Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

J. George
Google
L. Fang
Microsoft
E. Osborne
Level 3
R. Shakir
BT

March 9, 2015

MPLS / TE Model for Service Provider Networks
draft-openconfig-mpls-consolidated-model-00

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 September 10, 2015.

Copyright Notice

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 routers supplied by multiple vendors. The model is also intended to be readily mapped to existing implementations, however, 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 first version of the model 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 siginaling protocol-specific information under it where applicable.

This model does not (in the current iteration) 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.

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. In addition, operational state data for MPLS LSPs is not included in this version, but will be added in the next revision.

2. Model overview

The overall MPLS model is defined across several YANG modules and submodules but at a high level is organized into 3 main sections:

- global -- configuration affecting MPLS 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 signaling-protocols
      |     ...
      +--rw lsps
```

...

2.1. MPLS global overview

The global section of the framework provides configuration control for MPLS 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 specific device interfaces where MPLS forwarding will be permitted. Timers are also specified which determine the length of time an LSP must be present before considered viable for forwarding use (mpls-lsp-install-delay), and the length of

time between LSP teardown and removal of the LSP from the network element's forwarding information base (mpls-lsp-cleanup- delay). Also specified are the name to value mappings of MPLS administrative groups (mpls-admin-groups).

```
+--rw mpls!
   +--rw global
      +--rw mpls-interfaces* [interface-name]
      |  +--rw interface-name            string
      |  +--rw interface-admin-groups*  -> /mpls/global/...
      +--rw mpls-lsp-install-delay?   uint16
      +--rw mpls-lsp-cleanup-delay?   uint16
      +--rw mpls-admin-groups* [admin-group-name]
         +--rw admin-group-name     string
         +--rw admin-group-value?   uint32
```

2.2. Signaling protocol overview

The signaling protocol section of the framework provides configuration elements for configuring three major methods of signaling label switched paths: RSVP, segment routing, and label distribution protocol (LDP). Configuration of RSVP is centered around interfaces on the device which participate in the protocol. A key focus is to expose common RSVP configuration parameters which are used to enhance scale and reliability (refresh-reduction, refresh-reduction-reliable). From the same principles, configuration is available to configure the sensitivity of IGP flooding events upon bandwidth change on an RSVP interface (ted-update-threshold). Also specified are options to configure RSVP soft-preemption (soft-

preemption), and for MPLS protection (link-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.


```
     +--rw mpls!
     +--rw signaling-protocols
        +--rw rsvp
        |  +--rw interfaces* [interface-name]
        |  |  +--rw interface-name                string
        |  |  +--rw hello-interval?               uint16
        |  |  +--rw refresh-reduction?            boolean
        |  |  +--rw refresh-reduction-reliable?   boolean
        |  |  +--rw subscription?                 mplst:percentage
        |  |  +--rw ted-update-threshold?         mplst:percentage
        |  |  +--rw link-protection!
        |  |     +--rw link-protection-only?       boolean
        |  |     +--rw bypass-optimize-interval?   uint16
        |  +--rw soft-preemption!
        |     +--rw soft-preemption-timeout?   uint16
        +--rw segment-routing
        +--rw ldp
           +--rw timers
```

2.3. 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 is shown below:


```
+--rw mpls!
      +--rw lsps
         +--rw constrained-path
            +--rw path-information
            |  +--rw path* [path-name]
            |     +--rw path-name    string
            |     +--rw hop* [address]
            |           ...
            +--rw label-switched-path* [signaled-name]
               +--rw signaled-name              string
               +--rw lsp-description?           string
               +--rw path-computation-method
               |  +--rw path-computation?   identityref
               |  +--rw explicit-path
               |  |     ...
```

```
                |  +--rw queried-path
                |  |     ...
                |  +--rw locally-computed
                |        ...
                +--rw path-attributes
                |  +--rw metric?                        te-metric-type
                |  +--rw bandwidth
                |  |     ...
                |  +--rw lsp-placement-constraints
                |  |     ...
                |  +--rw protection
                |        ...
                +--rw path-setup
                   +--rw rsvp!
                   |     ...
                   +--rw segment-routing!
                         ...
```

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. The LSP

will specify the path setup method as RSVP inside the path-setup container, indicating the LSP desires RSVP signaling. The LSP would be configured as locally-computed under the path-computation-method container, specifying the use of cspf (use-cspf). Additional attributes such as bandwidth (explicit or auto), protection style, and placement constraints are available in the path-attributes container.

The structure to support these is shown in the constrained-path portion of the data model below:


```
+--rw mpls!
```

```
+--rw lsps
   +--rw constrained-path
      +--rw path-information
      |  +--rw path* [path-name]
      |     +--rw path-name    string
      |     +--rw hop* [address]
      |           ...
      +--rw label-switched-path* [signaled-name]
         +--rw signaled-name              string
         +--rw lsp-description?           string
         +--rw path-computation-method
         |  +--rw path-computation?   identityref
         |  +--rw explicit-path
         |  |     ...
         |  +--rw queried-path
         |  |     ...
         |  +--rw locally-computed
         |        ...
         +--rw path-attributes
         |  +--rw metric?                       te-metric-type
         |  +--rw bandwidth
         |  |     ...
         |  +--rw lsp-placement-constraints
         |  |     ...
         |  +--rw protection
         |        ...
         +--rw path-setup
            +--rw rsvp!
            |     ...
            +--rw segment-routing!
                  ...
```

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.

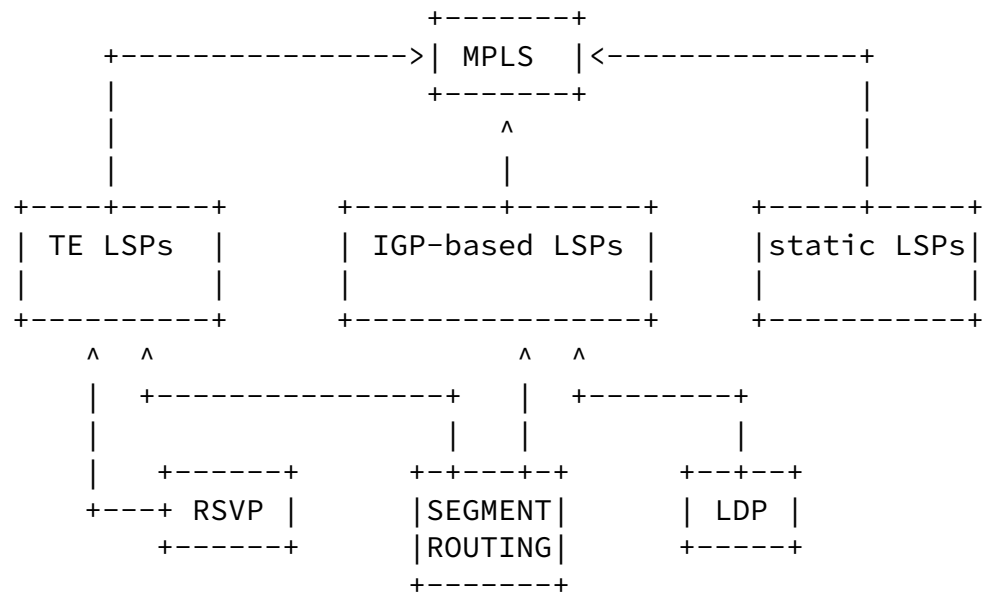
Most of the data elements in the configuration model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

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].

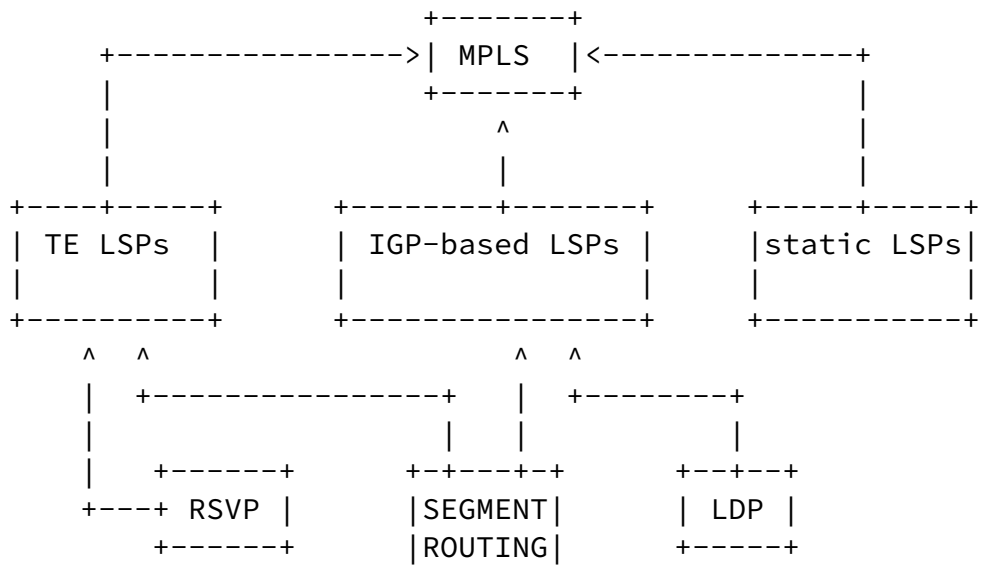
6.1. MPLS base modules

```
<CODE BEGINS> file mpls.yang
module mpls {

  yang-version "1";

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

  prefix "mpls";


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

  // include submodules
  include mpls-te;
  include mpls-igp;
  include 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 "2014-12-12" {
    description
      "Initial revision";
    reference "TBD";
  }

  // extension statements

  // feature statements

  // identity statements


  // grouping statements

  grouping path-setup-common {
    description "common definitions for all signaling protocols";

    // TODO: not clear we really need this
    leaf path-setup-type {
```



```
      type identityref {
        base mplst:path-setup-protocol;
      }
      description "path setup protocol to use with the LSP";
    }
  }

  grouping mpls-administrative-groups {
    description
      "global level definitions for MPLS link admin groups";

    list mpls-admin-groups {

      key admin-group-name;
      description "configuration of value to name mapping for mpls
      affinities/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-global {
      description "global level definitions for MPLS protocol
      operation";

      // TODO: this should be made a reference to an interface in the
      // interfaces model
      // TODO - should probably have as key the interface name, also
      // need an easy way to specify all interfaces and to exclude
      // interfaces.
      list mpls-interfaces {

        key interface-name;
        description "interfaces for which MPLS is enabled";

        leaf interface-name {
          type string;
```



```
          description "reference to interface name";
          // TODO: add ref to interface model
        }

        leaf-list interface-admin-groups {
          type leafref {
            path "/mpls:mpls/mpls:global/mpls:mpls-admin-groups/"
            + "mpls:admin-group-name";
          }
          description
            "list of configured admin-groups on the interface";
```

```
        }
      }

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

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

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

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

      container global {
        description "general mpls configuration across LSP and tunnel
        types";

        uses mpls-global;
        uses mpls-administrative-groups;

      }

      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 mpls-te-global;

        uses path-definitions;

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

          uses te-lsp-common;
          uses te-lsp-setup;
        }
      }

      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 mpls-types.yang
  module 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 "2015-02-01" {
      description
        "Initial revision";
      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";
  }

  // typedef statements

  typedef percentage {
    type uint8 {
      range "0..100";
    }
    description
      "Integer indicating a percentage value";
  }


  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)";
    }

    // grouping statements

    // data definition statements

    // augment statements

    // rpc statements

    // notification statements
```



```
  }
  <CODE ENDS>
```

6.2. MPLS LSP submodules

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

    yang-version "1";

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


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


    // 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 "2014-07-07" {
    description
      "Initial revision";
    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";
    }

   typedef mpls-protection-style {
     type enumeration {
       enum UNPROTECTED {
         description "no protection is desired for the lsp";
       }
       enum LINK-PROTECTION-REQUESTED {
         description "link protection is desired for the lsp";
       }
       enum LINK-NODE-PROTECTION-REQUESTED {
         description "node and link protection is desired for the lsp";
       }
     }
     default UNPROTECTED;
     description
      "Specifies the protection type for the LSP";
   }

    // grouping statements

    grouping te-lsp-common {
```



```
      description "common definitions for traffic-engineered LSPs";

      leaf signaled-name {
            type string;
            description "LSP name, also carried in signaling
```

```
        messages when appropriate";
      }

      leaf lsp-description {
          type string;
          description "optional text description for the LSP";
      }

      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;
      }

      container path-attributes {
        description "general path attribute settings for TE-LSP
        tunnels";

        // XXX - no, this is also there for LDP - also removed the
        // reference to "igp metric" as this is going to be confusing,
        // unless we mandate for the LSP to have the same metric as the
        // Igp, which is going to be hard with some vendors
        // implementations.
        leaf metric {
          type te-metric-type;
          description "LSP metric, either explicit or IGP";
        }

        container bandwidth {
          description "bandwidth specification for the LSP";

          choice lsp-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 {
  container auto-bandwidth {
    presence "presence of this container indicates
    auto-bandwidth is enabled for the LSP";

    description "configure auto-bandwidth operation in
    which devices automatically adjust bandwidth to meet
    requirements";

    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 mplst: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 {

      presence "presence of this element indicates overflow
```

```
                is configured for the lsp";

                description "configuration for mpls lsp bandwidth
                overflow adjustment";

                leaf overflow-threshold {
                  type mplst: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";
                }
              }

              container underflow {
                presence
                  "presence of this element indicates underflow
                  is configured for the lsp";

                description
                  "configuration for mpls lsp bandwidth
                  underflow adjustment";

                leaf underflow-threshold {
                  type mplst: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";
                }
              }
            }
```

```
        }
      }
    }

    container lsp-placement-constraints {
      description
```



```
        "constraints on lsp routing such as admin-groups";

      container admin-groups {
        description
          "Include/Exclude constraints for link affinities";

        list exclude-groups {

          key admin-group-name;

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

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

        list include-all-groups {

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

          leaf admin-group-name {
            type leafref {
              path "/mpls/global/mpls-admin-groups/" +
                "admin-group-name";
            }
```

```
                description
                  "name of the admin group -- references a defined
                  admin group";
              }
            }

            list include-any-groups {

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

              leaf admin-group-name {
```



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

        container protection {
          description "failure protection properties for the LSP";

          leaf protection-style-requested {
            type mpls-protection-style {
            }
            description "style of mpls frr protection desired. both
            facility backup and one-to-one are options";
          }
        }
      }
    }
    // TODO - note that this is only currently defined for
    // RSVP-like entities
    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/"
                + "path-information/path/path-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 path-definitions {
      description "describe path configuration for specifying LSP
      hops";

      container path-information {
        description "common information for MPLS path definition";

        list path {
          key path-name;
          description "specification of LSP path";

          leaf path-name {
            type string;
            description "identifier for the LSP path";
          }
```



```
          list hop {
            key address;
            description "specification of the strict and loose hops in
              the path";

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

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

```
        }
      }
    }
  }

  grouping te-lsp-setup {
    description "signaling protocol configuration for traffic
    engineered LSPs";

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

      // uses path-setup-common;
      uses rsvp:te-lsp-rsvp-setup;
      uses sr:te-lsp-sr-setup;
    }
  }

  grouping mpls-te-global {
    description "global level defintions for mpls traffic
    engineered LSPs";


  }

  // data definition statements

  // augment statements

  // rpc statements

  // notification statements
}
```


<CODE ENDS>

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

  yang-version "1";
```

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


  // import some basic types
  import mpls-ldp { prefix ldp; }
  import 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 "2014-07-07" {
    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 mpls-static.yang
submodule mpls-static {

  yang-version "1";

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

  // import some basic types
  import 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 "2015-02-01" {
       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>
```

6.3. MPLS signaling protocol modules

```
<CODE BEGINS> file mpls-rsvp.yang
module 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 mpls-types { prefix mplst; }


  // meta
  organization "OpenConfig working group";

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

  description
```

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

  revision "2014-07-07" {
    description
      "Initial revision";
    reference "TBD";
  }

  // extension statements

  // feature statements

  // identity statements

  // typedef statements

  // grouping statements
```



```
  grouping rsvp-global {
    description "Global RSVP protocol configuration";

    container rsvp {
      description "RSVP global signaling protocol configuration";

      // interfaces, bw percentages, hello timers, etc goes here";

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

        // TODO: update to interface ref -- move to separate
        // augmentation.
        leaf interface-name {
          type string;
          description "references a configured IP interface";
        }

        leaf hello-interval {
          type uint16 {
            range 0..60000;
```

```
        }
        units milliseconds;
        description "set the interval in ms between RSVP hello
        messages";
      }

      // TODO: confirm that the described semantics are supported
      // on various implementations. Finer grain configuration
      // will be vendor-specific
      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";
      }

      leaf refresh-reduction-reliable {
        type boolean;
        default true;
        description "enables RSVP refresh reduction reliable
        delivery and message_ID";
        reference "RFC 2961 RSVP Refresh Overhead Reduction
        Extensions";
```



```
      }

      leaf subscription {
        type mplst:percentage;
        description "percentage of the interface bandwidth that
        RSVP can reserve";
      }

      leaf ted-update-threshold {
        type mplst:percentage;
        description "percentage of interface bandwidth change
        that triggers an update event into the IGP TED";
      }
```

```
        // TODO: this may need to be moved to common TE LSP config
        container link-protection {
          description "link-protection (NHOP) related configuration";
          presence "presence of this container indicates facility
           protection is configured on the interface";

          leaf link-protection-only {
            type boolean;
            default false;
            description "disables node protection on this interface,
             and forces only link protection";
          }

          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.
        }
      }

      container soft-preemption {
        description "options relating to RSVP soft preemption";
        presence "presence of this container enables RSVP soft
          preemption on a node";

        leaf soft-preemption-timeout {
          type uint16 {
```



```
            range 0..300;
          }
          default 0;
          description "timeout value for soft preemption to revert
            to hard preemption";
        }
      }
    }
```

```
    }
    grouping te-tunnel-rsvp {
      description "defintiions for RSVP-signaled LSP tunnel types,
      .e.g., applicable to point-to-point LSPs";

      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";

          leaf destination {
            type inet:ip-address;
            description "destination egress node for the LSP";
          }

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

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



```
          }
```

```
      } // p2p-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";

        leaf-list destination {
          type inet:ip-address;
          description "list of destinations / egress nodes for the
          multipoint LSP tunnel";
        }
      }
    }
  }

  grouping te-lsp-rsvp-setup {
    description "data definitions for RSVP-signalled LSPs";

    container rsvp {

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

      description "Configuration for RSVP-signalled TE LSPs";


      container path-specification {
        description "Definition of primary/backup paths for purpose
        of signaling the LSP";
      }

      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";
      }

      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 that other LSPs
          will not preempt the LSPs once established";
        }

        leaf retry-timer {
          type uint16 {
            range 1..600;
          }
          units seconds;
          description "sets the time between attempts to establish the
          LSP";
        }
        //TODO: add other RSPV parms: optimize delay, switchover delay,
        // etc.

        // include tunnel (p2p, p2mp, etc).
        uses te-tunnel-rsvp;
      }
    }

    // data definition statements

    // augment statements

    // rpc statements

    // notification statements

  }
  <CODE ENDS>


  <CODE BEGINS> file mpls-sr.yang
  module 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 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 "2014-07-07" {
       description
         "Initial revision";
       reference "TBD";
     }

     // extension statements

     // feature statements

     // identity statements

     // typedef statements

     // grouping statements

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

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

       }
     }
```

```
grouping te-tunnel-sr {
  description "defintiions for SR-signaled LSP tunnel types,
  .e.g., applicable to point-to-point LSPs";

  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 "Config for point-to-point tunnels";

      // fill out the configuration details per segment
      // routing requrements
    }
  }
}


grouping te-lsp-sr-setup {
  description "data definitions for SR signaling";

  container segment-routing {

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

    description "Configuration for signaling SR-based TE LSPs";

    uses te-tunnel-sr;
  }
}
```

```
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";

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

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 mpls-ldp.yang
    module 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 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 "2014-07-07" {
    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>
```

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

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

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

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

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Shakir, R., Tantsura, J., and E. Crabbe, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-00 (work in progress), December 2014.

[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.

[RFC5443] Jork, M., Atlas, A., and L. Fang, "LDP IGP Synchronization", RFC 5443, March 2009.

Authors' Addresses

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

Email: jgeorge@google.com

Luyuan Fang
Microsoft
205 108th Ave. NE, Suite 400
Bellevue, WA
US

Email: lufang@microsoft.com

Eric Osborne
Level 3

Email: eric.osborne@level3.com

Rob Shakir
BT
pp. C3L, BT Centre
81, Newgate Street
London EC1A 7AJ
UK

Email: rob.shakir@bt.com
URI: http://www.bt.com/