encompasses the signaling protocols, label-switched paths (configuration and operational state), and generic TE attributes. The model is designed from an operational and functional perspective, rather than focusing on protocol-centric configuration. This allows protocol-independent functions to be logically separated from protocol-specific details.

One question that arises in this approach is how the consolidated model is integrated with routing instances (e.g., VRFs). This model should be considered as part of a higher level network device model which includes definitions for other routing protocols and system services. For example, in [I-D.openconfig-netmod-model-structure], VRFs and other logical instances are defined with MPLS/TE components within VRFs as appropriate. In particular, some parts of the MPLS model would be instantiated within a VRF, while other parts would have common definitions across VRFs.

Where possible, naming in the model follows conventions used in available standards documents, and otherwise tries to be self-explanatory with sufficient descriptions of the intended behavior. Similarly, configuration data value constraints and default values, where used, are based on recommendations in current standards documentation. Since implementations vary widely in this respect, this version of the model specifies only a limited set of defaults and ranges with the expectation of being more prescriptive in future versions based on actual operator use.

Note that this version of the model is a work-in-progress in several respects. Although we present a complete framework for MPLS and traffic engineering from an operational perspective, some signaling protocol configuration will be completed in future revisions. The current revision has focus on traffic engineered LSPs signaled with RSVP.

2. Model overview

The overall MPLS model is defined across several YANG modules and submodules but at a high level is organized into 4 main sections:

- global -- configuration affecting MPLS behavior which exists independently of the underlying signaling protocol or label switched path configuration.
- te-global-attributes -- configuration affecting MPLS-TE behavior which exists independently of the underlying signaling protocol or label switched path configuration.
- signaling protocols -- configuration specific to signaling protocols used to setup and manage label switched paths.
- label switched paths -- configuration specific to instantiating and managing individual label switched paths.

The top level of the model is shown in the tree view below:

```
+--rw mpls!
      +--rw global
      |     ...
      +--rw te-global-attributes
      |     ...
      +--rw signaling-protocols
      |     ...
      +--rw lsps
            ...
```

2.1. MPLS global

The global section of the framework provides configuration data for MPLS items which exist independently of an individual label switched path or signaling protocol and are applicable to the MPLS protocol itself. Items such as the depth of the label stack supported, or specific label ranges may be included here.

2.2. TE global attributes

The TE global attributes section of the framework provides configuration control for MPLS-TE items which exist independently of an individual label switched path or signaling protocol. These standalone items are applicable to the entire logical routing device, and establish fundamental configuration such as the threshold for interface bandwidth change that triggers update events into the IGP traffic engineering database (TED). Timers are also specified which determine the length of time an LSP must be present before being considered viable for forwarding use (te-lsp-install-delay), and the length of time between LSP teardown and removal of the LSP from the network element’s forwarding information base (te-lsp-cleanup-delay). Also specified are the name to value mappings of MPLS administrative groups (mpls-admin-groups) and shared risk link groups (mpls-te-srlg).

```
    +--rw te-global-attributes
         |  +--rw mpls-te-srlg
         |  |  +--rw srlg* [srlg-name]
         |  |     +--rw srlg-name       leafref
         |  |     +--rw config
         |  |     |  +--rw srlg-name?    string
         |  |     |  +--rw srlg-value?   uint32
         |  |     |  +--rw srlg-cost?    uint32
         |  |     +--ro state
         ...
         |  |     +--rw members-list* [from-address]
         |  |        +--rw from-address    leafref
         |  |        +--rw config
         |  |        |  +--rw from-address?   inet:ip-address
         |  |        |  +--rw to-address?     inet:ip-address
         ...
         |  +--rw igp-flooding-bandwidth
         |  |  +--rw config
         |  |  |  +--rw threshold-type?            enumeration
         |  |  |  +--rw delta-percentage?          oc-types:percentage
         |  |  |  +--rw threshold-specification?   enumeration
         |  |  |  +--rw up-thresholds*             oc-types:percentage
         |  |  |  +--rw down-thresholds*           oc-types:percentage
         |  |  |  +--rw up-down-thresholds*        oc-types:percentage
         |  |  +--ro state
         ...
         |  +--rw mpls-admin-groups
         |  |  +--rw admin-group* [admin-group-name]
         |  |     +--rw admin-group-name    leafref
         |  |     +--rw config
         |  |     |  +--rw admin-group-name?    string
         |  |     |  +--rw admin-group-value?   uint32
         |  |     +--rw state
         ...
         |  +--rw te-lsp-timers
         |     +--rw config
         |     |  +--rw te-lsp-install-delay?      uint16
         |     |  +--rw te-lsp-cleanup-delay?      uint16
         |     |  +--rw te-lsp-reoptimize-timer?   uint16
         |     +--ro state
         ...
```

2.3. TE interface attributes overview

The TE interface attributes section of the framework provides configuration and state related to traffic engineering such as te-metric or shared risk link group configuration.

```
        +--rw te-intf-attributes
        |  +--rw interface* [interface-name]
        |     +--rw interface-name    leafref
        |     +--rw config
        |     |  +--rw interface-name?          ocif:interface-ref
        |     |  +--rw te-metric?               uint32
        |     |  +--rw srlg* [srlg-name]
        |     |     ...
        |     |  +--rw admin-group* [admin-group-name]
        |     |     ...
        |     |  +--rw igp-flooding-bandwidth
        |     |    ...
        |     +--ro state
        ...
```

2.4. Signaling protocol overview

The signaling protocol section of the framework provides configuration elements for configuring three major methods of signaling label switched paths: RSVP-TE, segment routing, and label distribution protocol (LDP). BGP-LU will be included in a future version of this draft by definitions in the BGP model ([I-D.ietf-idr-bgp-model]) and corresponding augmentations to the MPLS model.

```
+--rw signaling-protocols
      |  +--rw rsvp-te
      |     ...
      |  +--rw segment-routing
      |     ...
      |  +--rw ldp
      |     ...
```

Configuration of RSVP-TE is centered around interfaces on the device which participate in the protocol. A key focus is to expose common RSVP-TE configuration parameters which are used to enhance scale and reliability. Items which are applicable globally in the RSVP-TE protocol such as graceful restart, soft preemption and various statistics are grouped into a global section under the protocol. RSVP neighbor and session state are also available in the RSVP section.

```
         +--rw rsvp-te
         |  |  +--rw rsvp-sessions
         |  |  |  +--rw config
         |  |  |  +--ro state
         |  |  |     +--ro rsvp-session* [source-port destination-port
                           source-address destination-address]
                     ...
         |  |  +--rw rsvp-neighbors
         |  |  |  +--rw config
         |  |  |  +--ro state
         |  |  |     +--ro rsvp-neighbor* [neighbor-address]
                     ...
         |  |  +--rw global
         |  |  |  +--rw graceful-restart
                     ...
         |  |  |  +--rw soft-preemption
                     ...
         |  |  |  +--ro statistics
         |  |  |     +--ro counters
                     ....
         |  |  +--rw interface-attributes
         |  |     +--rw interface* [interface-name]
         |  |        +--rw interface-name    leafref
                     ...
         |  |        +--rw rsvp-hellos
                     ...
         |  |        +--rw authentication
                     ...
         |  |        +--rw subscription
                     ...
         |  |        +--rw protection
                     ...
         ...
```

Containers for specifying signaling via segment routing and LDP are also present. Specific subelements will be added for those protocols, as well as for BGP labeled unicast, in the next revision.

2.5. LSP overview

This part of the framework contains LSP information. At the high level, LSPs are split into three categories: traffic-engineering-capable (constrained-path), non-traffic-engineered determined by the IGP (unconstrained-path), and hop-by-hop configured (static).

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
         |     ...
         +--rw unconstrained-path
         |     ...
         +--rw static-lsps
               ...
```

The first two categories, constrained-path and unconstrained-path are the ones for which multiple signaling protocols exist, and are organized in protocol-specific and protocol-independent sections. For example, traffic-engineered (constrained path) LSPs may be set up using RSVP-TE or segment routing, and unconstrained LSPs that follow the IGP path may be signaled with LDP or with segment routing. IGP-determined LSPs may also be signaled by RSVP but this usage is not considered in the current version of the model.

A portion of the data model for constrained path traffic-engineered LSPs signaled with RSVP is shown below. It contains configuration for named explicit paths and for tunnels. Tunnel configuration differs for p2p and p2mp LSPs. In either case, some part of the model is signaling-protocol independent. For example for a p2p LSP, attributes such as the path computation method, the constraints for the the path, the bandwidth allocated to it, and even the frequency of reoptimization are signaling-protocol independent, while other data, such as the setup and hold priorities are protocol-specific and are specified in the protocol specific part of the model.

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
       |  +--rw explicit-path* [name]
            ...
          |  +--rw tunnel* [name type]
          |     +--rw name      leafref
          |     +--rw type      leafref
          |     +--rw config
          |     |  +--rw name?                          string
          |     |  +--rw type?                          identityref
          |     |  +--rw local-id?                      union
          |     |  +--rw description?                   string
          |     |  +--rw admin-status?                  identityref
          |     |  +--rw preference?                    uint8
          |     |  +--rw metric?                        te-metric-type
          |     |  +--rw (bandwidth)?
                   ...
          |     |  +--rw protection-style-requested?    identityref
          |     |  +--rw te-lsp-reoptimize-timer?       uint16
          |     |  +--rw (signaling-specific-tunnel-attributes)?
          |     |  |  +--:(RSVP)
          |     |  |     +--rw source?                        inet:ip-address
          |     |  |     +--rw soft-preemption?               boolean
          |     |  +--rw (tunnel-type)?
          |     |     +--:(p2p)
          |     |     |  +--rw destination?                   inet:ip-address
          |     |     |  +--rw primary-paths* [name]
          |     |     |  |  +--rw name                        string
          |     |     |  |  +--rw preference?                 uint8
          |     |     |  |  +--rw path-computation-method
                                ...
          |     |     |  |  +--rw admin-groups
                                ...
          |     |     |  |  +--rw no-cspf?                    empty
          |     |     |  |  +--rw (sigaling-specific-path-attributes)?
          |     |     |  |     +--:(RSVP)
          |     |     |  |     |  +--rw setup-priority?             uint8
          |     |     |  |     |  +--rw hold-priority?              uint8
          |     |     |  |     |  +--rw retry-timer?                uint16
          |     |     |  |     +--:(SR)
          |     |     |  |        +--rw sid-selection-mode?         enumeration
          |     |     |  |        +--rw sid-protection-required?    boolean
          |     |     |  +--rw secondary-paths* [name]
                            ...
          |     +--ro state
                ...
```

Similarly, the partial model for non-traffic-engineered, or IGP-based, LSPs is shown below:

```
+--rw mpls!
      +--rw lsps
         +--rw unconstrained-path
            +--rw path-setup-protocol
               +--rw ldp!
               |     ...
               +--rw segment-routing!
                     ...
```

3. Example use cases

3.1. Traffic engineered p2p LSP signaled with RSVP

A possible scenario may be the establishment of a mesh of traffic-engineered LSPs where RSVP signaling is desired, and the LSPs use a local constrained path calculation to determine their path. These LSPs would fall into the category of a constrained-path LSP, and the tunnel type is p2p. Attributes such as metric, bandwidth or the style of protection desired are also defined at this (protocol-independent) level in the model. The path is defined to be locally-computed under the path-computation-method container, specifying the use of CSPF (use-cspf). Additional attributes for the path, such as its RSVP priorities are specified at the path level under the protocol-specific stanza.

```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
            ...
         |  +--rw tunnel* [name type]
         |     +--rw name      leafref
         |     +--rw type      leafref
         |     +--rw config
         |     |  +--rw name?                          string
         |     |  +--rw type?                          identityref
         |     |  +--rw metric?                        te-metric-type
         |     |  +--rw (bandwidth)?
                  ...
         |     |  +--rw protection-style-requested?   identityref
         |     |  +--rw te-lsp-reoptimize-timer?      uint16
                  ...
         |     |  +--rw (tunnel-type)?
         |     |     +--:(p2p)
         |     |     |  +--rw destination?                  inet:ip-address
         |     |     |  +--rw primary-paths* [name]
         |     |     |  |  +--rw name                       string
         |     |     |  |  +--rw preference?                uint8
         |     |     |  |  +--rw path-computation-method
                              ...
         |     |     |  |  +--rw admin-groups
                              ...
         |     |     |  |  +--rw no-cspf?                   empty
         |     |     |  |  +--rw (sigaling-specific-path-attributes)?
         |     |     |  |     +--:(RSVP)
         |     |     |  |     |  +--rw setup-priority?            uint8
         |     |     |  |     |  +--rw hold-priority?             uint8
         |     |     |  |     |  +--rw retry-timer?               uint16
         |     |     |  |     +--:(SR)
         |     |     |  |        +--rw sid-selection-mode?        enumeration
         |     |     |  |        +--rw sid-protection-required?   boolean
         |     |     |  +--rw secondary-paths* [name]
                        ...
         |     +--ro state
```

3.2. Traffic engineered LSP signaled with SR

A possible scenario may be the establishment of disjoint paths in a network where there is no requirement for per-LSP state to be held on midpoint nodes within the network, or RSVP-TE is unsuitable (as described in [I-D.ietf-spring-segment-routing-mpls] and [I-D.shakir-rtgwg-sr-performance-engineered-lsps]). Such LSPs fall in the constrained-path category. Similar to any other traffic engineered LSPs, the path computation method must be specified. Path

attributes, such as the as lsp- placement-constraints (expressed as administrative groups) or metric must be defined. Finally, the path must be specified in a signaling- protocol specific manner appropriate for SR. The same configuration elements from the tree above apply in this case, except that path setup is done by the head-end by building a label stack, rather than signaled.

3.3. IGP-congruent LDP-signaled LSP

A possible scenario may be the establishment of a full mesh of LSPs. When traffic engineering is not an objective, no constraints are placed on the end-to-end path, and the best- effort path can be setup using LDP signaling simply for label distribution. The LSPs follow IGP-computed paths, and fall in the unconstrained-path category in the model. Protocol-specific configuration pertaining to the signaling protocol used, such as the FEC definition and metrics assigned are in the path- setup-protocol portion of the model.

The relevant part of the model for this case is shown below:

```
  +--rw mpls!
      +--rw lsps
         +--rw unconstrained-path
            +--rw path-setup-protocol
               +--rw ldp!
                  +--rw tunnel
                     +--rw tunnel-type?   mplst:tunnel-type
                     +--rw ldp-type?      enumeration
                     +--rw p2p-lsp
                     |  +--rw fec-address*   inet:ip-prefix
                     +--rw p2mp-lsp
                     +--rw mp2mp-lsp
```

A common operational issue encountered when using LDP is traffic blackholing under the following scenario: when an IGP failure occurs, LDP is not aware of it as these are two protocols running independently, resulting in traffic blackholing at the IGP failure point even though LDP is up and running. LDP-IGP synchronization [RFC5443] can be used to cost out the IGP failing point/segment to avoid the blackholing issue. The LDP-IGP synchronization function will be incorporated in a future version of this document.

Note that targeted LDP sessions are not discussed in this use case, and will be incorporated as a separate use case in a future version of this document.

4. Security Considerations

MPLS configuration has a significant impact on network operations, and as such any related protocol or model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer BGP configuration and operational data. Note that use of alternate transport or data encoding (e.g., JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

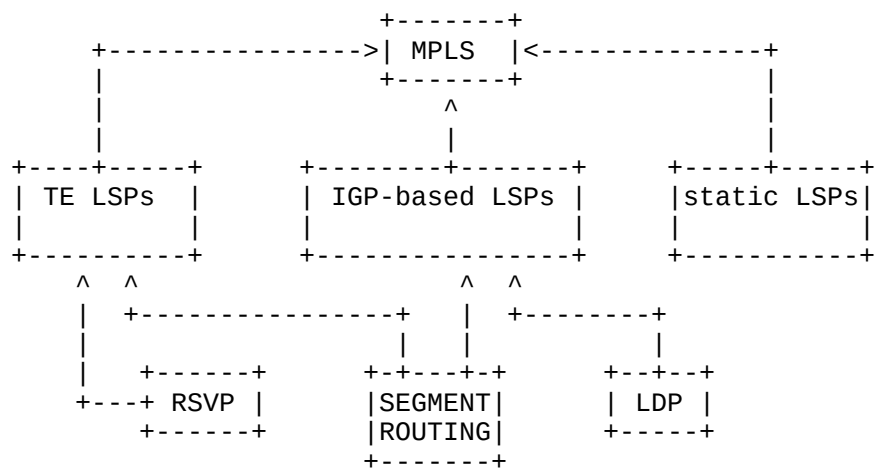
Most of the data elements in the configuration model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

5. IANA Considerations

This YANG data model and the component modules currently use a temporary ad-hoc namespace. If and when it is placed on redirected for the standards track, an appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The MPLS YANG modules will be registered in the "YANG Module Names" registry [RFC6020].

6. YANG modules

The modules and submodules comprising the MPLS configuration and operational model are currently organized as depcited below.



The base MPLS module includes submodules describing the three different types of support LSPs, i.e., traffic-engineered (constrained-path), IGP congruent (unconstrained-path), and static. The signaling protocol specific parts of the model are described in separate modules for RSVP, segment routing, and LDP. As mentioned earlier, support for BGP labeled unicast is also planned in a future revision.

A module defining various reusable MPLS types is included, and these modules also make use of the standard Internet types, such as IP addresses, as defined in RFC 6991 [RFC6991].

6.1. MPLS base modules

<CODE BEGINS> file openconfig-mpls.yang

```
module openconfig-mpls {

  yang-version "1";

  // namespace
  namespace "http://openconfig.net/yang/mpls";

  prefix "mpls";


  // import some basic types
  import openconfig-mpls-rsvp { prefix rsvp; }
  import openconfig-mpls-sr { prefix sr; }
  import openconfig-mpls-ldp { prefix ldp; }
  import openconfig-types { prefix oc-types; }
  import openconfig-interfaces { prefix ocif; }

  // include submodules
  include openconfig-mpls-te;
  include openconfig-mpls-igp;
  include openconfig-mpls-static;


  // meta
  organization "OpenConfig working group";

  contact
    "OpenConfig working group
    netopenconfig@googlegroups.com";

  description
    "This module provides data definitions for configuration of
```

Multiprotocol Label Switching (MPLS) and associated protocols for signaling and traffic engineering.

RFC 3031: Multiprotocol Label Switching Architecture

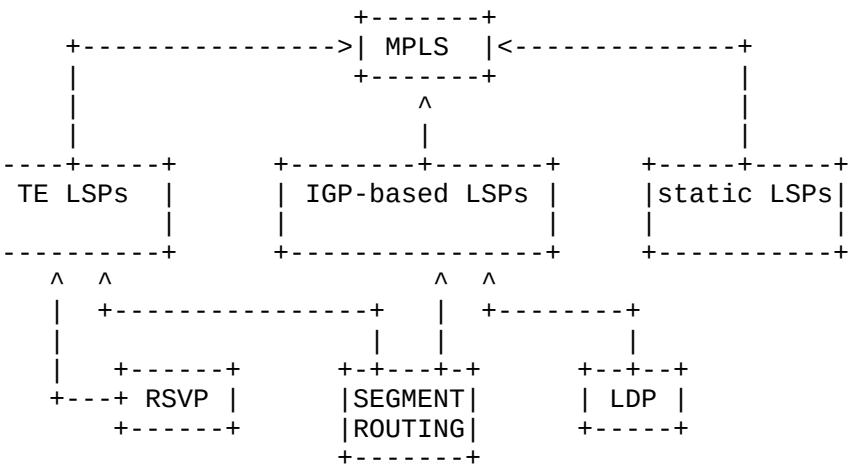
The MPLS / TE data model consists of several modules and submodules as shown below. The top-level MPLS module describes the overall framework. Three types of LSPs are supported:

i) traffic-engineered (or constrained-path)

ii) IGP-congruent (LSPs that follow the IGP path)

iii) static LSPs which are not signaled

The structure of each of these LSP configurations is defined in corresponding submodules. Companion modules define the relevant configuration and operational data specific to key signaling protocols used in operational practice.



";

revision "2015-10-14" {
description
"Work in progress";
reference "TBD";
}

// extension statements

// feature statements

```
     // identity statements


     // grouping statements

     grouping mpls-admin-group_config {
       description
         "configuration data for MPLS link admin groups";

         leaf admin-group-name {
           type string;
           description "name for mpls admin-group";
         }

         leaf admin-group-value {
           type uint32;
           description "value for mpls admin-group";
         }

     }

     grouping mpls-admin-groups-top {

       description "top-level mpls admin-groups config
         and state containers";

       container mpls-admin-groups {
         description
           "Top-level container for admin-groups configuration
           and state";

         list admin-group {
           key admin-group-name;
           description "configuration of value to name mapping
             for mpls affinities/admin-groups";

           leaf admin-group-name {
             type leafref {
               path "../mpls:config/mpls:admin-group-name";
             }
             description
               "name for mpls admin-group";
           }
           container config {
             description "Configurable items for admin-groups";
             uses mpls-admin-group_config;
           }
```

```
          container state {
            description "Operational state for admin-groups";
            uses mpls-admin-group_config;
          }
        }
      }
    }

    grouping mpls-te-igp-flooding-bandwidth_config {
      description
        "Configurable items for igp flooding bandwidth
        threshold configuration.";
      leaf threshold-type {
        type enumeration {
          enum DELTA {
            description "DELTA indicates that the local
              system should flood IGP updates when a
              change in reserved bandwidth >= the specified
              delta occurs on the interface.";
          }
          enum THRESHOLD-CROSSED {
            description "THRESHOLD-CROSSED indicates that
              the local system should trigger an update (and
              hence flood) the reserved bandwidth when the
              reserved bandwidth changes such that it crosses,
              or becomes equal to one of the threshold values.";
          }
        }
        description
          "The type of threshold that should be used to specify the
          values at which bandwidth is flooded. DELTA indicates that
          the local system should flood IGP updates when a change in
          reserved bandwidth >= the specified delta occurs on the
          interface. Where THRESHOLD-CROSSED is specified, the local
          system should trigger an update (and hence flood) the
          reserved bandwidth when the reserved bandwidth changes such
          that it crosses, or becomes equal to one of the threshold
          values";
      }

      leaf delta-percentage {
        when "../threshold-type = 'DELTA'" {
          description
            "The percentage delta can only be specified when the
            threshold type is specified to be a percentage delta of
            the reserved bandwidth";
        }
        type oc-types:percentage;
```

```
          description
            "The percentage of the maximum-reservable-bandwidth
            considered as the delta that results in an IGP update
            being flooded";
        }

        leaf threshold-specification {
          when "../threshold-type = 'THRESHOLD-CROSSED'" {
            description
              "The selection of whether mirrored or separate threshold
              values are to be used requires user specified thresholds to
              be set";
          }
          type enumeration {
            enum MIRRORED-UP-DOWN {
              description
                "MIRRORED-UP-DOWN indicates that a single set of
                threshold values should be used for both increasing
                and decreasing bandwidth when determining whether
                to trigger updated bandwidth values to be flooded
                in the IGP TE extensions.";
            }
            enum SEPARATE-UP-DOWN {
              description
                "SEPARATE-UP-DOWN indicates that a separate
                threshold values should be used for the increasing
                and decreasing bandwidth when determining whether
                to trigger updated bandwidth values to be flooded
                in the IGP TE extensions.";
            }
          }
          description

            "This value specifies whether a single set of threshold
            values should be used for both increasing and decreasing
            bandwidth when determining whether to trigger updated
            bandwidth values to be flooded in the IGP TE extensions.
            MIRRORED-UP-DOWN indicates that a single value (or set of
            values) should be used for both increasing and decreasing
            values, where SEPARATE-UP-DOWN specifies that the increasing
            and decreasing values will be separately specified";
        }

        leaf-list up-thresholds {
          when "../threshold-type = 'THRESHOLD-CROSSED'" +
            "and ../threshold-specification = 'SEPARATE-UP-DOWN'" {
              description
                "A list of up-thresholds can only be specified when the
```

```
              bandwidth update is triggered based on crossing a
              threshold and separate up and down thresholds are
              required";
          }
        type oc-types:percentage;
        description
          "The thresholds (expressed as a percentage of the maximum
          reservable bandwidth) at which bandwidth updates are to be
          triggered when the bandwidth is increasing.";
      }

      leaf-list down-thresholds {
        when "../threshold-type = 'THRESHOLD-CROSSED'" +
          "and ../threshold-specification = 'SEPARATE-UP-DOWN'" {
            description
              "A list of down-thresholds can only be specified when the
              bandwidth update is triggered based on crossing a
              threshold and separate up and down thresholds are
              required";
          }
        type oc-types:percentage;
        description
          "The thresholds (expressed as a percentage of the maximum
          reservable bandwidth) at which bandwidth updates are to be
          triggered when the bandwidth is decreasing.";
      }

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

    grouping mpls-te-igp-flooding-bandwdith-if {
      description "Interface-level group for traffic engineering
      database flooding options options";
```

```
        container igp-flooding-bandwidth {
         description "Interface bandwidth change percentages
         that trigger update events into the IGP traffic
         engineering database (TED)";
         uses  mpls-te-igp-flooding-bandwidth_config;
        }
    }


    grouping  mpls-te-igp-flooding-bandwidth {
      description "Top level group for traffic engineering
        database flooding options";
      container igp-flooding-bandwidth {
        description "Interface bandwidth change percentages
          that trigger update events into the IGP traffic
          engineering database (TED)";
        container config {
          description "Configuration parameters for TED
            update threshold ";
          uses  mpls-te-igp-flooding-bandwidth_config;
        }
        container state {
          config false;
          description "State parameters for TED update threshold ";
          uses  mpls-te-igp-flooding-bandwidth_config;
        }
      }
    }


    grouping te_lsp_delay_config {
      description "Group for the timers goerning the delay
      in installation and cleanup of TE LSPs";

      leaf te-lsp-install-delay {
        type uint16 {
          range 0..3600;
        }
        units seconds;
        description "delay the use of newly installed te lsp for a
        specified amount of time.";
      }

      leaf te-lsp-cleanup-delay {
        type uint16;
        units seconds;
        description "delay the removal of old te lsp for a specified
        amount of time";
```

```
      }
    }

    grouping te-interface-attributes-top {
      description
        "Top level grouping for attributes
        for TE interfaces.";

      list interface {
        key interface-name;
        description "List of TE interfaces";

        leaf interface-name {
          type leafref {
            path "../config/interface-name";
            require-instance true;
          }
          description "The interface name";
        }

        container config {
          description
            "Configuration parameters related to TE interfaces:";
          uses te-interface-attributes-config;
        }

        container state {
          config false;
          description "State parameters related to TE interfaces";
          uses te-interface-attributes-config;
        }
      }
    }

    grouping te-interface-attributes-config {
      description "global level definitions for interfaces
        on which TE is run";

      leaf interface-name {
        type ocif:interface-ref;
        description "reference to interface name";
      }

      leaf te-metric {
        type uint32;
        description "TE specific metric for the link";
      }
```

```
        list srlg {
          key srlg-name;
          description "list of shared risk link groups on the
            interface";
          leaf srlg-name {
            type string;
            description "The SRLG group identifier";
          }
        }

        list admin-group {
          key admin-group-name;
          description "list of admin groups on the
            interface";
          leaf admin-group-name {
            type string;
            description "The admin group identifier";
          }
        }

        uses  mpls-te-igp-flooding-bandwdith-if;

      }

      grouping mpls-te-lsp-timers {
        description
          "Grouping for traffic engineering timers";
        container te-lsp-timers {
          description
          "definition for delays associated with setup
          and cleanup of TE LSPs";

          container config {
            description
            "Configuration parameters related
            to timers for TE LSPs";

            uses te_lsp_delay_config;
            uses te-tunnel-reoptimize_config;
          }
          container state {
            config false;
            description "State related to timers for TE LSPs";

            uses te_lsp_delay_config;
            uses te-tunnel-reoptimize_config;
          }
        }
```

```
     }

   container mpls {
       presence "top-level container for MPLS config and operational
       state";

       description "Anchor point for mpls configuration and operational
       data";

       container global {
         // entropy label support, label ranges will be added here.
        description "general mpls configuration applicable to any
        type of LSP and signaling protocol - label ranges,
        entropy label supportmay be added here";
       }

       container te-global-attributes {
         description "traffic-engineering global attributes";
         uses mpls-te-srlg-top;
         uses mpls-te-igp-flooding-bandwidth;
         uses mpls-admin-groups-top;
         uses mpls-te-lsp-timers;
       }

       container te-intf-attributes {
         description "traffic engineering attributes specific
         for interfaces";
         uses te-interface-attributes-top;
       }

       container signaling-protocols {
         description "top-level signaling protocol configuration";

         uses rsvp:rsvp-global;
         uses sr:sr-global;
         uses ldp:ldp-global;
       }

       container lsps {
         description "LSP definitions and configuration";

         container constrained-path {
           description "traffic-engineered LSPs supporting different
           path computation and signaling methods";
           uses explicit-paths-top;
           uses te-tunnels-top;

         }
```

```
        container unconstrained-path {
          description "LSPs that use the IGP-determined path, i.e., non
          traffic-engineered, or non constrained-path";

          uses igp-lsp-common;
          uses igp-lsp-setup;

        }

        container static-lsps {
          description "statically configured LSPs, without dynamic
            signaling";

          uses static-lsp-main;
        }
      }
    }

    // augment statements

    // rpc statements

    // notification statements
  }
```

<CODE ENDS>

<CODE BEGINS> file openconfig-mpls-types.yang

```
  module openconfig-mpls-types {

    yang-version "1";

    // namespace
    namespace "http://openconfig.net/yang/mpls-types";

    prefix "mplst";

    // meta
    organization "OpenConfig working group";

    contact
      "OpenConfig working group
      netopenconfig@googlegroups.com";

    description
      "General types for MPLS / TE data model";
```

```
     revision "2015-10-04" {
       description
         "Work in progress";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // using identities rather than enum types to simplify adding new
     // signaling protocols as they are introduced and supported
     identity path-setup-protocol {
       description "base identity for supported MPLS signaling
       protocols";
     }

     identity path-setup-rsvp {
       base path-setup-protocol;
       description "RSVP-TE signaling protocol";
     }

     identity path-setup-sr {
       base path-setup-protocol;
       description "Segment routing";
     }

     identity path-setup-ldp {
       base path-setup-protocol;
       description "lDP - RFC 5036";
     }


     identity protection-type {
       description "base identity for protection type";
     }

     identity unprotected {
       base protection-type;
       description "no protection is desired";
     }

     identity link-protection-requested {
       base protection-type;
       description "link protection is desired";
     }
```

```
identity link-node-protection-requested {
  base protection-type;
  description "node and link protection are both desired";
}

identity lsp-role {
  description
    "Base identity for describing the role of
     label switched path at the current node";
}

identity INGRESS {
  base "lsp-role";
  description
    "Label switched path is an ingress (headend)
     LSP";
}

identity EGRESS {
  base "lsp-role";
  description
    "Label switched path is an egress (tailend)
     LSP";
}

identity TRANSIT {
  base "lsp-role";
  description
    "Label switched path is a transit LSP";
}


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

identity P2P {
  base tunnel-type;
  description
    "TE point-to-point tunnel type.";
}

identity P2MP {
  base tunnel-type;
  description
    "TE point-to-multipoint tunnel type.";
```

```
     }


     identity lsp-oper-status {
       description
         "Base identity for LSP operational status";
     }

     identity DOWN {
       base "lsp-oper-status";
       description
         "LSP is operationally down or out of service";
     }

     identity UP {
       base "lsp-oper-status";
       description
         "LSP is operationally active and available
          for traffic.";
     }

     identity tunnel-admin-status {
       description
         "Base identity for tunnel administrative status";
     }

     identity ADMIN_DOWN {
       base "tunnel-admin-status";
       description
         "LSP is administratively down";
     }

     identity ADMIN_UP {
       base "tunnel-admin-status";
       description
         "LSP is administratively up";
     }

     // typedef statements
     typedef mpls-label {
       type union {
         type uint32 {
           range 16..1048575;
         }
         type enumeration {
           enum IPV4_EXPLICIT_NULL {
             value 0;
             description "valid at the bottom of the label stack,
```

```
            indicates that stack must be popped and packet forwarded
            based on IPv4 header";
          }
          enum ROUTER_ALERT {
            value 1;
            description "allowed anywhere in the label stack except
            the bottom, local router delivers packet to the local CPU
            when this label is at the top of the stack";
          }
          enum IPV6_EXPLICIT_NULL {
            value 2;
            description "valid at the bottom of the label stack,
            indicates that stack must be popped and packet forwarded
            based on IPv6 header";
          }
          enum IMPLICIT_NULL {
            value 3;
            description "assigned by local LSR but not carried in
            packets";
          }
          enum ENTROPY_LABEL_INDICATOR {
            value 7;
            description "Entropy label indicator, to allow an LSR
            to distinguish between entropy label and applicaiton
            labels RFC 6790";
          }
        }
      }
      description "type for MPLS label value encoding";
      reference "RFC 3032 - MPLS Label Stack Encoding";
    }

    typedef tunnel-type {
      type enumeration {
        enum P2P {
          description "point-to-point label-switched-path";
        }
        enum P2MP {
          description "point-to-multipoint label-switched-path";
        }
        enum MP2MP {
          description "multipoint-to-multipoint label-switched-path";
        }
      }
      description "defines the tunnel type for the LSP";
      reference
        "RFC 6388 - Label Distribution Protocol Extensions for
        Point-to-Multipoint and Multipoint-to-Multipoint Label Switched
```

```
         Paths
         RFC 4875 - Extensions to  Resource Reservation Protocol
         - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE
         Label Switched Paths (LSPs)";
     }

     typedef bandwidth-kbps {
       type uint64;
       units kbps;
     }

     typedef bandwidth-mbps {
       type uint64;
       units mbps;
     }

     typedef bandwidth-gbps {
       type uint64;
       units gbps;
     }

     // grouping statements

     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                   <CODE ENDS>
```

6.2. MPLS LSP submodules

```
                   <CODE BEGINS> file openconfig-mpls-te.yang
   submodule openconfig-mpls-te {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }
```

```
     // import some basic types
     import ietf-inet-types { prefix inet; }
     import openconfig-mpls-rsvp { prefix rsvp; }
     import openconfig-mpls-sr { prefix sr; }
     import openconfig-mpls-types {prefix mplst; }
     import openconfig-types { prefix oc-types; }
     import ietf-yang-types { prefix yang; }


     // meta
     organization "OpenConfig working group";

     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Configuration related to constrained-path LSPs and traffic
       engineering.  These definitions are not specific to a particular
       signaling protocol or mechanism (see related submodules for
       signaling protocol-specific configuration).";

     revision "2015-10-04" {
       description
         "Work in progress";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // using identities for path comp method, though enums may also
     // be appropriate if we decided these are the primary computation
     // mechanisms in future.
     identity path-computation-method {
       description
        "base identity for supported path computation
         mechanisms";
     }

     identity locally-computed {
       base path-computation-method;
       description
         "indicates a constrained-path LSP in which the
         path is computed by the local LER";
```

```
    }

    identity externally-queried {
      base path-computation-method;
      description
       "constrained-path LSP in which the path is
        obtained by querying an external source, such as a PCE server";
    }

    identity explicitly-defined {
      base path-computation-method;
      description
       "constrained-path LSP in which the path is
        explicitly specified as a collection of strict or/and loose
        hops";
    }

    // typedef statements

    typedef mpls-hop-type {
      type enumeration {
        enum LOOSE {
          description "loose hop in an explicit path";
        }
        enum STRICT {
          description "strict hop in an explicit path";
        }
      }
      description
       "enumerated type for specifying loose or strict
        paths";
    }

    typedef te-metric-type {
      type union {
        type enumeration {
          enum IGP {
            description
             "set the LSP metric to track the underlying
              IGP metric";
          }
        }
        type uint32;
      }
      description
       "union type for setting the LSP TE metric to a
        static value, or to track the IGP metric";
    }
```

```
     typedef cspf-tie-breaking {
       type enumeration {
         enum RANDOM {
           description
            "CSPF calculation selects a random path among
             multiple equal-cost paths to the destination";
         }
         enum LEAST_FILL {
           description
            "CSPF calculation selects the path with greatest
             available bandwidth";
         }
         enum MOST_FILL {
           description "CSPF calculation selects the path with the least
             available bandwidth";
         }
       }
       default RANDOM;
       description
        "type to indicate the CSPF selection policy when
         multiple equal cost paths are available";
     }


     // grouping statements

     grouping te-tunnel-reoptimize_config {
       description "Definition for reoptimize timer configuration";
       leaf te-lsp-reoptimize-timer {
         type uint16;
         units seconds;
         description
          "frequency of reoptimization of
           a traffic engineered LSP";
       }
     }

     grouping path-placement-constraints {
       description
         "Top level grouping for path placement constraints";

         container admin-groups {
           description
             "Include/Exclude constraints for link affinities";
           uses te-lsp-exclude-admin-group_config;
           uses te-lsp-include-any-admin-group_config;
           uses te-lsp-include-all-admin-group_config;
         }
```

```
    }

    grouping te-tunnel-bandwidth_config {
      description "Bandwidth configuration for TE LSPs";
      choice bandwidth {
        default explicit;
        description
         "select how bandwidth for the LSP will be
          specified and managed";
        case explicit {
          leaf set-bandwidth {
            type uint32;
            description
             "set bandwidth explicitly, e.g., using
              offline calculation";
          }
        }
        case auto {
          uses te-lsp-auto-bandwidth_config;
        }
      }
    }

    grouping te-lsp-auto-bandwidth_config {
      description "Configuration parameters related to autobandwidth";
      container auto-bandwidth {
        description
         "configure auto-bandwidth operation in
          which devices automatically adjust bandwidth to meet
          requirements";

        leaf enabled {
          type boolean;
          default false;
          description
           "enables mpls auto-bandwidth on the
            lsp";
        }

        leaf min-bw {
          type uint32;
          description
           "set the minimum bandwidth in Mbps for an
            auto-bandwidth LSP";
        }

        leaf max-bw {
          type uint32;
```

```
          description
           "set the maximum bandwidth in Mbps for an
            auto-bandwidth LSP";
        }

        leaf adjust-interval {
          type uint32;
          description
           "time in seconds between adjustments to
            LSP bandwidth";
        }

        leaf adjust-threshold {
          type oc-types:percentage;
          description
           "percentage difference between the LSP's
            specified bandwidth and its current bandwidth
            allocation -- if the difference is greater than the
            specified percentage, auto-bandwidth adjustment is
            triggered";
        }

        container overflow {
          description
           "configuration of MPLS overflow bandwidth
            adjustement for the LSP";
          uses te-lsp-overflow_config;
        }

        container underflow {
          description
           "configuration of MPLS underflow bandwidth
            adjustement for the LSP";
          uses te-lsp-underflow_config;
        }
      }
    }

    grouping  te-lsp-overflow_config {
      description
       "configuration for mpls lsp bandwidth
        overflow adjustment";

      leaf enabled {
        type boolean;
        default false;
        description
         "enables mpls lsp bandwidth overflow
```

```
            adjustment on the lsp";
        }

        leaf overflow-threshold {
          type oc-types:percentage;
          description
           "bandwidth percentage change to trigger
            an overflow event";

        }

        leaf trigger-event-count {
          type uint16;
          description
           "number of consecutive overflow sample
            events needed to trigger an overflow adjustment";
        }
      }

      grouping te-lsp-underflow_config {
        description
          "configuration for mpls lsp bandwidth
          underflow adjustment";

        leaf enabled {
          type boolean;
          default false;
          description
           "enables bandwidth underflow
            adjustment on the lsp";
        }

        leaf underflow-threshold {
          type oc-types:percentage;
          description
           "bandwidth percentage change to trigger
            and underflow event";
        }

        leaf trigger-event-count {
          type uint16;
          description
           "number of consecutive underflow sample
            events needed to trigger an underflow adjustment";
        }
      }

      grouping te-tunnel-metric_config {
```

```
      description "Configuration parameters related to LSP metric";
      leaf metric {
        type te-metric-type;
        description "LSP metric, either explicit or IGP";
      }
    }

    grouping te-lsp-exclude-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        to exclude in path calculation";
      list exclude-groups {

        key exclude-admin-group-name;

        description
          "list of admin-groups to exclude in path calculation";

        leaf exclude-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined admin
            group";
        }
      }
    }

    grouping te-lsp-include-all-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        which all must be included in the path calculation";
      list include-all-groups {

        key all-admin-group-name;
        description
          "list of admin-groups of which all must be included";

        leaf all-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined
            admin group";
```

```
        }
      }
    }

    grouping te-lsp-include-any-admin-group_config {
      description
       "Configuration parameters related to admin-groups
        of which one must be included in the path calculation";
      list include-any-groups {

        key any-admin-group-name;
        description
          "list of admin-groups of which one must be included";

        leaf any-admin-group-name {
          type leafref {
            path "/mpls/te-global-attributes/mpls-admin-groups/" +
              "admin-group/admin-group-name";
          }
          description
            "name of the admin group -- references a defined
            admin group";
        }
      }
    }

    grouping te-tunnel-protection_config {
      description
       "Configuration parameters related to LSP
        protection";
      leaf protection-style-requested {
        type identityref {
          base mplst:protection-type;
        }
        default mplst:unprotected;
        description
          "style of mpls frr protection desired: can be
          link, link-node or unprotected.";
      }
    }

    grouping te-lsp-comp-explicit {
      description
       "definitions for LSPs in which hops are explicitly
        specified";

      container explicit-path {
        description "LSP with explicit path specification";
```

```
        leaf path-name {
          type leafref {
            path "/mpls/lsps/constrained-path/"
              + "explicit-path/config/named-explicit-paths/name";
            require-instance true;
          }
          description "reference to a defined path";
        }
      }
    }

    grouping te-lsp-comp-queried {
      description "definitons for LSPs computed by querying a remote
        service, e.g., PCE server";

      container queried-path {
        description "LSP with path queried from an external server";

        leaf path-computation-server {
          type inet:ip-address;
          description
           "Address of the external path computation
            server";
        }
      }
    }

    grouping te-lsp-comp-local {
      description "definitons for locally-computed LSPs";

      container locally-computed {
        description "LSP with path computed by local ingress LSR";

        leaf use-cspf {
          type boolean;
          description "Flag to enable CSPF for locally computed LSPs";
        }
        leaf cspf-tiebreaker {
          type cspf-tie-breaking;
          description
            "Determine the tie-breaking method to choose between
            equally desirable paths during CSFP computation";
        }
      }
    }


    grouping explicit-route-subobject {
```

```
      description
        "The explicit route subobject grouping";
      choice type {
        description
          "The explicit route subobject type";
        case ipv4-address {
          description
            "IPv4 address explicit route subobject";
          leaf address {
            type inet:ip-address;
            description "router hop for the LSP path";
          }

          leaf hop-type {
            type mpls-hop-type;
            description "strict or loose hop";
          }
        }

        case label {
          leaf value {
            type uint32;
            description "the label value";
          }
          description
            "The Label ERO subobject";
        }
      }
    }


    // Explicit paths config somewhat following the IETF model
    grouping named-explicit-path_config {
      description
        "Global explicit path configuration
        grouping";
      list named-explicit-paths {
        key "name";
        description
          "A list of explicit paths";
        leaf name {
          type string;
          description
            "A string name that uniquely identifies
            an explicit path";
        }
        list explicit-route-objects {
          key "index";
```

```
          description
            "List of explicit route objects";
          leaf index {
            type uint8 {
              range "0..255";
            }
            description
              "Index of this explicit route object,
               to express the order of hops in path";
          }
          uses explicit-route-subobject;
        }
      }
    }

    grouping explicit-paths-top {
      description
        "common information for MPLS explicit path definition";
        list explicit-path {
          key name;
          description "Explicit path definition";

          leaf name {
            type leafref {
              path "/mpls/lsps/constrained-path/"
                + "explicit-path/config/named-explicit-paths/name";
              require-instance true;
            }
            description "definition for naming an explicit  path";

          }
          container config {
            description "configuration for an explicit path";
            uses named-explicit-path_config;
          }
          container state {
            config false;
            description "operational state for LSP path name";
            uses named-explicit-path_config;
          }
        }
    }

    grouping mpls-te-srlg_config {
      description
        "Configuration of various attributes associated
        with the SRLG";
```

```
        leaf srlg-name {
          type string;
          description "SRLG group identifier";
        }

        leaf srlg-value {
          type uint32;
          description "group ID for the SRLG";
        }

        leaf srlg-cost {
          type uint32;
          description
            "The cost of the SRLG to the computation
            algorithm";
        }
      }

      grouping mpls-te-srlg-members_config {
        description "Configuration of the membership of the SRLG";

        leaf from-address {
          type inet:ip-address;
          description "IP address of the a-side of the SRLG link";
        }

        leaf to-address {
          type inet:ip-address;
          description "IP address of the z-side of the SRLG link";
        }
      }

      grouping mpls-te-srlg-top {
        description
          "Top level grouping for MPLS shared
          risk link groups.";
        container mpls-te-srlg {
          description
            "Shared risk link groups attributes";
          list srlg {
            key srlg-name;
            description "List of shared risk link groups";

            leaf srlg-name {
              type leafref {
                path "../config/srlg-name";
                require-instance true;
              }
```

```
            description "The SRLG group identifier";
          }

          container config {
            description "Configuration parameters related to the SRLG";
            uses mpls-te-srlg_config;
          }

          container state {
            config false;
            description "State parameters related to the SRLG";
            uses mpls-te-srlg_config;
          }

          list members-list {
            key from-address;
            description
              "List of SRLG members, which are expressed
              as IP address endpoints of links contained in the SRLG";

            leaf from-address {
              type leafref {
                path "../config/from-address";
                require-instance true;
              }
              description "The from address of the link in the SRLG";
            }

            container config {
              description
                "Configuration parameters relating to the
                SRLG members";
              uses mpls-te-srlg-members_config;
            }

            container state {
              config false;
              description
                "State parameters relating to the SRLG
                members";
              uses mpls-te-srlg-members_config;
            }
          }
        }
      }
    }

    grouping tunnel-path_config {
```

```
      description
        "Tunnel path properties grouping";
      container path-computation-method {
        description
          "select and configure the way the LSP path is
          computed";

        leaf path-computation {
          type identityref {
            base path-computation-method;
          }
          description "path computation method to use with the LSP";
        }

        uses te-lsp-comp-explicit;
        uses te-lsp-comp-queried;
        uses te-lsp-comp-local;
      }

      uses path-placement-constraints;

      leaf no-cspf {
        type empty;
        description
          "Indicates no CSPF is to be attempted on this
          path.";
      }

      choice sigaling-specific-path-attributes {
        description "Signaling-protocol specific path attributes.";
        case RSVP {
          uses rsvp:rsvp-p2p-path-attributes_config;
        }
        case SR {
          uses sr:sr-path-attributes_config;
        }
      }
    }

    grouping te-tunnel_config {
      description
        "Configuration parameters relevant to a single
         traffic engineered tunnel.";

      leaf name {
        type string;
        description "The tunnel name";
      }
```

```
        leaf type {
          type identityref {
            base mplst:tunnel-type;
          }
          description "Tunnel type, p2p or p2mp";
        }

        leaf local-id {
          type union {
            type uint32;
            type string;
          }
          description
            "locally signficant optional identifier for the
             tunnel; may be a numerical or string value";
        }

        leaf description {
          type string;
          description "optional text description for the tunnel";
        }

        leaf admin-status {
          type identityref {
            base mplst:tunnel-admin-status;
          }
          default mplst:ADMIN_UP;
          description "TE tunnel administrative state.";
        }

        leaf preference {
          type uint8 {
            range "1..255";
          }
          description "Specifies a preference for this tunnel.
            A lower number signifies a better preference";
        }

        uses te-tunnel-metric_config;
        uses te-tunnel-bandwidth_config;
        uses te-tunnel-protection_config;
        uses te-tunnel-reoptimize_config;

        choice signaling-specific-tunnel-attributes {
          description "Signaling-protocol specific path attributes.";
          case RSVP {
            uses rsvp:rsvp-p2p-tunnel-attributes_config;
          }
```

```
      }

      choice tunnel-type {
        description
          "Describes tunnel by type type";
        case p2p {
          leaf destination {
            type inet:ip-address;
            description
              "P2P tunnel destination address";
          }
          /* P2P list of path(s) */
          list primary-paths {
            key "name";
            leaf name {
              type string;
              description "Path name";
            }
            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_config;
          }

          list secondary-paths {
            key "name";
            description
              "List of secondary paths for this
                tunnel.";
            leaf name {
              type string;
              description "Path name";
            }
            leaf preference {
              type uint8 {
                range "1..255";
              }
              description
```

```
                "Specifies a preference for
                  this path. The lower the
                  number higher the
                  preference";
              }
              uses tunnel-path_config;
            }
          }
          case p2mp {
            // TODO - complete
          }
        }
      }

      grouping te-tunnel_state {
        description
          "Counters and statistical data relevent to a single
           tunnel.";


        leaf oper-status {
          type identityref {
            base mplst:lsp-oper-status;
          }
          description
            "The operational status of the TE tunnel";
        }

        leaf role {
          type identityref {
            base mplst:lsp-role;
          }
          description
            "The lsp role at the current node, whether it is headend,
            transit or tailend.";
        }

        container counters {
          description
            "State data for MPLS label switched paths. This state
            data is specific to a single label switched path.";

          leaf bytes {
            type yang:counter64;
            description
              "Number of bytes that have been forwarded over the
               label switched path.";
          }
```

```
          leaf packets {
            type yang:counter64;
            description
              "Number of pacets that have been forwarded over the
               label switched path.";
          }

          leaf path-changes {
            type yang:counter64;
            description
              "Number of path changes for the label switched path";
          }

          leaf state-changes {
            type yang:counter64;
            description
              "Number of state changes for the label switched path";
          }

          leaf online-time {
            type yang:date-and-time;
            description
              "Indication of the time the label switched path
               transitioned to an Oper Up or in-service state";
          }

          leaf current-path-time {
            type yang:date-and-time;
            description
              "Indicates the time the LSP switched onto its
               current path. This is reset upon a LSP path
               change.";
          }

          leaf next-reoptimization-time {
            type yang:date-and-time;
            description
              "Indicates the next scheduled time the LSP
               will be reoptimized.";
          }
        }
      }

      grouping te-tunnels-top {
        description
          "Top level grouping for TE tunnels";

        list tunnel {
```

```
        key "name type";
        description "List of TE tunnels";

        leaf name {
          type leafref {
            path "../config/name";
            require-instance true;
          }
          description "The tunnel name";
        }

        leaf type {
          type leafref {
            path "../config/type";
            require-instance true;
          }
          description "The tunnel type, p2p or p2mp.";
        }

        container config {
          description
            "Configuration parameters related to TE tunnels:";
          uses te-tunnel_config;
        }

        container state {
          config false;
          description "State parameters related to TE interfaces";
          uses te-tunnel_config;
          uses te-tunnel_state;
        }
      }
    }

  // data definition statements

  // augment statements

  // rpc statements

  // notification statements
  }
```

<CODE ENDS>

6.3. MPLS signaling protocol modules

```
                  <CODE BEGINS> file openconfig-mpls-rsvp.yang
   module openconfig-mpls-rsvp {

     yang-version "1";

     // namespace
     namespace "http://openconfig.net/yang/rsvp";

     prefix "rsvp";

     // import some basic types
     import ietf-inet-types { prefix inet; }
     import openconfig-mpls-types { prefix mplst; }
     import ietf-yang-types { prefix yang; }
     import openconfig-types { prefix oc-types; }


     // meta
     organization "OpenConfig working group";

     contact
       "OpenConfig working group
        netopenconfig@googlegroups.com";

     description
       "Configuration for RSVP-TE signaling, including global protocol
        parameters and LSP-specific configuration for constrained-path
        LSPs";

     revision "2015-09-18" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements
```

```
    grouping mpls-rsvp-soft-preemption_config {
      description "Configuration for MPLS soft preemption";
      leaf enable {
        type boolean;
        default false;
        description "Enables soft preemption on a node.";
      }

      leaf soft-preemption-timeout {
        type uint16 {
          range 0..max;
        }
        // The RFC actually recommends 30 seconds as default.
        default 0;
        description
          "Timeout value for soft preemption to revert
           to hard preemption";
        reference "RFC5712 MPLS-TE soft preemption";
      }
    }

    grouping mpls-rsvp-soft-preemption {
      description "Top level group for MPLS soft preemption";
      container soft-preemption {
        description
          "Protocol options relating to RSVP
           soft preemption";
        container config {
          description
            "Configuration parameters relating to RSVP
             soft preemption support";
          uses mpls-rsvp-soft-preemption_config;
        }
        container state {
          config false;
          description
            "State parameters relating to RSVP
             soft preemption support";
          uses mpls-rsvp-soft-preemption_config;
        }
      }
    }

    grouping mpls-rsvp-hellos_config {
      description "RSVP protocol options configuration.";

      leaf hello-interval {
        type uint16 {
```

```
          range 1000..60000;
        }
        units milliseconds;
        default 9000;
        description
          "set the interval in ms between RSVP hello
           messages";
        reference
          "RFC 3209: RSVP-TE: Extensions to RSVP for
           LSP Tunnels.
           RFC 5495: Description of the Resource
           Reservation Protocol - Traffic-Engineered
           (RSVP-TE) Graceful Restart Procedures";
      }

      leaf refresh-reduction {
        type boolean;
        default true;
        description
          "enables all RSVP refresh reduction message
           bundling, RSVP message ID, reliable message delivery
           and summary refresh";
        reference
          "RFC 2961 RSVP Refresh Overhead Reduction
           Extensions";
      }
    }

    grouping mpls-rsvp-hellos {
      description "Top level grouping for RSVP hellos parameters";
      // TODO: confirm that the described semantics are supported
      // on various implementations. Finer grain configuration
      // will be vendor-specific

      container rsvp-hellos {
        description "Top level container for RSVP hello parameters";
        container config {
          description
            "Configuration parameters relating to RSVP
             hellos";
          uses mpls-rsvp-hellos_config;
        }
        container state {
          config false;
          description "State information associated with RSVP hellos";
          uses mpls-rsvp-hellos_config;
        }
      }
```

```
    }

    grouping mpls-rsvp-subscription_config {
      description "RSVP subscription configuration";
      leaf subscription {
        type oc-types:percentage;
        description
          "percentage of the interface bandwidth that
           RSVP can reserve";
      }
    }
    grouping mpls-rsvp-subscription {
      description "Top level group for RSVP subscription options";
      container subscription {
        description
          "Bandwidth percentage reservable by RSVP
           on an interface";
        container config {
          description
            "Configuration parameters relating to RSVP
             subscription options";
          uses mpls-rsvp-subscription_config;
        }
        container state {
          config false;
          description
          "State parameters relating to RSVP
             subscription options";
          uses mpls-rsvp-subscription_config;
        }
      }
    }

    grouping mpls-rsvp-graceful-restart_config {
      description
        "Configuration parameters relating to RSVP Graceful-Restart";

      leaf enable {
        type boolean;
        default false;
        description "Enables graceful restart on the node.";
      }

      leaf restart-time {
        type uint32;
        description
          "Graceful restart time (seconds).";
        reference
```

```
          "RFC 5495: Description of the Resource
           Reservation Protocol - Traffic-Engineered
           (RSVP-TE) Graceful Restart Procedures";
      }
      leaf recovery-time {
        type uint32;
        description
          "RSVP state recovery time";
      }
    }

    grouping mpls-rsvp-graceful-restart {
      description
        "Top level group for RSVP graceful-restart
         parameters";
      container graceful-restart {
        description "TODO";
        container config {
          description
            "Configuration parameters relating to
             graceful-restart";
          uses mpls-rsvp-graceful-restart_config;
        }
        container state {
          config false;
          description
            "State information associated with
             RSVP graceful-restart";
          uses mpls-rsvp-graceful-restart_config;
        }
      }
    }

    grouping mpls-rsvp-authentication_config {
      description "RSVP authentication parameters container.";
      leaf enable {
        type boolean;
        default false;
        description "Enables RSVP authentication on the node.";
      }
      leaf authentication-key {
        type string {
          // Juniper supports 1..16 while
          // Cisco has a much bigger range, up to 60.
          length "1..32";
        }
        description
          "authenticate RSVP signaling
```

```
           messages";
        reference
          "RFC 2747: RSVP Cryptographic Authentication";
      }
    }

    grouping mpls-rsvp-authentication {
      description
        "Top level group for RSVP authentication,
         as per RFC2747";
      container authentication {
        description "TODO";
        container config {
          description
            "Configuration parameters relating
             to authentication";
          uses mpls-rsvp-authentication_config;
        }
        container state {
          config false;
          description
            "State information associated
             with authentication";
          uses mpls-rsvp-authentication_config;
        }
      }
    }

    grouping mpls-rsvp-protection_config {
      description "RSVP facility (link/node) protection configuration";

      leaf link-protection-style-requested {
        type identityref {
          base mplst:protection-type;
        }
        default mplst:link-node-protection-requested;
        description
          "style of mpls frr protection desired:
          link, link-node, or unprotected";
      }

      leaf bypass-optimize-interval {
        type uint16;
        units seconds;
        description
          "interval between periodic optimization
          of the bypass LSPs";
        // note: this is interface specific on juniper
```

```
        // on iox, this is global. need to resolve.
      }
      // to be completed, things like enabling link protection,
      // optimization times, etc.
    }

    grouping mpls-rsvp-link-protection {
      description "Top level group for RSVP protection";
      container protection {
        description "link-protection (NHOP) related configuration";
        container config {
          description "Configuration for link-protection";
          uses mpls-rsvp-protection_config;
        }
        container state {
          config false;
          description "State for link-protection";
          uses mpls-rsvp-protection_config;
        }
      }
    }

    grouping mpls-rsvp-error-statistics {
      description "RSVP-TE packet statistics";
      container error {
        description "RSVP-TE error statistics";
        leaf authentication-failure {
          type yang:counter32;
          description
            "Authentication failure count";
        }

        leaf path-error {
          type yang:counter32;
          description
            "Path error to client count";
        }

        leaf resv-error {
          type yang:counter32;
          description
            "Resv error to client count";
        }

        leaf path-timeout {
          type yang:counter32;
          description
            "Path timeout count";
```

```
        }

        leaf resv-timeout {
          type yang:counter32;
          description
            "Resv timeout count";
        }

        leaf rate-limit {
          type yang:counter32;
          description
            "Count of packets that were rate limited";
        }

        // TODO - complete the other error statistics
      }
    }

    grouping mpls-rsvp-protocol-statistics {
      description "RSVP protocol statistics";
      container protocol {
        description "RSVP-TE protocol statistics";
        leaf hello-sent {
          type yang:counter32;
          description
            "Hello sent count";
        }

        leaf hello-rcvd {
          type yang:counter32;
          description
            "Hello received count";
        }

        leaf path-sent {
          type yang:counter32;
          description
            "Path sent count";
        }

        leaf path-rcvd {
          type yang:counter32;
          description
            "Path received count";
        }

        // TODO - To be completed the other packet statistics
      }
```

```
    }

    grouping mpls-rsvp-statistics {
      description "Top level grouping for RSVP protocol state";
      uses mpls-rsvp-protocol-state;
    }

    grouping rsvp-global {
      description "Global RSVP protocol configuration";
      container rsvp-te {
        description "RSVP-TE global signaling protocol configuration";

        container rsvp-sessions {
          description "Configuration and state of RSVP sessions";

          container config {
            description
              "Configuration of RSVP sessions on the device";
          }

          container state {
            config false;
            description
              "State information relating to RSVP sessions
              on the device";
            uses mpls-rsvp-session-state;
          }
        }

        container rsvp-neighbors {
          description
            "Configuration and state for RSVP neighbors connecting
            to the device";

          container config {
            description "Configuration of RSVP neighbor information";
          }

          container state {
            config false;
            description
              "State information relating to RSVP neighbors";
            uses mpls-rsvp-neighbor-state;
          }
        }

        container global {
          description "Platform wide RSVP configuration and state";
```

```
            uses mpls-rsvp-graceful-restart;
            uses mpls-rsvp-soft-preemption;

            container statistics {
              config false;
              description "Platform wide RSVP state, including counters";
              // TODO - reconcile global and per-interface
              // protocol-related statistics

              container counters {
                config false;
                description
                  "Platform wide RSVP statistics and counters";
                uses mpls-rsvp-global-protocol-state;
                uses mpls-rsvp-statistics;
              }
            }
          }

          container interface-attributes {
            // interfaces, bw percentages, hello timers, etc goes here";

            list interface {
              key interface-name;
              description "list of per-interface RSVP configurations";

              // TODO: update to interface ref -- move to separate
              // augmentation.
              leaf interface-name {
                type leafref {
                  path "../config/interface-name";
                  require-instance true;
                }
                description "references a configured IP interface";
              }


              container config {
                description
                  "Configuration of per-interface RSVP parameters";

                leaf interface-name {
                  type string;
                  description "Name of configured IP interface";
                }
              }

              container state {
```

```
              config false;
              description
                "Per-interface RSVP protocol and state information";
              uses mpls-rsvp-interfaces-state;

              container counters {
                config false;
                description
                  "Interface specific RSVP statistics and counters";
                uses mpls-rsvp-protocol-state;
              }
            }

            uses mpls-rsvp-hellos;
            uses mpls-rsvp-authentication;
            uses mpls-rsvp-subscription;
            uses mpls-rsvp-link-protection;
          }
        }
      }
    }

    grouping rsvp-p2p-tunnel-attributes_config {
      description "properties of RSPP point-to-point paths";

     leaf source {
       type inet:ip-address;
       description
         "tunnel source address";
     }

     leaf soft-preemption {
        type boolean;
        default false;
        description "enables RSVP soft-preemption on this LSP";
      }
    }


    grouping rsvp-p2p-path-attributes_config {
      description "properties of RSPP point-to-point paths";
      leaf setup-priority {
        type uint8 {
          range 0..7;
        }
        default 7;
        description
          "preemption priority during LSP setup, lower is
```

```
            higher priority; default 7 indicates that LSP will not
            preempt established LSPs during setup";
          reference "RFC 3209 - RSVP-TE: Extensions to RSVP for
            LSP Tunnels";
        }

        leaf hold-priority {
          type uint8 {
            range 0..7;
          }
          default 0;
          description
            "preemption priority once the LSP is established,
             lower is higher priority; default 0 indicates other LSPs
             will not preempt the LSPs once established";
          reference "RFC 3209 - RSVP-TE: Extensions to RSVP for
            LSP Tunnels";
        }

        leaf retry-timer {
          type uint16 {
            range 1..600;
          }
          units seconds;
          description
            "sets the time between attempts to establish the
             LSP";
        }
      }

      grouping mpls-rsvp-neighbor-state {
        description "State information for RSVP neighbors";

        list rsvp-neighbor {
          key "neighbor-address";
          description
            "List of RSVP neighbors connecting to the device,
            keyed by neighbor address";

          leaf neighbor-address {
            type inet:ip-address;
            description "Address of RSVP neighbor";
          }

          leaf detected-interface {
            type string;
            description "Interface where RSVP neighbor was detected";
          }
```

```
        leaf neighbor-status {
          type enumeration {
            enum UP {
              description
                "RSVP hello messages are detected from the neighbor";
            }
            enum DOWN {
              description
                "RSVP neighbor not detected as up, due to a
                 communication failure or IGP notification
                 the neighbor is unavailable";
            }
          }
          description "Enumuration of possible RSVP neighbor states";
        }

        leaf neighbor-refresh-reduction {
          type boolean;
          description
            "Suppport of neighbor for RSVP refresh reduction";
          reference
            "RFC 2961 RSVP Refresh Overhead Reduction
             Extensions";
        }
      }
    }

    grouping mpls-rsvp-session-state {
      description "State information for RSVP TE sessions";
      list rsvp-session {
        key "source-port destination-port
         source-address destination-address";
        description "List of RSVP sessions";

        leaf source-address {
          type inet:ip-address;
          description "Origin address of RSVP session";
        }

        leaf destination-address {
          type inet:ip-address;
          description "Destination address of RSVP session";
        }

        leaf source-port {
          type uint16;
          description "RSVP source port";
          reference "RFC 2205";
```

```
        }

        leaf destination-port {
          type uint16;
          description "RSVP source port";
          reference "RFC 2205";

        }

        leaf session-state {
          type enumeration {
            enum UP {
              description "RSVP session is up";
            }
            enum DOWN {
              description "RSVP session is down";
            }
          }
          description "Enumeration of RSVP session states";
        }

        leaf session-type {
          type enumeration {
            enum SOURCE {
              description "RSVP session originates on this device";
            }
            enum TRANSIT {
              description "RSVP session transits this device only";
            }
            enum DESTINATION {
              description "RSVP session terminates on this device";
            }
          }
          description "Enumeration of possible RSVP session types";
        }

        leaf tunnel-id {
          type uint16;
          description "Unique identifier of RSVP session";
        }

        leaf label-in {
          type mplst:mpls-label;
          description
            "Incoming MPLS label associated with this RSVP session";
        }

        leaf label-out {
```

```
          type mplst:mpls-label;
          description
            "Outgoing MPLS label associated with this RSVP session";
        }

        leaf-list associated-lsps {
          type leafref {
            path "/mpls/lsps/constrained-path/tunnel/" +
            "config/name";
          }
          description
            "List of label switched paths associated with this RSVP
             session";
        }
      }
    } //rsvp-session-state

    grouping mpls-rsvp-interfaces-state {
      description "RSVP state information relevant to an interface";

      list bandwidth {
        key priority;
        description
          "Available and reserved bandwidth by priority on
           the interface.";

        leaf priority {
          type uint8 {
            range 0..7;
          }
          description
            "RSVP priority level for LSPs traversing the interface";
        }

        leaf available-bandwidth {
          type mplst:bandwidth-mbps;
          description "Bandwidth currently available";
        }

        leaf reserved-bandwidth {
          type mplst:bandwidth-mbps;
          description "Bandwidth currently reserved";
        }
      }

      leaf highwater-mark {
        type mplst:bandwidth-mbps;
        description "Maximum bandwidth ever reserved";
```

```
      }

      leaf active-reservation-count {
        type yang:gauge64;
        description "Number of active RSVP reservations";
      }
    }

    grouping mpls-rsvp-global-protocol-state {
      description "RSVP protocol statistics which may not apply
        on an interface, but are significant globally.";

      leaf path-timeouts {
        type yang:counter64;
        description "TODO";
      }

      leaf reservation-timeouts {
        type yang:counter64;
        description "TODO";
      }

      leaf rate-limited-messages {
        type yang:counter64;
        description "RSVP messages dropped due to rate limiting";
      }
    }

    grouping mpls-rsvp-protocol-state {
      description "RSVP protocol statistics and message counters";
      leaf in-path-messages {
        type yang:counter64;
        description "Number of received RSVP Path messages";
      }

      leaf in-path-error-messages {
        type yang:counter64;
        description "Number of received RSVP Path Error messages";
      }

      leaf in-path-tear-messages {
        type yang:counter64;
        description "Number of received RSVP Path Tear messages";
      }

      leaf in-reservation-messages {
        type yang:counter64;
        description "Number of received RSVP Resv messages";
```

```
      }

      leaf in-reservation-error-messages {
        type yang:counter64;
        description "Number of received RSVP Resv Error messages";
      }

      leaf in-reservation-tear-messages {
        type yang:counter64;
        description "Number of received RSVP Resv Tear messages";
      }

      leaf in-rsvp-hello-messages {
        type yang:counter64;
        description "Number of received RSVP hello messages";
      }

      leaf in-rsvp-srefresh-messages {
        type yang:counter64;
        description "Number of received RSVP summary refresh messages";
      }

      leaf in-rsvp-ack-messages {
        type yang:counter64;
        description
          "Number of received RSVP refresh reduction ack
           messages";
      }

      leaf out-path-messages {
        type yang:counter64;
        description "Number of sent RSVP PATH messages";
      }

      leaf out-path-error-messages {
        type yang:counter64;
        description "Number of sent RSVP Path Error messages";
      }

      leaf out-path-tear-messages {
        type yang:counter64;
        description "Number of sent RSVP Path Tear messages";
      }

      leaf out-reservation-messages {
        type yang:counter64;
        description "Number of sent RSVP Resv messages";
      }
```

```
      leaf out-reservation-error-messages {
        type yang:counter64;
        description "Number of sent RSVP Resv Error messages";
      }

      leaf out-reservation-tear-messages {
        type yang:counter64;
        description "Number of sent RSVP Resv Tear messages";
      }

      leaf out-rsvp-hello-messages {
        type yang:counter64;
        description "Number of sent RSVP hello messages";
      }

      leaf out-rsvp-srefresh-messages {
        type yang:counter64;
        description "Number of sent RSVP summary refresh messages";
      }

      leaf out-rsvp-ack-messages {
        type yang:counter64;
        description
          "Number of sent RSVP refresh reduction ack messages";
      }
    }



    // data definition statements

    // augment statements

    // rpc statements

    // notification statements

  }
                  <CODE ENDS>



                  <CODE BEGINS> file openconfig-mpls-sr.yang
  module openconfig-mpls-sr {

    yang-version "1";

    // namespace
```

```
     namespace "http://openconfig.net/yang/sr";

     prefix "sr";


     // import some basic types
     import ietf-inet-types { prefix inet; }
     import openconfig-mpls-types { prefix mplst; }

     // meta
     organization "OpenConfig working group";

     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Configuration for MPLS with segment routing-based LSPs,
       including global parameters, and LSP-specific configuration for
       both constrained-path and IGP-congruent LSPs";

     revision "2015-10-14" {
       description
         "Work in progress";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     grouping srgb_config {

       // Matches the "global" configuration options in
       // draft-litkowski-spring-yang...
       // TODO: request to Stephane for this to be a separate
       // grouping such that it can be included.

       leaf lower-bound {
         type uint32;
         description
           "Lower value in the block.";
       }
       leaf upper-bound {
```

```
          type uint32;
          description
            "Upper value in the block.";
        }
        description
          "List of global blocks to be advertised.";
      }

      grouping srgb_state {
        description
          "State parameters relating to the SRGB";

        leaf size {
          type uint32;
          description
            "Number of indexes in the SRGB block";
        }
        leaf free {
          type uint32;
          description
            "Number of SRGB indexes that have not yet been allocated";
        }
        leaf used {
          type uint32;
          description
            "Number of SRGB indexes that are currently allocated";
        }

        // TODO: where do we put LFIB entries?

      }

      grouping adjacency-sid_config {
        description
          "Configuration related to an Adjacency Segment Identifier
          (SID)";

        // tuned from draft-litkowski-spring-yang
        // TODO: need to send a patch to Stephane

        leaf-list advertise {
          type enumeration {
            enum "PROTECTED" {
              description
                "Advertise an Adjacency-SID for this interface, which is
                eligible to be protected using a local protection
                mechanism on the local LSR. The local protection
                mechanism selected is dependent upon the configuration
```

```
              of RSVP-TE FRR or LFA elsewhere on the system";
          }
          enum UNPROTECTED {
            description
              "Advertise an Adajcency-SID for this interface, which is
              explicitly excluded from being protected by any local
              protection mechanism";
          }
        }
        description
          "Specifies the type of adjacency SID which should be
          advertised for the specified entity.";
      }

      leaf-list groups {
        type uint32;
        description
          "Specifies the groups to which this interface belongs.
          Setting a value in this list results in an additional AdjSID
          being advertised, with the S-bit set to 1. The AdjSID is
          assumed to be protected";
      }
    }

    grouping interface_config {
      description
        "Configuration parameters relating to a Segment Routing
        enabled interface";

      leaf interface {
        type string;
        // TODO: this should be changed to a leafref.
        description
          "Reference to the interface for which segment routing
          configuration is to be applied.";
      }
    }

    // grouping statements

    grouping sr-global {
      description "global segment routing signaling configuration";

      container segment-routing {
        description "SR global signaling config";

        list srgb {
          key "lower-bound upper-bound";
```

```
            uses srgb_config;
            container config {
              description
                "Configuration parameters relating to the Segment Routing
                Global Block (SRGB)";
              uses srgb_config;
            }
            container state {
              config false;
              description
                "State parameters relating to the Segment Routing Global
                Block (SRGB)";
              uses srgb_config;
              uses srgb_state;
            }
            description
              "List of Segment Routing Global Block (SRGB) entries. These
              label blocks are reserved to be allocated as domain-wide
              entries.";
          }

          list interfaces {
            key "interface";
            uses interface_config;
            container config {
              description
                "Interface configuration parameters for Segment Routing
                relating to the specified interface";
              uses interface_config;
            }
            container state {
              config false;
              description
                "State parameters for Segment Routing features relating
                to the specified interface";
              uses interface_config;
            }
            container adjacency-sid {
              description
                "Configuration for Adjacency SIDs that are related to
                the specified interface";
              container config {
                description
                  "Configuration parameters for the Adjacency-SIDs
                  that are related to this interface";
                uses adjacency-sid_config;
              }
              container state {
```

```
              config false;
              description
                "State parameters for the Adjacency-SIDs that are
                related to this interface";
              uses adjacency-sid_config;
            }
          }
          description
            "List of interfaces with associated segment routing
            configuration";
        }
      }
    }

    grouping sr-path-attributes_config {
      description
        "Configuration parameters relating to SR-TE LSPs";

      leaf sid-selection-mode {
        type enumeration {
          enum "ADJ-SID-ONLY" {
            description
              "The SR-TE tunnel should only use adjacency SIDs
              to build the SID stack to be pushed for the LSP";
          }
          enum "MIXED-MODE" {
            description
              "The SR-TE tunnel can use a mix of adjacency
              and prefix SIDs to build the SID stack to be pushed
              to the LSP";
          }
        }
        default "MIXED-MODE";
        description
          "The restrictions placed on the SIDs to be selected by the
          calculation method for the SR-TE LSP";
      }

      leaf sid-protection-required {
        type boolean;
        default "false";
        description
          "When this value is set to true, only SIDs that are
          protected are to be selected by the calculating method
          for the SR-TE LSP.";
      }
    }
```

```
    grouping sr_fec-address_config {
      description
        "Configuration parameters relating to a FEC that is to be
        advertised by Segment Routing";

      leaf fec-address {
        type inet:ip-prefix;
        description
          "FEC that is to be advertised as part of the Prefix-SID";
      }
    }

    grouping sr_fec-prefix-sid_config {
      description
        "Configuration parameters relating to the nature of the
        Prefix-SID that is to be advertised for a particular FEC";

      leaf type {
        type enumeration {
          enum "INDEX" {
            description
              "Set when the value of the prefix SID should be specified
              as an off-set from the SRGB's zero-value. When multiple
              SRGBs are specified, the zero-value is the minimum
              of their lower bounds";
          }
          enum "ABSOLUTE" {
            description
              "Set when the value of a prefix SID is specified as the
              absolute value within an SRGB. It is an error to specify
              an absolute value outside of a specified SRGB";
          }
        }
        default "INDEX";
        description
          "Specifies how the value of the Prefix-SID should be
          interpreted - whether as an offset to the SRGB, or as an
          absolute value";
      }

      leaf node-flag {
        type boolean;
        description
          "Specifies that the Prefix-SID is to be treated as a Node-SID
          by setting the N-flag in the advertised Prefix-SID TLV in the
          IGP";
      }
```

```
        leaf last-hop-behavior {
          type enumeration {
            enum "EXPLICIT-NULL" {
              description
                "Specifies that the explicit null label is to be used
                when the penultimate hop forwards a labelled packet to
                this Prefix-SID";
            }
            enum "UNCHANGED" {
              description
                "Specicies that the Prefix-SID's label value is to be
                left in place when the penultimate hop forwards to this
                Prefix-SID";
            }
            enum "PHP" {
              description
                "Specicies that the penultimate hop should pop the
                Prefix-SID label before forwarding to the eLER";
            }
          }
          description
            "Configuration relating to the LFIB actions for the
            Prefix-SID to be used by the penultimate-hop";
        }
      }


      grouping igp-tunnel-sr {
        description "defintiions for SR-signaled, IGP-based LSP tunnel
        types";

        container tunnel {
          description "contains configuration stanzas for different LSP
          tunnel types (P2P, P2MP, etc.)";

          leaf tunnel-type {
            type mplst:tunnel-type;
            description "specifies the type of LSP, e.g., P2P or P2MP";
          }

          container p2p-lsp {
            when "tunnel-type = 'P2P'" {
              description "container active when LSP tunnel type is
              point to point";
            }
            description "properties of point-to-point tunnels";

            list fec {
```

```
            key "fec-address";
            uses sr_fec-address_config;

            description
              "List of FECs that are to be originated as SR LSPs";

            container config {
              description
                "Configuration parameters relating to the FEC to be
                advertised by SR";
              uses sr_fec-address_config;
            }
            container state {
              config false;
              description
                "Operational state relating to a FEC advertised by SR";
              uses sr_fec-address_config;
            }
            container prefix-sid {
              description
                "Parameters relating to the Prefix-SID
                used for the originated FEC";

              container config {
                description
                  "Configuration parameters relating to the Prefix-SID
                  used for the originated FEC";
                uses sr_fec-prefix-sid_config;
              }
              container state {
                config false;
                description
                  "Operational state parameters relating to the
                  Prefix-SID used for the originated FEC";
                uses sr_fec-prefix-sid_config;
              }
            }
          }
        }
      }
    }

    grouping igp-lsp-sr-setup {
      description "grouping for SR-IGP path setup for IGP-congruent
      LSPs";

      container segment-routing {
```

```
          presence "Presence of this container sets the LSP to use
          SR signaling";

          description "segment routing signaling extensions for
          IGP-confgruent LSPs";

          uses igp-tunnel-sr;

        }
      }

      // data definition statements

      // augment statements

      // rpc statements

      // notification statements

    }
                    <CODE ENDS>


                    <CODE BEGINS> file openconfig-mpls-ldp.yang
    module openconfig-mpls-ldp {

      yang-version "1";

      // namespace
      namespace "http://openconfig.net/yang/ldp";

      prefix "ldp";

      // import some basic types
      import ietf-inet-types { prefix inet; }
      import openconfig-mpls-types { prefix mplst; }


      // meta
      organization "OpenConfig working group";

      contact
        "OpenConfig working group
        netopenconfig@googlegroups.com";

      description
        "Configuration of Label Distribution Protocol global and LSP-
```

```
       specific parameters for IGP-congruent LSPs";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements

      grouping ldp-global {
       description "global LDP signaling configuration";

       container ldp {
         description "LDP global signaling configuration";

         container timers {
           description "LDP timers";
         }
       }
     }


     grouping igp-tunnel-ldp {
       description "common defintiions for LDP-signaled LSP tunnel
       types";

       container tunnel {
         description "contains configuration stanzas for different LSP
         tunnel types (P2P, P2MP, etc.)";

         leaf tunnel-type {
           type mplst:tunnel-type;
             description "specifies the type of LSP, e.g., P2P or P2MP";
         }

         leaf ldp-type {
           type enumeration {
             enum BASIC {
               description "basic hop-by-hop LSP";
```

```
            }
            enum TARGETED {
              description "tLDP LSP";
            }
          }
          description "specify basic or targeted LDP LSP";
        }

        container p2p-lsp {
          when "tunnel-type = 'P2P'" {
            description "container active when LSP tunnel type is
            point to point";
          }

          description "properties of point-to-point tunnels";

          leaf-list fec-address {
            type inet:ip-prefix;
            description "Address prefix for packets sharing the same
            forwarding equivalence class for the IGP-based LSP";
          }
        }

        container p2mp-lsp {
          when "tunnel-type = 'P2MP'" {
            description "container is active when LSP tunnel type is
            point to multipoint";
          }

          description "properties of point-to-multipoint tunnels";

          // TODO: specify group/source, etc.

        }

        container mp2mp-lsp {
          when "tunnel-type = 'MP2MP'" {
            description "container is active when LSP tunnel type is
            multipoint to multipoint";
          }

          description "properties of multipoint-to-multipoint tunnels";

          // TODO: specify group/source, etc.

        }
      }
    }
```

```
     grouping igp-lsp-ldp-setup {
       description "grouping for LDP setup attributes";

       container ldp {

         presence "Presence of this container sets the LSP to use
         LDP signaling";

         description "LDP signaling setup for IGP-congruent LSPs";

         // include tunnel (p2p, p2mp, ...)

         uses igp-tunnel-ldp;

       }
     }

     // data definition statements

     // augment statements

     // rpc statements

     // notification statements

   }
                   <CODE ENDS>



                   <CODE BEGINS> file openconfig-mpls-igp.yang
   submodule openconfig-mpls-igp {

     yang-version "1";

     belongs-to "openconfig-mpls" {
       prefix "mpls";
     }


     // import some basic types
     import openconfig-mpls-ldp { prefix ldp; }
     import openconfig-mpls-sr { prefix sr; }



     // meta
     organization "OpenConfig working group";
```

```
     contact
       "OpenConfig working group
       netopenconfig@googlegroups.com";

     description
       "Configuration generic configuration parameters for IGP-congruent
       LSPs";

     revision "2015-07-04" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements


     grouping igp-lsp-common {
       description "common definitions for IGP-congruent LSPs";

       // container path-attributes {
       //  description "general path attribute settings for IGP-based
       //  LSPs";

       //}

     }


     grouping igp-lsp-setup {
       description "signaling protocol definitions for IGP-based LSPs";

       container path-setup-protocol {
         description "select and configure the signaling method for
             the LSP";

         // uses path-setup-common;
         uses ldp:igp-lsp-ldp-setup;
         uses sr:igp-lsp-sr-setup;
       }
```

```
    }


    // data definition statements

    // augment statements

    // rpc statements

    // notification statements

  }
```

<CODE ENDS>

<CODE BEGINS> file openconfig-mpls-static.yang

```
  submodule openconfig-mpls-static {

    yang-version "1";

    belongs-to "openconfig-mpls" {
      prefix "mpls";
    }

    // import some basic types
    import openconfig-mpls-types {prefix mplst; }
    import ietf-inet-types { prefix inet; }


    // meta
    organization "OpenConfig working group";

    contact
      "OpenConfig working group
      netopenconfig@googlegroups.com";

    description
      "Defines static LSP configuration";

    revision "2015-07-04" {
      description
        "Initial revision";
      reference "TBD";
    }

    // extension statements
```

```
     // feature statements

     // identity statements

     // typedef statements

     // grouping statements

     grouping static-lsp-common {
       description "common definitions for static LSPs";

       leaf next-hop {
         type inet:ip-address;
         description "next hop IP address for the LSP";
       }

       leaf incoming-label {
         type mplst:mpls-label;
         description "label value on the incoming packet";
       }

       leaf push-label {
         type mplst:mpls-label;
         description "label value to push at the current hop for the
         LSP";
       }
     }

     grouping static-lsp-main {
       description "grouping for top level list of static LSPs";


       list label-switched-path {
         key name;
         description "list of defined static LSPs";

         leaf name {
           type string;
           description "name to identify the LSP";
         }

         // TODO: separation into ingress, transit, egress may help
         // to figure out what exactly is configured, but need to
         // consider whether implementations can support the
         // separation
         container ingress {
           description "Static LSPs for which the router is an
             ingress node";
```

```
          uses static-lsp-common;
        }

        container transit {
          description "static LSPs for which the router is a
            transit node";

          uses static-lsp-common;
        }

        container egress {
          description "static LSPs for which the router is a
            egress  node";

          uses static-lsp-common;
        }
      }
    }

    // data definition statements

    // augment statements

    // rpc statements

    // notification statements

  }
                  <CODE ENDS>
```

7. Contributing Authors

The following people contributed significantly to this document and are listed below:

Ina Minei
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US
Email: inaminei@google.com

Anees Shaikh
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: aashaikh@google.com

Phil Bedard
Cox Communications
Atlanta, GA 30319
US
Email: phil.bedard@cox.com

8. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Deepak Bansal, Nabil Bitar, Feihong Chen, Mazen Khaddam.

9. References

[I-D.ietf-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-00 (work in progress), July 2015.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., rjs@rob.sh, r., Tantsura, J., and E. Crabbe, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-02 (work in progress), October 2015.

[I-D.openconfig-netmod-model-structure]
Shaikh, A., Shakir, R., D'Souza, K., and L. Fang, "Operational Structure and Organization of YANG Models", draft-openconfig-netmod-model-structure-00 (work in progress), March 2015.

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

[I-D.shaikh-idr-bgp-model]
Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Alex, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-shaikh-idr-bgp-model-02 (work in progress), June 2015.

[I-D.shakir-rtgwg-sr-performance-engineered-lsps]
Shakir, R., Vernals, D., and A. Capello, "Performance Engineered LSPs using the Segment Routing Data-Plane", draft-shakir-rtgwg-sr-performance-engineered-lsps-00 (work in progress), July 2013.

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

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, DOI 10.17487/RFC5443, March 2009, <http://www.rfc-editor.org/info/rfc5443>.

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

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

Authors' Addresses

Joshua George
Google
1600 Amphitheatre Pkwy
Mountain View, CA 94043
US

Email: jgeorge@google.com

Luyuan Fang
Microsoft
15590 NE 31st St
Redmond, WA 98052
US

Email: lufang@microsoft.com

Eric Osborne
Level 3

Email: eric.osborne@level3.com

Rob Shakir
Jive Communications, Inc.
1275 West 1600 North, Suite 100
Orem, UT 84057

Email: rjs@rob.sh