MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 19, 2017

K. Raza
R. Asati
Cisco Systems, Inc.

X. Liu
Ericsson

S. Esale
Juniper Networks

X. Chen
Huawei Technologies

H. Shah
Ciena Corporation

August 18, 2016

YANG Data Model for MPLS LDP and mLDP
draft-ietf-mpls-ldp-mldp-yang-00

Abstract

This document describes a YANG data model for Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP) and Multipoint LDP (mLDP).

Status of This Memo

Table of Contents

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is one of the network management protocols that defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML tree format, and is used as a data modelling language for the NETCONF.

This document introduces a YANG data model for MPLS Label Distribution Protocol (LDP) [RFC5036] and Multipoint LDP (mLDP) [RFC6388]. For LDP, it also covers LDP IPv6 [RFC7552] and LDP capabilities [RFC5561].

The data model is defined for following constructs that are used for managing the protocol:

o Configuration

o Operational State

o Executables (Actions)

o Notifications

This document is organized to define the data model for each of the above constructs (configuration, state, action, and notifications) in the sequence as listed earlier. Given that mLDP is tightly coupled with LDP, mLDP data model is defined under LDP tree and in the same sequence as listed above.

2. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In this document, the word "IP" is used to refer to both IPv4 and IPv6, unless otherwise explicitly stated. For example, "IP address family" means and be read as "IPv4 and/or IPv6 address family"

3. LDP YANG Model

3.1. Overview

This document defines a new module named "ietf-mpls-ldp" for LDP/mLDP data model where this module augments /rt:routing/rt:control-plane-protocols that is defined in [I-D.ietf-netmod-routing-cfg].

There are four main containers in "ietf-mpls-ldp" module as follows:

- Read-Write parameters for configuration (Discussed in Section 3.2)

- Read-only parameters for operational state (Discussed in Section 3.3)

- Notifications for events (Discussed in Section 3.4)

- RPCs for executing commands to perform some action (Discussed in Section 3.5)

For the configuration and state data, this model follows the similar approach described in [I-D.openconfig-netmod-opstate] to represent the configuration (intended state) and operational (applied and derived) state. This means that for every configuration (rw) item, there is an associated (ro) item under "state" container to represent the applied state. Furthermore, protocol derived state is also kept under "state" tree corresponding to the protocol area (discovery, peer etc.). [Ed note: This document will be (re-)aligned with [I-D.openconfig-netmod-opstate] once that specification is adopted as a WG document]

Following diagram depicts high level LDP yang tree organization and hierarchy:

```
       module: ietf-mpls-ldp
            +-- rw routing
              +-- rw control-plane-protocols
                 +-- rw mpls-ldp
                     +-- rw global
                     |   +-- rw config
                     |       +-- rw ...
                     |   +-- ro state
                     |       +-- ro ...
                     |       .
                     +-- rw ...
                     |
                     |
                     +-- rw ...
                     ...

       rpcs:
          +-- x mpls-ldp-rpc
          +-- x . . . . .

       notifications:
          +--- n mpls-ldp-notif
          +--- n ...
```

Figure 1

Before going into data model details, it is important to take note of the following points:

o This module aims to address only the core LDP/mLDP parameters as per RFC specification, as well as some widely used and deployed non-RFC features (such as label policies, session authentication etc). Any vendor specific feature should be defined in a vendor-specific augmentation of this model.

o Multi-topology LDP [RFC7307] and Multi-topology mLDP [I-D.iwijnand-mpls-mldp-multi-topology] are beyond the scope of this document.

o This module does not cover any applications running on top of LDP and mLDP, nor does it cover any OAM procedures for LDP and mLDP.

o This model is a VPN Forwarding and Routing (VRF)-centric model. It is important to note that [RFC4364] defines VRF tables and default forwarding tables as different, however from a yang modelling perspective this introduces unnecessary complications, hence we are treating the default forwarding table as just another VRF.

- o A "network-instance" as defined in [I-D.rtgyangdt-rtgwg-ni-model] refers to a VRF instance (both default and non-default) within the scope of this model.

- o This model supports two address-families, namely "ipv4" and "ipv6".

- o This model assumes platform-wide label space (i.e. label space Id of zero). However, when Upstream Label assignment [RFC6389] is in use, an upstream assigned label is looked up in a Context-Specific label space as defined in [RFC5331].

- o The label and peer policies (including filters) are defined using a prefix-list. When used for a peer policy, the prefix refers to the LSR Id of the peer. The prefix-list is referenced from routing-policy model as defined in [I-D.ietf-rtgwg-policy-model].

- o The use of grouping (templates) for bundling and grouping the configuration items is not employed in current revision, and is a subject for consideration in future.

- o This model uses the terms LDP "neighbor"/"adjacency", "session", and "peer" with the following semantics:

  * Neighbor/Adjacency: An LDP enabled LSR that is discovered through LDP discovery mechanisms.

  * Session: An LDP neighbor with whom a TCP connection has been established.

  * Peer: An LDP session which has successfully progressed beyond its initialization phase and is either already exchanging the bindings or is ready to do so.

  It is to be noted that LDP Graceful Restart mechanisms defined in [RFC3478] allow keeping the exchanged bindings for some time after a session goes down with a peer. We call such a state -- i.e. keeping peer bindings without established or recovered peering -- a "stale" peer. When used in this document, the above terms will refer strictly to the semantics and definitions defined for them.

A graphical representation of LDP YANG data model is presented in Figure 3, Figure 5, Figure 11, and Figure 12. Whereas, the actual model definition in YANG is captured in Section 6.

While presenting the YANG tree view and actual .yang specification, this document assumes the reader is familiar with the concepts of YANG modeling, its presentation and its compilation.

3.2. Configuration

This specification defines the configuration parameters for base LDP as specified in [RFC5036] and LDP IPv6 [RFC7552]. Moreover, it incorporates provisions to enable LDP Capabilities [RFC5561], and defines some of the most significant and commonly used capabilities such as Typed Wildcard FEC [RFC5918], End-of-LIB [RFC5919], and LDP Upstream Label Assignment [RFC6389].

This specification supports VRF-centric configuration. For implementations that support protocol-centric configuration, with provision for inheritance and items that apply to all vrfs, we recommend an augmentation of this model such that any protocol-centric or all-vrf configuration is defined under their designated containers within the standard network-instance (please see Section 3.2.2)

This model augments /rt:routing/rt:control-plane-protocols that is defined in [I-D.ietf-netmod-routing-cfg]. For LDP interfaces, this model refers the MPLS interface as defined under MPLS base specification [I-D.saad-mpls-base-yang]. Furthermore, as mentioned earlier, the configuration tree presents read-write intended configuration leave/items as well as read-only state of the applied configuration. The former is listed under "config" container and latter under "state" container.

Following is high-level configuration organization for LDP/mLDP:

```
       module: ietf-mpls-ldp
          +-- routing
           +-- control-plane-protocols
                 +-- mpls-ldp
                     +-- global
                     |    +-- ...
                     |    +-- ...
                     |    +-- address-family* [afi]
                     |        +-- . . .
                     |        +-- . . .
                     |    +-- discovery
                     |        +-- . . .
                     +-- peers
                          +-- ...
                          +-- ...
```

Figure 2

Given the configuration hierarchy, the model allows inheritance such that an item in a child tree is able to derive value from a similar or related item in one of the parent. For instance, hello holdtime can be configured per-VRF or per-VRF-interface, thus allowing inheritance as well flexibility to override with a different value at any child level.

Following is a simplified graphical representation of the data model for LDP configuration

```
   +--rw mpls-ldp!
      +--rw global
      |  +--rw config
      |  |  +--rw capability
      |  |  |  +--rw end-of-lib {capability-end-of-lib}?
      |  |  |  |  +--rw enable?   boolean
      |  |  |  +--rw typed-wildcard-fec {capability-typed-wildcard-fec}?
      |  |  |  |  +--rw enable?   boolean
      |  |  |  +--rw upstream-label-assignment {capability-upstream-label-assign
ment}?
      |  |  |     +--rw enable?   boolean
      |  |  +--rw graceful-restart
      |  |  |  +--rw enable?               boolean
      |  |  |  +--rw helper-enable?        boolean {graceful-restart-helper-mod
e}?
      |  |  |  +--rw reconnect-time?       uint16
      |  |  |  +--rw recovery-time?        uint16
      |  |  |  +--rw forwarding-holdtime?  uint16
      |  |  +--rw igp-synchronization-delay?  uint16
      |  |  +--rw lsr-id?                     yang:dotted-quad
      |  +--rw address-family* [afi]
      |  |  +--rw afi       ldp-address-family
      |  |  +--rw config
      |  |     +--rw enable?         boolean
      |  |     +--rw label-policy
      |  |     |  +--rw independent-mode
      |  |     |  |  +--rw assign {policy-label-assignment-config}?
      |  |     |  |  |  +--rw (prefix-option)?
      |  |     |  |  |     |  +--rw prefix-list?        prefix-list-ref
      |  |     |  |  |        +--rw host-routes-only?   boolean
      |  |     |  |  +--rw advertise
      |  |     |  |  |  +--rw explicit-null
      |  |     |  |  |  |  +--rw enable?        boolean
      |  |     |  |  |  |  +--rw prefix-list?   prefix-list-ref
      |  |     |  |  |  +--rw prefix-list?     prefix-list-ref
      |  |     |  |  +--rw accept
      |  |     |  |     +--rw prefix-list?   prefix-list-ref
      |  |     |  +--rw ordered-mode {policy-ordered-label-config}?
      |  |     |     +--rw egress-lsr
      |  |     |     |  +--rw prefix-list?   prefix-list-ref
      |  |     |     +--rw advertise
      |  |     |     |  +--rw prefix-list?   prefix-list-ref
      |  |     |     +--rw accept
      |  |     |        +--rw prefix-list?   prefix-list-ref
      |  |     +--rw ipv4
      |  |     |  +--rw transport-address?   inet:ipv4-address
      |  |     +--rw ipv6
      |  |        +--rw transport-address?   inet:ipv6-address
      |  +--rw discovery
      |  |  +--rw interfaces
      |  |  |  +--rw config
      |  |  |  |  +--rw hello-holdtime?   uint16
      |  |  |  |  +--rw hello-interval?   uint16
      |  |  |  +--rw interface* [interface]
      |  |  |     +--rw interface         mpls-interface-ref
      |  |  |     +--rw config
      |  |  |     |  +--rw hello-holdtime?              uint16
      |  |  |     |  +--rw hello-interval?              uint16
      |  |  |     |  +--rw igp-synchronization-delay?   uint16 {per-interface-ti
mer-config}?
      |  |  |     +--rw address-family* [afi]
      |  |  |        +--rw afi       ldp-address-family
      |  |  |        +--rw config
      |  |  |           +--rw enable?   boolean
      |  |  |           +--rw ipv4
      |  |  |           |  +--rw transport-address?   union
      |  |  |           +--rw ipv6
      |  |  |              +--rw transport-address?   union
      |  |  +--rw targeted
```

```
   |  |     +--rw config
   |  |     |  +--rw hello-holdtime?   uint16
   |  |     |  +--rw hello-interval?   uint16
   |  |     |  +--rw hello-accept {policy-extended-discovery-config}?
   |  |     |     +--rw enable?          boolean
   |  |     |     +--rw neighbor-list?   neighbor-list-ref
   |  |     +--rw address-family* [afi]
   |  |        +--rw afi      ldp-address-family
   |  |        +--rw ipv4
   |  |        |  +--rw target* [adjacent-address]
   |  |        |     +--rw adjacent-address    inet:ipv4-address
   |  |        |     +--rw config
   |  |        |        +--rw enable?          boolean
   |  |        |        +--rw local-address?   inet:ipv4-address
   |  |        +--rw ipv6
   |  |           +--rw target* [adjacent-address]
   |  |              +--rw adjacent-address    inet:ipv6-address
   |  |              +--rw config
   |  |                 +--rw enable?          boolean
   |  |                 +--rw local-address?   inet:ipv6-address
   |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
   |  |  +--rw interfaces
   |  |     +--rw interface* [interface]
   |  |        +--rw interface         mpls-interface-ref
   |  |        +--rw address-family* [afi]
   |  |           +--rw afi       ldp-address-family
   |  |           +--rw config
   |  |              +--rw ldp-disable?    boolean
   |  +--rw label-policy
   |     +--rw independent-mode
   |     |  +--rw assign {policy-label-assignment-config}?
   |     |  |  +--rw (prefix-option)?
   |     |  |        +--rw prefix-list?        prefix-list-ref
   |     |  |        +--rw host-routes-only?   boolean
   |     |  +--rw advertise
   |     |  |  +--rw explicit-null
   |     |  |  |  +--rw enable?        boolean
   |     |  |  |  +--rw prefix-list?   prefix-list-ref
   |     |  |  +--rw prefix-list?     prefix-list-ref
   |     |  +--rw accept
   |     |     +--rw prefix-list?   prefix-list-ref
   |     +--rw ordered-mode {policy-ordered-label-config}?
   |        +--rw egress-lsr
   |        |  +--rw prefix-list?   prefix-list-ref
   |        +--rw advertise
   |        |  +--rw prefix-list?   prefix-list-ref
   |        +--rw accept
   |           +--rw prefix-list?   prefix-list-ref
```

```
      +--rw peers
         +--rw config
         |  +--rw session-authentication-md5-password?   string
         |  +--rw session-ka-holdtime?                   uint16
         |  +--rw session-ka-interval?                   uint16
         |  +--rw session-downstream-on-demand {session-downstream-on-demand-con
fig}?
         |     +--rw enable?      boolean
         |     +--rw peer-list?   peer-list-ref
         +--rw peer* [lsr-id]
            +--rw lsr-id    yang:dotted-quad
            +--rw config
               +--rw admin-down?                            boolean
               +--rw capability
               +--rw label-policy
               |  +--rw advertise
               |  |  +--rw prefix-list?   prefix-list-ref
               |  +--rw accept
               |     +--rw prefix-list?   prefix-list-ref
               +--rw session-authentication-md5-password?   string
               +--rw graceful-restart
               |  +--rw enable?           boolean
               |  +--rw reconnect-time?   uint16
               |  +--rw recovery-time?    uint16
               +--rw session-ka-holdtime?                   uint16
               +--rw session-ka-interval?                   uint16
               +--rw address-family
                  +--rw ipv4
                  |  +--rw label-policy
                  |     +--rw advertise
                  |     |  +--rw prefix-list?   prefix-list-ref
                  |     +--rw accept
                  |        +--rw prefix-list?   prefix-list-ref
                  +--rw ipv6
                     +--rw label-policy
                        +--rw advertise
                        |  +--rw prefix-list?   prefix-list-ref
                        +--rw accept
                           +--rw prefix-list?   prefix-list-ref
```

Figure 3

3.2.1. Configuration Hierarchy

The LDP configuration container is logically divided into following high-level config areas:

```
      Per-VRF parameters
          o Global parameters
          o Per-address-family parameters
          o LDP Capabilities parameters
          o Hello Discovery parameters
              - interfaces
                 - Per-interface:
                    Global
                    Per-address-family
              - targeted
                 - Per-target
         o Peer parameters
              - Global
              - Per-peer
                 Per-address-family
                 Capabilities parameters
         o Forwarding parameters
```

Figure 4

Following subsections briefly explain these configuration areas.

3.2.1.1. Per-VRF parameters

LDP module resides under an network-instance and the scope of any LDP configuration defined under this tree is per network-instance (per-VRF). This configuration is further divided into sub categories as follows.

3.2.1.1.1. Per-VRF global parameters

There are configuration items that are available directly under a VRF instance and do not fall under any other sub tree. Example of such a parameter is LDP LSR id that is typically configured per VRF. To keep legacy LDP features and applications working in an LDP IPv4 networks with this model, this document recommends an operator to pick a routable IPv4 unicast address as an LSR Id.

3.2.1.1.2. Per-VRF Capabilities parameters

This container falls under global tree and holds the LDP capabilities that are to be enabled for certain features. By default, an LDP capability is disabled unless explicitly enabled. These capabilities are typically used to negotiate with LDP peer(s) the support/non-support related to a feature and its parameters. The scope of a capability enabled under this container applies to all LDP peers in the given VRF instance. There is also a peer level capability container that is provided to override a capability that is enabled/ specified at VRF level.

3.2.1.1.3. Per-VRF Per-Address-Family parameters

Any LDP configuration parameter related to IP address family (AF) whose scope is VRF wide is configured under this tree. The examples of per-AF parameters include enabling LDP for an address family, prefix-list based label policies, and LDP transport address.

3.2.1.1.4. Per-VRF Hello Discovery parameters

This container is used to hold LDP configuration related to Hello and discovery process for both basic (link) and extended (targeted) discovery.

The "interfaces" is a container to configure parameters related to VRF interfaces. There are parameters that apply to all interfaces (such as hello timers), as well as parameters that can be configured per-interface. Hence, an interface list is defined under "interfaces" container. The model defines parameters to configure per-interface non AF related items, as well as per-interface per-AF items. The example of former is interface hello timers, and example of latter is enabling hellos for a given AF under an interface.

The "targeted" container under a VRF instance allows to configure LDP targeted discovery related parameters. Within this container, the "target" list provides a mean to configure multiple target addresses to perform extended discovery to a specific destination target, as well as to fine-tune the per-target parameters.

3.2.1.1.5. Per-VRF Peer parameters

This container is used to hold LDP configuration related to LDP sessions and peers under a VRF instance. This container allows to configure parameters that either apply on VRF’s all peers or a subset (peer-list) of VRF peers. The example of such parameters include authentication password, session KA timers etc. Moreover, the model also allows per-peer parameter tuning by specifying a "peer" list under the "peers" container. A peer is uniquely identified using its LSR Id and hence LSR Id is the key for peer list

Like per-interface parameters, some per-peer parameters are AF-agnostic (i.e. either non AF related or apply to both IP address families), and some that belong to an AF. The example of former is per-peer session password configuration, whereas the example of latter is prefix-list based label policies (inbound and outbound) that apply to a given peer.

3.2.1.1.6. Per-VRF Forwarding parameters

This container is used to hold configuration used to control LDP forwarding behavior under a VRF instance. One example of a configuration under this container is when a user wishes to enable neighbor discovery on an interface but wishes to disable use of the same interface as forwarding nexthop. This example configuration makes sense only when there are more than one LDP enabled interfaces towards the neighbor.

3.2.2. All-VRFs Configuration

[Ed note: TODO]

3.3. Operational State

Operational state of LDP can be queried and obtained from read-only state containers that fall under the same tree (/rt:routing/ rt:control-plane-protocols/) as the configuration.

Please note this state tree refers both the configuration "applied" state as well as the "derived" state related to the protocol. [Ed note: This is where this model differs presently from [I-D.openconfig-netmod-opstate] and subject to alignment in later revisions]

Following is a simplified graphical representation of the data model for LDP operational state.

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
      +--rw global
      |  +--ro state
      |  |  +--ro capability
      |  |  |  +--ro end-of-lib {capability-end-of-lib}?
      |  |  |  |  +--ro enable?   boolean
      |  |  |  +--ro typed-wildcard-fec {capability-typed-wildcard-fec}?
      |  |  |  |  +--ro enable?   boolean
      |  |  |  +--ro upstream-label-assignment {capability-upstream-label-assign
ment}?
      |  |  |  |  +--ro enable?   boolean
      |  |  +--ro graceful-restart
      |  |  |  +--ro enable?                boolean
      |  |  |  +--ro helper-enable?         boolean {graceful-restart-helper-mod
e}?
      |  |  |  +--ro reconnect-time?        uint16
      |  |  |  +--ro recovery-time?         uint16
      |  |  |  +--ro forwarding-holdtime?   uint16
```

```
     |  |  +--ro igp-synchronization-delay?   uint16
     |  |  +--ro lsr-id?                      yang:dotted-quad
     |  +--rw address-family* [afi]
     |  |  +--rw afi       ldp-address-family
     |  |  +--ro state
     |  |     +--ro enable?         boolean
     |  |     +--ro label-policy
     |  |     |  +--ro independent-mode
     |  |     |  |  +--ro assign {policy-label-assignment-config}?
     |  |     |  |  |  +--ro (prefix-option)?
     |  |     |  |  |     +--:(prefix-list)
     |  |     |  |  |     |  +--ro prefix-list?        prefix-list-ref
     |  |     |  |  |     +--:(host-routes-only)
     |  |     |  |  |        +--ro host-routes-only?   boolean
     |  |     |  |  +--ro advertise
     |  |     |  |  |  +--ro explicit-null
     |  |     |  |  |  |  +--ro enable?        boolean
     |  |     |  |  |  |  +--ro prefix-list?   prefix-list-ref
     |  |     |  |  |  +--ro prefix-list?     prefix-list-ref
     |  |     |  |  +--ro accept
     |  |     |  |     +--ro prefix-list?   prefix-list-ref
     |  |     |  +--ro ordered-mode {policy-ordered-label-config}?
     |  |     |     +--ro egress-lsr
     |  |     |     |  +--ro prefix-list?   prefix-list-ref
     |  |     |     +--ro advertise
     |  |     |     |  +--ro prefix-list?   prefix-list-ref
     |  |     |     +--ro accept
     |  |     |        +--ro prefix-list?   prefix-list-ref
     |  |     +--ro ipv4
     |  |     |  +--ro transport-address?   inet:ipv4-address
     |  |     |  +--ro bindings
     |  |     |     +--ro address* [address]
     |  |     |     |  +--ro address      inet:ipv4-address
     |  |     |     |  +--ro advertisement-type?   advertised-received
     |  |     |     |  +--ro peer?        leafref
     |  |     |     +--ro fec-label* [fec]
     |  |     |        +--ro fec     inet:ipv4-prefix
     |  |     |        +--ro peer* [peer advertisement-type]
     |  |     |           +--ro peer                  leafref
     |  |     |           +--ro advertisement-type    advertised-received
     |  |     |           +--ro label?                mpls:mpls-label
     |  |     |           +--ro used-in-forwarding?   boolean
     |  |     +--ro ipv6
     |  |        +--ro transport-address?   inet:ipv6-address
     |  |        +--ro binding
     |  |           +--ro address* [address]
     |  |           |  +--ro address      inet:ipv6-address
     |  |           |  +--ro advertisement-type?   advertised-received
```

```
      |  |           |  +--ro peer?        leafref
      |  |           +--ro fec-label* [fec]
      |  |              +--ro fec     inet:ipv6-prefix
      |  |              +--ro peer* [peer advertisement-type]
      |  |                 +--ro peer                    leafref
      |  |                 +--ro advertisement-type    advertised-received
      |  |                 +--ro label?                mpls:mpls-label
      |  |                 +--ro used-in-forwarding?   boolean
      |  +--rw discovery
      |  |  +--rw interfaces
      |  |  |  +--ro state
      |  |  |  |  +--ro hello-holdtime?   uint16
      |  |  |  |  +--ro hello-interval?   uint16
      |  |  |  +--rw interface* [interface]
      |  |  |     +--ro state
      |  |  |     |  +--ro hello-holdtime?              uint16
      |  |  |     |  +--ro hello-interval?              uint16
      |  |  |     |  +--ro igp-synchronization-delay?   uint16 {per-interface-ti
mer-config}?
      |  |  |     |  +--ro next-hello?                  uint16
      |  |  |     +--rw address-family* [afi]
      |  |  |        +--rw afi       ldp-address-family
      |  |  |        +--ro state
      |  |  |           +--ro enable?   boolean
      |  |  |           +--ro ipv4
      |  |  |           |  +--ro transport-address?   union
      |  |  |           |  +--ro hello-adjacencies* [adjacent-address]
      |  |  |           |     +--ro adjacent-address    inet:ipv4-address
      |  |  |           |     +--ro flag*               identityref
      |  |  |           |     +--ro hello-holdtime
      |  |  |           |     |  +--ro adjacent?     uint16
      |  |  |           |     |  +--ro negotiated?   uint16
      |  |  |           |     |  +--ro remaining?    uint16
      |  |  |           |     +--ro next-hello?         uint16
      |  |  |           |     +--ro statistics
      |  |  |           |     |  +--ro discontinuity-time    yang:date-and-time
      |  |  |           |     |  +--ro hello-received?       yang:counter64
      |  |  |           |     |  +--ro hello-dropped?        yang:counter64
      |  |  |           |     +--ro peer?               leafref
      |  |  |           +--ro ipv6
      |  |  |              +--ro transport-address?   union
      |  |  |              +--ro hello-adjacencies* [adjacent-address]
      |  |  |                 +--ro adjacent-address    inet:ipv6-address
      |  |  |                 +--ro flag*               identityref
      |  |  |                 +--ro hello-holdtime
      |  |  |                 |  +--ro adjacent?     uint16
      |  |  |                 |  +--ro negotiated?   uint16
      |  |  |                 |  +--ro remaining?    uint16
      |  |  |                 +--ro next-hello?         uint16
      |  |  |                 +--ro statistics
      |  |  |                 |  +--ro discontinuity-time    yang:date-and-time
      |  |  |                 |  +--ro hello-received?       yang:counter64
      |  |  |                 |  +--ro hello-dropped?        yang:counter64
      |  |  |                 +--ro peer?              leafref
      |  |  +--rw targeted
      |  |     +--ro state
      |  |     |  +--ro hello-holdtime?   uint16
      |  |     |  +--ro hello-interval?   uint16
      |  |     |  +--ro hello-accept {policy-extended-discovery-config}?
      |  |     |     +--ro enable?          boolean
      |  |     |     +--ro neighbor-list?   neighbor-list-ref
      |  |     +--rw address-family* [afi]
      |  |        +--rw afi      ldp-address-family
      |  |        +--ro state
      |  |        |  +--ro ipv4
      |  |        |  |  +--ro hello-adjacencies* [local-address adjacent-address
]
      |  |        |  |     +--ro local-address       inet:ipv4-address
      |  |        |  |     +--ro adjacent-address    inet:ipv4-address
      |  |        |  |     +--ro flag*               identityref
      |  |        |  |     +--ro hello-holdtime
      |  |        |  |     |  +--ro adjacent?     uint16
      |  |        |  |     |  +--ro negotiated?   uint16
      |  |        |  |     |  +--ro remaining?    uint16
      |  |        |  |     +--ro next-hello?         uint16
      |  |        |  |     +--ro statistics
      |  |        |  |     |  +--ro discontinuity-time    yang:date-and-time
      |  |        |  |     |  +--ro hello-received?       yang:counter64
      |  |        |  |     |  +--ro hello-dropped?        yang:counter64
      |  |        |  |     +--ro peer?              leafref
      |  |        |  +--ro ipv6
      |  |        |     +--ro hello-adjacencies* [local-address adjacent-address
]
      |  |        |        +--ro local-address       inet:ipv6-address
      |  |        |        +--ro adjacent-address    inet:ipv6-address
      |  |        |        +--ro flag*               identityref
      |  |        |        +--ro hello-holdtime
      |  |        |        |  +--ro adjacent?     uint16
      |  |        |        |  +--ro negotiated?   uint16
      |  |        |        |  +--ro remaining?    uint16
      |  |        |        +--ro next-hello?         uint16
      |  |        |        +--ro statistics
      |  |        |        |  +--ro discontinuity-time    yang:date-and-time
      |  |        |        |  +--ro hello-received?       yang:counter64
      |  |        |        |  +--ro hello-dropped?        yang:counter64
      |  |        |        +--ro peer?              leafref
      |  |        +--rw ipv4
      |  |        |  +--rw target* [adjacent-address]
      |  |        |     +--rw adjacent-address    inet:ipv4-address
      |  |        |     +--ro state
      |  |        |        +--ro enable?          boolean
      |  |        |        +--ro local-address?   inet:ipv4-address
      |  |        +--rw ipv6
      |  |           +--rw target* [adjacent-address]
      |  |              +--rw adjacent-address    inet:ipv6-address
      |  |              +--ro state
      |  |                 +--ro enable?          boolean
      |  |                 +--ro local-address?   inet:ipv6-address
      |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
      |  |  +--rw interfaces
      |  |     +--rw interface* [interface]
      |  |        +--rw interface         mpls-interface-ref
      |  |        +--rw address-family* [afi]
      |  |           +--rw afi       ldp-address-family
      |  |           +--ro state
      |  |              +--ro ldp-disable?    boolean
      +--rw peers
         +--ro state
         |  +--ro session-authentication-md5-password?   string
         |  +--ro session-ka-holdtime?                   uint16
         |  +--ro session-ka-interval?                   uint16
         |  +--ro session-downstream-on-demand {session-downstream-on-demand-con
fig}?
         |     +--ro enable?      boolean
         |     +--ro peer-list?   peer-list-ref
         +--rw peer* [lsr-id]
            +--rw lsr-id    yang:dotted-quad
            +--ro state
               +--ro admin-down?                            boolean
               +--ro capability
               +--ro label-policy
               |  +--ro advertise
               |  |  +--ro prefix-list?   prefix-list-ref
               |  +--ro accept
               |     +--ro prefix-list?   prefix-list-ref
               +--ro session-authentication-md5-password?   string
               +--ro graceful-restart
               |  +--ro enable?           boolean
               |  +--ro reconnect-time?   uint16
               |  +--ro recovery-time?    uint16
               +--ro session-ka-holdtime?                   uint16
               +--ro session-ka-interval?                   uint16
               +--ro address-family
               |  +--ro ipv4
               |  |  +--ro label-policy
               |  |  |  +--ro advertise
               |  |  |  |  +--ro prefix-list?   prefix-list-ref
               |  |  |  +--ro accept
```

```
     |  |  |     +--ro prefix-list?   prefix-list-ref
     |  |  +--ro hello-adjacencies* [local-address adjacent-address]
     |  |     +--ro local-address       inet:ipv4-address
     |  |     +--ro adjacent-address    inet:ipv4-address
     |  |     +--ro flag*               identityref
     |  |     +--ro hello-holdtime
     |  |     |  +--ro adjacent?     uint16
     |  |     |  +--ro negotiated?   uint16
     |  |     |  +--ro remaining?    uint16
     |  |     +--ro next-hello?         uint16
     |  |     +--ro statistics
     |  |     |  +--ro discontinuity-time    yang:date-and-time
     |  |     |  +--ro hello-received?       yang:counter64
     |  |     |  +--ro hello-dropped?        yang:counter64
     |  |     +--ro interface?          mpls-interface-ref
     |  +--ro ipv6
     |     +--ro label-policy
     |     |  +--ro advertise
     |     |  |  +--ro prefix-list?   prefix-list-ref
     |     |  +--ro accept
     |     |     +--ro prefix-list?   prefix-list-ref
     |     +--ro hello-adjacencies* [local-address adjacent-address]
     |        +--ro local-address       inet:ipv6-address
     |        +--ro adjacent-address    inet:ipv6-address
     |        +--ro flag*               identityref
     |        +--ro hello-holdtime
     |        |  +--ro adjacent?     uint16
     |        |  +--ro negotiated?   uint16
     |        |  +--ro remaining?    uint16
     |        +--ro next-hello?         uint16
     |        +--ro statistics
     |        |  +--ro discontinuity-time    yang:date-and-time
     |        |  +--ro hello-received?       yang:counter64
     |        |  +--ro hello-dropped?        yang:counter64
     |        +--ro interface?          mpls-interface-ref
     +--ro label-advertisement-mode
     |  +--ro local?        label-adv-mode
     |  +--ro peer?         label-adv-mode
     |  +--ro negotiated?   label-adv-mode
     +--ro next-keep-alive?                      uint16
     +--ro peer-ldp-id?                          yang:dotted-quad
     +--ro received-peer-state
     |  +--ro graceful-restart
     |  |  +--ro enable?           boolean
     |  |  +--ro reconnect-time?   uint16
     |  |  +--ro recovery-time?    uint16
     |  +--ro capability
     |     +--ro end-of-lib
```

```
               |     |  +--ro enable?   boolean
               |     +--ro typed-wildcard-fec
               |     |  +--ro enable?   boolean
               |     +--ro upstream-label-assignment
               |        +--ro enable?   boolean
               +--ro session-holdtime
               |  +--ro peer?         uint16
               |  +--ro negotiated?   uint16
               |  +--ro remaining?    uint16
               +--ro session-state?                        enumeration
               +--ro tcp-connection
               |  +--ro local-address?    inet:ip-address
               |  +--ro local-port?       inet:port-number
               |  +--ro remote-address?   inet:ip-address
               |  +--ro remote-port?      inet:port-number
               +--ro up-time?                              string
               +--ro statistics
                  +--ro discontinuity-time          yang:date-and-time
                  +--ro received
                  |  +--ro total-octets?          yang:counter64
                  |  +--ro total-messages?        yang:counter64
                  |  +--ro address?               yang:counter64
                  |  +--ro address-withdraw?      yang:counter64
                  |  +--ro initialization?        yang:counter64
                  |  +--ro keepalive?             yang:counter64
                  |  +--ro label-abort-request?   yang:counter64
                  |  +--ro label-mapping?         yang:counter64
                  |  +--ro label-release?         yang:counter64
                  |  +--ro label-request?         yang:counter64
                  |  +--ro label-withdraw?        yang:counter64
                  |  +--ro notification?          yang:counter64
                  +--ro sent
                  |  +--ro total-octets?          yang:counter64
                  |  +--ro total-messages?        yang:counter64
                  |  +--ro address?               yang:counter64
                  |  +--ro address-withdraw?      yang:counter64
                  |  +--ro initialization?        yang:counter64
                  |  +--ro keepalive?             yang:counter64
                  |  +--ro label-abort-request?   yang:counter64
                  |  +--ro label-mapping?         yang:counter64
                  |  +--ro label-release?         yang:counter64
                  |  +--ro label-request?         yang:counter64
                  |  +--ro label-withdraw?        yang:counter64
                  |  +--ro notification?          yang:counter64
                  +--ro total-addresses?            uint32
                  +--ro total-labels?               uint32
                  +--ro total-fec-label-bindings?   uint32
```

Figure 5

3.3.1. Derived States

Following are main areas for which LDP operational "derived" state is defined:

Neighbor Adjacencies

Peer

Bindings (FEC-label and address)

Capabilities

3.3.1.1. Adjacency state

Neighbor adjacencies are per address-family hello adjacencies that are formed with neighbors as result of LDP basic or extended discovery. In terms of organization, there is a source of discovery (e.g. interface or target address) along with its associated parameters and one or more discovered neighbors along with neighbor discovery related parameters. For the basic discovery, there could be more than one discovered neighbor for a given source (interface), whereas there is at most one discovered neighbor for an extended discovery source (local-address and target-address). This is also to be noted that the reason for a targeted neighbor adjacency could be either an active source (locally configured targeted) or passive source (to allow any incoming extended/targeted hellos). A neighbor/adjacency record also contains session-state that helps highlight whether a given adjacency has progressed to subsequent session level or to eventual peer level.

Following captures high level tree hierarchy for neighbor adjacency state.

```
   +--rw mpls-ldp!
      +--rw discovery
         +--rw interfaces
         |  +--rw interface* [interface]
         |     +--rw address-family* [af]
         |        +--ro state
         |           +--ro ipv4 (or ipv6)
         |              +--ro hello-adjacencies* [adjacent-address]
         |                 +--ro adjacent-address
         |                    . . . .
         |                    . . . .
         +--rw targeted
            +--rw address-family* [afi]
                +--rw afi      address-family
                   +--ro state
                      +--ro ipv4 (or ipv6)
                         +--ro hello-adjacencies* [local-address adjacent-addres
s]
                            +--ro local-address
                            +--ro adjacent-address
                               . . . .
                               . . . .
```

Figure 6

3.3.1.2. Peer state

Peer related derived state is presented under peers tree. This is one of the core state that provides info on the session related parameters (mode, authentication, KA timeout etc.), TCP connection info, hello adjacencies for the peer, statistics related to messages and bindings, and capabilities exchange info.

Following captures high level tree hierarchy for peer state.

```
  +--rw mpls-ldp!
      +--rw peers
         +--rw peer* [lsr-id]
            +--rw lsr-id
            +--ro state
               +--ro session-ka-holdtime?
               +-- . . . .
               +-- . . . .
               +--ro capability
               +  +ro -- . . .
               +--ro address-family
               |  +--ro ipv4 (or ipv6)
               |     +--ro hello-adjacencies* [local-address adjacent-address]
               |        . . . .
               |        . . . .
               +--ro received-peer-state
               |  +--ro . . . .
               |  +--ro capability
               |     +--ro . . . .
               +--ro statistics
                  +-- . . . .
                  +-- . . . .
```

Figure 7

3.3.1.3. Bindings state

Binding state provides information on LDP FEC-label bindings as well as address binding for both inbound (received) as well as outbound (advertised) direction. FEC-label bindings are presented as a FEC-centric view, and address bindings are presented as an address-centric view:

```
        FEC-Label bindings:
            FEC 200.1.1.1/32:
              advertised: local-label 16000
                peer 192.168.0.2:0
                peer 192.168.0.3:0
                peer 192.168.0.4:0
              received:
                peer 192.168.0.2:0, label 16002, used-in-forwarding=Yes
                peer 192.168.0.3:0, label 17002, used-in-forwarding=No
            FEC 200.1.1.2/32:
               . . . .
            FEC 201.1.0.0/16:
               . . . .


        Address bindings:
            Addr 1.1.1.1:
              advertised
            Addr 1.1.1.2:
              advertised
            Addr 2.2.2.2:
              received, peer 192.168.0.2
            Addr 2.2.2.22:
              received, peer 192.168.0.2
            Addr 3.3.3.3:
              received, peer 192.168.0.3
            Addr 3.3.3.33:
              received, peer 192.168.0.3
```

Figure 8

Note that all local addresses are advertised to all peers and hence no need to provide per-peer information for local address advertisement. Furthermore, note that it is easy to derive a peer-centric view for the bindings from the information already provided in this model.

Following captures high level tree hierarchy for bindings state.

```
   +--rw mpls-ldp!
      +--rw global
         +--rw address-family* [afi]
            +--rw afi       address-family
            +--ro state
               +--ro ipv4 (or ipv6)
                  +--ro bindings
                     +--ro address* [address]
                     |  +--ro address
                     |  +--ro direction?   advertised-received
                     |  +--ro peer?        leafref
                     +--ro fec-label* [fec]
                        +--ro fec     inet:ipv4-prefix
                        +--ro peer* [peer advertisement-type]
                           +--ro peer                  leafref
                           +--ro advertisement-type    advertised-received
                           +--ro label?                mpls:mpls-label
                           +--ro used-in-forwarding?   boolean
```

Figure 9

3.3.1.4. Capabilities state

LDP capabilities state comprise two types of information - global information (such as timer etc.), and per-peer information.

Following captures high level tree hierarchy for LDP capabilities state.

```
      +--rw mpls-ldp!
         +--rw global
         |  +--ro state
         |     +--ro capability
         |        +--ro . . . .
         |        +--ro . . . .
         +--rw peers
            +--rw peer* [lsr-id]
               +--rw lsr-id    yang:dotted-quad
               +--ro state
                  +--ro received-peer-state
                     +--ro capability
                        +--ro . . . .
                        +--ro . . . .
```

Figure 10

3.4. Notifications

This model defines a list of notifications to inform client of important events detected during the protocol operation. These events include events related to changes in the operational state of an LDP peer, hello adjacency, and FEC etc. It is to be noted that an LDP FEC is treated as operational (up) as long as it has at least 1 NHLFE with outgoing label.

Following is a simplified graphical representation of the data model for LDP notifications.

```
module: ietf-mpls-ldp
notifications:
   +---n mpls-ldp-peer-event
   |  +--ro event-type?   oper-status-event-type
   |  +--ro peer-ref?     leafref
   +---n mpls-ldp-hello-adjacency-event
   |  +--ro event-type?   oper-status-event-type
   |  +--ro (hello-adjacency-type)?
   |     +--:(targeted)
   |     |  +--ro targeted
   |     |     +--ro target-address?   inet:ip-address
   |     +--:(link)
   |        +--ro link
   |           +--ro next-hop-interface?   mpls-interface-ref
   |           +--ro next-hop-address?     inet:ip-address
   +---n mpls-ldp-fec-event
      +--ro event-type?   oper-status-event-type
      +--ro prefix?       inet:ip-prefix
```

Figure 11

3.5. Actions

This model defines a list of rpcs that allow performing an action or executing a command on the protocol. For example, it allows to clear (reset) LDP peers, hello-adjacencies, and statistics. The model makes an effort to provide different level of control so that a user is able to either clear all, or clear all for a given type, or clear a specific entity.

Following is a simplified graphical representation of the data model for LDP actions.

```
   module: ietf-mpls-ldp
   rpcs:
      +---x mpls-ldp-clear-peer
      |  +---w input
      |     +---w lsr-id?   union
      +---x mpls-ldp-clear-hello-adjacency
      |  +---w input
      |     +---w hello-adjacency
      |        +---w (hello-adjacency-type)?
      |           +--:(targeted)
      |           |  +---w targeted!
      |           |     +---w target-address?   inet:ip-address
      |           +--:(link)
      |              +---w link!
      |                 +---w next-hop-interface?   mpls-interface-ref
      |                 +---w next-hop-address?     inet:ip-address
      +---x mpls-ldp-clear-peer-statistics
         +---w input
            +---w lsr-id?   union
```

Figure 12

4. mLDP YANG Model

4.1. Overview

Due to tight dependency of mLDP on LDP, mLDP model builds on top of LDP model defined earlier in the document. Following are the main mLDP areas and documents that are within the scope of this model:

- o mLDP Base Specification [RFC6388]
- o mLDP Recursive FEC [RFC6512]
- o Targeted mLDP [RFC7060]
- o mLDP Fast-Reroute (FRR)
  - * Node Protection [RFC7715]
  - * Multicast-only
- o Hub-and-Spoke Multipoint LSPs [RFC7140]
- o mLDP In-band Signaling [RFC6826] (future revision)
- o mLDP In-band signaling in a VRF [RFC7246]

- mLDP In-band Signaling with Wildcards [RFC7438] (future revision)

- Configured Leaf LSPs (manually provisioned)

[Ed Note: Some of the topics in the above list are to be addressed/ added in later revision of this document].

4.2. Configuration

4.2.1. Configuration Hierarchy

In terms of overall configuration layout, following figure highlights extensions to LDP configuration model to incorporate mLDP:

```
      +-- mpls-ldp
         +-- ...
         +-- ...
         +-- mldp
         |  +-- ...
         |  +-- ...
         |  +-- address-family* [af]
         |     +-- af
         |        +-- ...
         |        +-- ...
         +-- global
         |  +-- ...
         |  +-- capability
         |     +-- ...
         |     +-- ...
         |     +-- mldp
         |        +-- ...
         |        +-- ...
         +-- discovery
         |  +-- ...
         |  +-- ...
         +-- forwarding-nexthop
         |  +-- interfaces
         |     +-- interface* [interface]
         |        +-- interface
         |        +-- address-family* [af]
         |           +-- af
         |           +-- ...
         |           +-- mldp-disable
         +-- peers
            +-- ...
            +-- ...
            +-- peer* [lsr-id]
                +-- ...
                +-- ...
                +-- capability
                    +-- ...
                    +-- ...
                    +-- mldp
                        +-- ...
                        +-- ...
```

Figure 13

From above hierarchy, we can categorize mLDP configuration parameters into two types:

- o Parameters that leverage/extend LDP containers and parameters

- o Parameters that are mLDP specific

Following subsections first describe mLDP specific configuration parameters, followed by those leveraging LDP.

4.2.2. mldp container

mldp container resides directly under "mpls-ldp" and holds the configuration related to items that are mLDP specific. The main items under this container are:

- o mLDP enabling: To enable mLDP under a (VRF) routing instance, mldp container is enabled under LDP. Given that mLDP requires LDP signalling, it is not sensible to allow disabling LDP control plane under a (VRF) network-instance while requiring mLDP to be enabled for the same. However, if a user wishes only to allow signalling for multipoint FECs on an LDP/mLDP enabled VRF instance, he/she can use LDP label-policies to disable unicast FECs under the VRF.

- o mLDP per-AF features: mLDP manages its own list of IP address-families and the features enabled underneath. The per-AF mLDP configuration items include:

  - * Multicast-only FRR: This enables Multicast-only FRR functionality for a given AF under mLDP. The feature allows route-policy to be configured for finer control/applicability of the feature.

  - * Recursive FEC: The recursive-fec feature [RFC6512] can be enabled per AF with a route-policy.

  - * Configured Leaf LSPs: To provision multipoint leaf LSP manually, a container is provided per-AF under LDP. The configuration is flexible and allows a user to specify MP LSPs of type p2mp or mp2mp with IPv4 or IPv6 root address(es) by using either LSP-Id or (S,G).

Targeted mLDP feature specification [RFC7060] do not require any mLDP specific configuration. It, however, requires LDP upstream-label-assignment capability [RFC6389] to be enabled.

4.2.3. Leveraging LDP containers

mLDP configuration model leverages following configuration areas and containers that are already defined for LDP:

- o Capabilities: A new container "mldp" is defined under Capabilities container. This new container specifies any mLDP specific capabilities and their parameters. Moreover, a new "mldp" container is also added under per-peer capability container to override/control mLDP specific capabilities on a peer level. In the scope of this document, the most important capabilities related to mLDP are p2mp, mp2mp, make-before-break, hub-and-spoke, and node-protection.

- o Discovery and Peer: mLDP requires LDP discovery and peer procedures to form mLDP peering. A peer is treated as mLDP peer only when either P2MP or MP2MP capabilities have been successfully exchanged with the peer. If a user wish to selectively enable or disable mLDP with a LDP-enabled peer, he/she may use per-peer mLDP capabilities configuration. [Ed Note: The option to control mLDP enabling/disabling on a peer-list is being explored for future]. In most common deployments, it is desirable to disable mLDP (capabilities announcements) on a targeted-only LDP peering, where targeted-only peer is the one whose discovery sources are targeted only. In future revision, a configuration option for this support will also be provided.

- o Forwarding: By default, mLDP is allowed to select any of the LDP enabled interface as a downstream interface towards a nexthop (LDP/mLDP peer) for MP LSP programming. However, a configuration option is provided to allow mLDP to exclude a given interface from such a selection. Note that such a configuration option will be useful only when there are more than one interfaces available for the downstream selection.

This goes without saying that mLDP configuration tree follows the same approach as LDP, where the tree comprise leafs for intended configuration.

4.2.4. YANG tree

The following figure captures the YANG tree for mLDP configuration. To keep the focus, the figure has been simplified to display only mLDP items without any LDP items.

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
```

```
      +--rw global
      |  +--rw config
      |  |  +--rw capability
      |  |     +--rw mldp {mldp}?
      |  |        +--rw p2mp
      |  |        |  +--rw enable?   boolean
      |  |        +--rw mp2mp
      |  |        |  +--rw enable?   boolean
      |  |        +--rw make-before-break
      |  |        |  +--rw enable?             boolean
      |  |        |  +--rw switchover-delay?   uint16
      |  |        |  +--rw timeout?            uint16
      |  |        +--rw hub-and-spoke {capability-mldp-hsmp}?
      |  |        |  +--rw enable?   boolean
      |  |        +--rw node-protection {capability-mldp-node-protection}?
      |  |           +--rw plr?           boolean
      |  |           +--rw merge-point
      |  |              +--rw enable?                                boolean
      |  |              +--rw targeted-session-teardown-delay?   uint16
      |  +--rw mldp {mldp}?
      |  |  +--rw config
      |  |  |  +--rw enable?   boolean
      |  |  +--rw address-family* [afi]
      |  |     +--rw afi                      ldp-address-family
      |  |     +--rw config
      |  |     |  +--rw multicast-only-frr {mldp-mofrr}?
      |  |     |  |  +--rw prefix-list?   prefix-list-ref
      |  |     |  +--rw recursive-fec
      |  |     |     +--rw prefix-list?   prefix-list-ref
      |  |     +--rw configured-leaf-lsps
      |  |        +--rw p2mp
      |  |        |  +--rw roots-ipv4
      |  |        |  |  +--rw root* [root-address]
      |  |        |  |     +--rw root-address    inet:ipv4-address
      |  |        |  |     +--rw lsp* [lsp-id source-address group-address]
      |  |        |  |        +--rw lsp-id            uint16
      |  |        |  |        +--rw source-address    inet:ipv4-address
      |  |        |  |        +--rw group-address     inet:ipv4-address-no-zone
      |  |        |  +--rw roots-ipv6
      |  |        |     +--rw root* [root-address]
      |  |        |        +--rw root-address    inet:ipv6-address
      |  |        |        +--rw lsp* [lsp-id source-address group-address]
      |  |        |           +--rw lsp-id            uint16
      |  |        |           +--rw source-address    inet:ipv6-address
      |  |        |           +--rw group-address     inet:ipv6-address-no-zone
      |  |        +--rw mp2mp
      |  |           +--rw roots-ipv4
      |  |           |  +--rw root* [root-address]
      |  |           |     +--rw root-address    inet:ipv4-address
      |  |           |     +--rw lsp* [lsp-id source-address group-address]
      |  |           |        +--rw lsp-id            uint16
      |  |           |        +--rw source-address    inet:ipv4-address
      |  |           |        +--rw group-address     inet:ipv4-address-no-zone
      |  |           +--rw roots-ipv6
      |  |              +--rw root* [root-address]
      |  |                 +--rw root-address    inet:ipv6-address
      |  |                 +--rw lsp* [lsp-id source-address group-address]
      |  |                    +--rw lsp-id            uint16
      |  |                    +--rw source-address    inet:ipv6-address
      |  |                    +--rw group-address     inet:ipv6-address-no-zone
      |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
      |     +--rw interfaces
      |        +--rw interface* [interface]
      |           +--rw interface         if:interface-ref
      |           +--rw address-family* [afi]
      |              +--rw afi       address-family
      |              +--rw config
      |                 +--rw mldp-disable?   boolean {mldp}?
      +--rw peers
         +--rw peer* [lsr-id]
               +--rw capability
                  +--rw mldp {mldp}?
                     +--rw p2mp
                     |  +--rw enable?   boolean
                     +--rw mp2mp
                     |  +--rw enable?   boolean
                     +--rw make-before-break
                     |  +--rw enable?             boolean
                     |  +--rw switchover-delay?   uint16
                     |  +--rw timeout?            uint16
                     +--rw hub-and-spoke {capability-mldp-hsmp}?
                     |  +--rw enable?   boolean
                      +--rw node-protection {capability-mldp-node-protection}?
                        +--rw plr?           boolean
                        +--rw merge-point
                           +--rw enable?                            boolean
                           +--rw targeted-session-teardown-delay?   uint16
```

Figure 14

4.3. Operational State

Operational state of mLDP can be queried and obtained from this read-only container "mldp" which resides under mpls-ldp container.

Please note this state tree refers both the configuration "applied" state as well as the "derived" state related to the mLDP protocol.

Following is a simplified graphical representation of the data model for mLDP operational state:

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
   +--rw mpls-ldp!
      +--rw global
      |  +--ro state
      |  |  +--ro capability
      |  |     +--ro mldp {mldp}?
      |  |        +--ro p2mp
      |  |        |  +--ro enable?   boolean
      |  |        +--ro mp2mp
      |  |        |  +--ro enable?   boolean
      |  |        +--ro make-before-break
      |  |        |  +--ro enable?             boolean
      |  |        |  +--ro switchover-delay?   uint16
      |  |        |  +--ro timeout?            uint16
      |  |        +--ro hub-and-spoke {capability-mldp-hsmp}?
      |  |        |  +--ro enable?   boolean
      |  |        +--ro node-protection {capability-mldp-node-protection}?
      |  |           +--ro plr?           boolean
      |  |           +--ro merge-point
      |  |              +--ro enable?                            boolean
      |  |              +--ro targeted-session-teardown-delay?   uint16
      |  |
      |  +--rw mldp {mldp}?
      |  |  +--ro state
      |  |     +--ro enable?   boolean
      |  |  +--rw address-family* [afi]
      |  |     +--rw afi                      ldp-address-family
      |  |     +--ro state
      |  |        +--ro multicast-only-frr {mldp-mofrr}?
      |  |        |  +--ro prefix-list?   prefix-list-ref
      |  |        +--ro recursive-fec
      |  |        |  +--ro prefix-list?   prefix-list-ref
      |  |        +--ro ipv4
      |  |        |  +--ro roots
      |  |        |  |  +--ro root* [root-address]
      |  |        |  |     +--ro root-address    inet:ipv4-address
      |  |        |  |     +--ro is-self?        boolean
      |  |        |  |     +--ro reachability* [address interface]
      |  |        |  |        +--ro address      inet:ipv4-address
      |  |        |  |        +--ro interface    mpls-interface-ref
      |  |        |  |        +--ro peer?        leafref
      |  |        |  +--ro bindings
      |  |        |     +--ro opaque-type-lspid
      |  |        |     |  +--ro fec-label* [root-address lsp-id recur-root-addr
ess recur-rd]
      |  |        |     |     +--ro root-address          inet:ipv4-address
      |  |        |     |     +--ro lsp-id                uint32
      |  |        |     |     +--ro recur-root-address    inet:ip-address
      |  |        |     |     +--ro recur-rd              route-distinguisher
      |  |        |     |     +--ro multipoint-type?      multipoint-type
      |  |        |     |     +--ro peer* [direction peer advertisement-type]
      |  |        |     |        +--ro direction             downstream-upstream
      |  |        |     |        +--ro peer                  leafref
      |  |        |     |        +--ro advertisement-type    advertised-received
      |  |        |     |        +--ro label?                mpls:mpls-label
      |  |        |     |        +--ro mbb-role?             enumeration
      |  |        |     |        +--ro mofrr-role?           enumeration
      |  |        |     +--ro opaque-type-src
      |  |        |     |  +--ro fec-label* [root-address source-address group-a
ddress rd recur-root-address recur-rd]
      |  |        |     |     +--ro root-address          inet:ipv4-address
      |  |        |     |     +--ro source-address        inet:ip-address
      |  |        |     |     +--ro group-address         inet:ip-address-no-zon
e
      |  |        |     |     +--ro rd                    route-distinguisher
      |  |        |     |     +--ro recur-root-address    inet:ip-address
      |  |        |     |     +--ro recur-rd              route-distinguisher
      |  |        |     |     +--ro multipoint-type?      multipoint-type
      |  |        |     |     +--ro peer* [direction peer advertisement-type]
      |  |        |     |        +--ro direction             downstream-upstream
      |  |        |     |        +--ro peer                  leafref
      |  |        |     |        +--ro advertisement-type    advertised-received
      |  |        |     |        +--ro label?                mpls:mpls-label
      |  |        |     |        +--ro mbb-role?             enumeration
      |  |        |     |        +--ro mofrr-role?           enumeration
      |  |        |     +--ro opaque-type-bidir
      |  |        |        +--ro fec-label* [root-address rp group-address rd re
cur-root-address recur-rd]
      |  |        |           +--ro root-address          inet:ipv4-address
      |  |        |           +--ro rp                    inet:ip-address
      |  |        |           +--ro group-address         inet:ip-address-no-zon
e
      |  |        |           +--ro rd                    route-distinguisher
      |  |        |           +--ro recur-root-address    inet:ip-address
      |  |        |           +--ro recur-rd              route-distinguisher
      |  |        |           +--ro multipoint-type?      multipoint-type
      |  |        |           +--ro peer* [direction peer advertisement-type]
      |  |        |              +--ro direction             downstream-upstream
      |  |        |              +--ro peer                  leafref
      |  |        |              +--ro advertisement-type    advertised-received
      |  |        |              +--ro label?                mpls:mpls-label
      |  |        |              +--ro mbb-role?             enumeration
      |  |        |              +--ro mofrr-role?           enumeration
      |  |        +--ro ipv6
      |  |           +--ro roots
      |  |           |  +--ro root* [root-address]
      |  |           |     +--ro root-address    inet:ipv6-address
      |  |           |     +--ro is-self?        boolean
      |  |           |     +--ro reachability* [address interface]
      |  |           |        +--ro address      inet:ipv6-address
      |  |           |        +--ro interface    mpls-interface-ref
      |  |           |        +--ro peer?        leafref
      |  |           +--ro bindings
      |  |              +--ro opaque-type-lspid
      |  |              |  +--ro fec-label* [root-address lsp-id recur-root-addr
ess recur-rd]
      |  |              |     +--ro root-address          inet:ipv6-address
      |  |              |     +--ro lsp-id                uint32
      |  |              |     +--ro recur-root-address    inet:ip-address
      |  |              |     +--ro recur-rd              route-distinguisher
      |  |              |     +--ro multipoint-type?      multipoint-type
      |  |              |     +--ro peer* [direction peer advertisement-type]
      |  |              |        +--ro direction             downstream-upstream
      |  |              |        +--ro peer                  leafref
      |  |              |        +--ro advertisement-type    advertised-received
      |  |              |        +--ro label?                mpls:mpls-label
      |  |              |        +--ro mbb-role?             enumeration
      |  |              |        +--ro mofrr-role?           enumeration
      |  |              +--ro opaque-type-src
      |  |              |  +--ro fec-label* [root-address source-address group-a
ddress rd recur-root-address recur-rd]
      |  |              |     +--ro root-address          inet:ipv6-address
      |  |              |     +--ro source-address        inet:ip-address
      |  |              |     +--ro group-address         inet:ip-address-no-zon
e
      |  |              |     +--ro rd                    route-distinguisher
      |  |              |     +--ro recur-root-address    inet:ip-address
      |  |              |     +--ro recur-rd              route-distinguisher
      |  |              |     +--ro multipoint-type?      multipoint-type
      |  |              |     +--ro peer* [direction peer advertisement-type]
      |  |              |        +--ro direction             downstream-upstream
      |  |              |        +--ro peer                  leafref
      |  |              |        +--ro advertisement-type    advertised-received
      |  |              |        +--ro label?                mpls:mpls-label
      |  |              |        +--ro mbb-role?             enumeration
      |  |              |        +--ro mofrr-role?           enumeration
      |  |              +--ro opaque-type-bidir
      |  |                 +--ro fec-label* [root-address rp group-address rd re
cur-root-address recur-rd]
      |  |                    +--ro root-address          inet:ipv6-address
      |  |                    +--ro rp                    inet:ip-address
      |  |                    +--ro group-address         inet:ip-address-no-zon
e
      |  |                    +--ro rd                    route-distinguisher
      |  |                    +--ro recur-root-address    inet:ip-address
      |  |                    +--ro recur-rd              route-distinguisher
```

```
     |  |                    +--ro multipoint-type?      multipoint-type
     |  |                    +--ro peer* [direction peer advertisement-type]
     |  |                       +--ro direction             downstream-upstream
     |  |                       +--ro peer                  leafref
     |  |                       +--ro advertisement-type    advertised-received
     |  |                       +--ro label?                mpls:mpls-label
     |  |                       +--ro mbb-role?             enumeration
     |  |                       +--ro mofrr-role?           enumeration
     |  +--rw forwarding-nexthop {forwarding-nexthop-config}?
     |     +--rw interfaces
     |        +--rw interface* [interface]
     |           +--rw address-family* [afi]
     |              +--ro state
     |                 +--ro mldp-disable?   boolean {mldp}?
     +--rw peers
        +--rw peer* [lsr-id]
           +--ro state
              +--ro capability
              |   +--ro mldp {mldp}?
              |     +--ro p2mp
              |     |  +--ro enable?   boolean
              |     +--ro mp2mp
              |     |  +--ro enable?   boolean
              |     +--ro make-before-break
              |     |  +--ro enable?             boolean
              |     |  +--ro switchover-delay?   uint16
              |     |  +--ro timeout?            uint16
              |     +--ro hub-and-spoke {capability-mldp-hsmp}?
              |     |  +--ro enable?   boolean
              |     +--ro node-protection {capability-mldp-node-protection}?
              |        +--ro plr?           boolean
              |        +--ro merge-point
              |           +--ro enable?                             boolean
              |           +--ro targeted-session-teardown-delay?   uint16
              +--ro received-peer-state
                 +--ro capability
                    +--ro mldp {mldp}?
                       +--ro p2mp
                       |  +--ro enable?   boolean
                       +--ro mp2mp
                       |  +--ro enable?   boolean
                       +--ro make-before-break
                       |  +--ro enable?   boolean
                       +--ro hub-and-spoke
                       |  +--ro enable?   boolean
                       +--ro node-protection
                          +--ro plr?           boolean
                          +--ro merge-point?   boolean
```

Figure 15

4.3.1. Derived states

Following are main areas for which mLDP operational derived state is defined:

- o Root

- o Bindings (FEC-label)

- o Capabilities

4.3.1.1. Root state

Root address is a fundamental construct for MP FEC bindings and LSPs. The root state provides information on all the known roots in a given address-familty, and their information on the root reachability (as learnt from RIB). In case of multi-path reachability to a root, the selection of upstream path is done on per-LSP basis at the time of LSP setup. Similarly, when protection mechanisms like MBB or MoFRR are in place, the path designation as active/standby or primary/ backup is also done on per LSP basis. It is to be noted that a given root can be shared amongst multiple P2MP and/or MP2MP LSPs. Moreover, an LSP can be signaled to more than one root for RNR purposes.

The following diagram illustrates a root database on a branch/transit LSR:

```
        root 1.1.1.1:
            path1:
                RIB: GigEthernet 1/0, 12.1.0.2;
                LDP: peer 192.168.0.1:0
            path2:
                RIB: GigEthernet 2/0, 12.2.0.2;
                LDP: peer 192.168.0.3:0

        root 2.2.2.2:
            path1:
                RIB: 3.3.3.3;             (NOTE: This is a recursive path)
                LDP: peer 192.168.0.3:0   (NOTE: T-mLDP peer)

         root 9.9.9.9:
            . . . .
```

Figure 16

A root entry on a root LSR itself will be presented as follows:

```
          root 9.9.9.9:
              is-self
```

Figure 17

4.3.1.2. Bindings state

Binding state provides information on mLDP FEC-label bindings for both P2MP and MP2MP FEC types. Like LDP, the FEC-label binding derived state is presented in a FEC-centric view per address-family, and provides information on both inbound (received) and outbound (advertised) bindings. The FEC is presented as (root-address, opaque-type-data) and the direction (upstream or downstream) is picked with respect to root reachability. In case of MBB or/and MoFRR, the role of a given peer binding is also provided with respect to MBB (active or standby) or/and MoFRR (primary or backup).

This document covers following type of opaque values with their keys in the operational model of mLDP bindings:

Opaque Type	Key	RFC
Generic LSP Identifier	LSP Id	[RFC6388]
Transit IPv4 Source	Source, Group	[RFC6826]
Transit IPv6 Source	Source, Group	[RFC6826]
Transit IPv4 Bidir	RP, Group	[RFC6826]
Transit IPv6 Bidir	RP, Group	[RFC6826]
Transit VPNv4 Source	Source, Group, RD	[RFC7246]
Transit VPNv6 Source	Source, Group, RD	[RFC7246]
Transit VPNv4 Bidir	RP, Group, RD	[RFC7246]
Transit VPNv6 Bidir	RP, Group, RD	[RFC7246]
Recursive Opaque	Root	[RFC6512]
VPN-Recursive Opaque	Root, RD	[RFC6512]

Table 1: MP Opaque Types and keys

It is to be noted that there are three basic types (LSP Id, Source, and Bidir) and then there are variants (VPN, recursive, VPN-recursive) on top of these basic types.

Following captures high level tree hierarchy for mLDP bindings state:

```
+--rw mpls-ldp!
    +--rw mldp
      +--rw address-family* [afi]
         +--rw afi       address-family
         +--ro state
            +--ro ipv4 (or ipv6)
               +--ro bindings
                  +--ro opaque-type-xxx [root-address, type-specific-key]
                     +--ro root-address
                     +--ro ...
                     +--ro recur-root-address    inet:ipv4-address
                     +--ro recur-rd              route-distinguisher
                     +--ro multipoint-type?      multipoint-type
                     +--ro peer* [direction peer advertisement-type]
                        +--ro direction             downstream-upstream
                        +--ro peer                  leafref
                        +--ro advertisement-type    advertised-received
                        +--ro label?                mpls:mpls-label
                        +--ro mbb-role?             enumeration
                        +--ro mofrr-role?           enumeration
```

Figure 18

In the above tree, the type-specific-key varies with the base type as listed in earlier Table 1. For example, if the opaque type is Generic LSP Identifier, then the type-specific-key will be a uint32 value corresponding to the LSP. Please see the complete model for all other types.

Moreover, the binding tree defines only three types of sub-trees (i.e. lspid, src, and bidir) which is able to map the respective variants (vpn, recursive, and vpn-recusrive) accordingly. For example, the key for opaque-type-src is [R, S, G, rd, recur-R, recur-RD], where basic type will specify (R, S,G,-, -, -), VPN type will specify (R, S,G, rd, -, -), recursive type will specify [R, S,G, -, recur-R, -] and VPN-recursive type will specify [R, S,G, -, recur-R, recur-rd].

It is important to take note of the following:

o The address-family ipv4/ipv4 applies to "root" address in the mLDP binding tree. The other addresses (source, group, RP etc) do not have to be of the same address family type as the root.

o The "recur-root-address" field applies to Recursive opaque type, and (recur-root-address, recur-rd) fields applies to VPN-Recursive opaque types as defined in [RFC6512]

o In case of a recursive FEC, the address-family of the recur-root-address could be different than the address-family of the root address of original encapsulated MP FEC

The following diagram illustrates the FEC-label binding information structure for a P2MP (Transit IPv4 Source type) LSP on a branch/ transit LSR:

```
         FEC (root 2.2.2.2, S=192.168.1.1, G=224.1.1.1):
           type: p2mp
           upstream:
             advertised:
               peer 192.168.0.1:0, label 16000 (local)
           downstream:
             received:
               peer 192.168.0.2:0, label 17000 (remote)
               peer 192.168.0.3:0, label 18000  (remote)
```

Figure 19

The following diagram illustrates the FEC-label binding information structure for a similar MP2MP LSP on a branch/transit LSR:

```
      FEC (root 2.2.2.2, RP=192.168.9.9, G=224.1.1.1):
        type: mp2mp
        upstream:
          advertised:
            peer 192.168.0.1:0, label 16000 (local)
          received:
            peer 192.168.0.1:0, label 17000 (remote)
        downstream:
          advertised:
            peer 192.168.0.2:0, label 16001 (local), MBB role=active
            peer 192.168.0.3:0, label 16002 (local), MBB role=standby
          received:
            peer 192.168.0.2:0, label 17001 (remote)
            peer 192.168.0.3:0, label 18001 (remote)
```

Figure 20

4.3.1.3. Capabilities state

Like LDP, mLDP capabilities state comprise two types of information - global information and per-peer information.

4.4. Notifications

mLDP notification module consists of notification related to changes in the operational state of an mLDP FEC. Following is a simplified graphical representation of the data model for mLDP notifications:

```
notifications:
   +---n mpls-mldp-fec-event
      +--ro event-type?       oper-status-event-type
      +--ro tree-type?        multipoint-type
      +--ro root?             inet:ip-address
      +--ro (lsp-key-type)?
         +--:(lsp-id-based)
         |  +--ro lsp-id?           uint16
         +--:(source-group-based)
            +--ro source-address?   inet:ip-address
            +--ro group-address?    inet:ip-address
```

Figure 21

4.5. Actions

Currently, no RPCs/actions are defined for mLDP.

5. Open Items

Following is a list of open items that are to be discussed and addressed in future revisions of this document:

- Close on augmentation off "mpls" list in "ietf-mpls" defined in [I-D.saad-mpls-base-yang]
- Align operational state modeling with other routing procols and [I-D.openconfig-netmod-opstate]
- Complete the section on Protocol-centric implementations and all-vrfs
- Specify default values for configuration parameters
- Revisit and cut down on the scope of the document and number of features it is trying to cover
- Split the model into a base and extended items
- Add statistics for mLDP root LSPs and bindings
- Extend the "Configured Leaf LSPs" for various type of opaque-types
- Extend mLDP notifications for other types of opaque values as well
- Close on single vs separate document for mLDP Yang

6. YANG Specification

Following are actual YANG definition for LDP and mLDP constructs defined earlier in the document.

```
 <CODE BEGINS> file "ietf-mpls-ldp@2016-07-08.yang" -->

 module ietf-mpls-ldp {
   namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
   // replace with IANA namespace when assigned
   prefix ldp;

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

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

   import ietf-interfaces {
     prefix "if";
   }

   import ietf-ip {
     prefix "ip";
   }

   import ietf-routing {
     prefix "rt";
   }

   import ietf-mpls {
     prefix "mpls";
   }

   organization
     "IETF MPLS Working Group";
   contact
     "WG Web:   <http://tools.ietf.org/wg/teas/>
      WG List:  <mailto:teas@ietf.org>

      WG Chair: Loa Andersson
                <mailto:loa@pi.nu>

      WG Chair: Ross Callon
                <mailto:rcallon@juniper.net>

      WG Chair: George Swallow
                <mailto:swallow.ietf@gmail.com>

      Editor:   Kamran Raza
                <mailto:skraza@cisco.com>

      Editor:   Rajiv Asati
                <mailto:rajiva@cisco.com>

      Editor:   Xufeng Liu
                <mailto:xliu@kuatrotech.com>

      Editor:   Santosh Esale
```

```
                <mailto:sesale@juniper.net>

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

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

   description
     "This YANG module defines the essential components for the
      management of Multi-Protocol Label Switching (MPLS) Label
      Distribution Protocol (LDP) and Multipoint LDP (mLDP).";

   revision 2016-07-08 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: YANG Data Model for MPLS LDP and mLDP.";
   }

   /*
    * Features
    */

   feature admin-down-config {
     description
       "This feature indicates that the system allows to configure
        administrative down on a VRF instance and a peer.";
   }

   feature all-af-policy-config {
     description
       "This feature indicates that the system allows to configure
        policies that are applied to all address families.";
   }

   feature capability-end-of-lib {
     description
       "This feature indicates that the system allows to configure
        LDP end-of-lib capability.";
   }

   feature capability-mldp-hsmp {
     description
       "This feature indicates that the system allows to configure
        mLDP hub-and-spoke-multipoint capability.";
   }
   feature capability-mldp-node-protection {
     description
       "This feature indicates that the system allows to configure
        mLDP node-protection capability.";
   }

   feature capability-typed-wildcard-fec {
     description
       "This feature indicates that the system allows to configure
        LDP typed-wildcard-fec capability.";
   }

   feature capability-upstream-label-assignment {
     description
       "This feature indicates that the system allows to configure
        LDP upstream label assignment capability.";
   }

   feature forwarding-nexthop-config {
     description
       "This feature indicates that the system allows to configure
        forwarding nexthop on interfaces.";
   }

   feature global-session-authentication {
     description
       "This feature indicates that the system allows to configure
        authentication at global level.";
   }

   feature graceful-restart-helper-mode {
     description
       "This feature indicates that the system supports graceful
        restart helper mode.";
   }

   feature mldp {
     description
       "This feature indicates that the system supports Multicast
        LDP (mLDP).";
   }

   feature mldp-mofrr {
     description
       "This feature indicates that the system supports mLDP
        Multicast only FRR (MoFRR).";
   }
   feature per-interface-timer-config {
     description
       "This feature indicates that the system allows to configure
        interface hello timers at the per-interface level.";
   }

   feature per-peer-graceful-restart-config {
     description
       "This feature indicates that the system allows to configure
        graceful restart at the per-peer level.";
   }

   feature per-peer-session-attributes-config {
     description
       "This feature indicates that the system allows to configure
        session attributes at the per-peer level.";
   }

   feature policy-extended-discovery-config {
     description
       "This feature indicates that the system allows to configure
        policies to control the acceptance of extended neighbor
        discovery hello messages.";
   }

   feature policy-label-assignment-config {
     description
       "This feature indicates that the system allows to configure
        policies to assign labels according to certain prefixes.";
   }

   feature policy-ordered-label-config {
     description
       "This feature indicates that the system allows to configure
        ordered label policies.";
   }

   feature session-downstream-on-demand-config {
     description
       "This feature indicates that the system allows to configure
        session downstream-on-demand";
   }

   /*
    * Typedefs
    */
   typedef ldp-address-family {
     type identityref {
       base rt:address-family;
     }
     description
       "LDP address family type.";
   }

   typedef duration32-inf {
     type union {
       type uint32;
       type enumeration {
         enum "infinite" {
           description "The duration is infinite.";
         }
       }
     }
     units seconds;
     description
       "Duration represented as 32 bit seconds with infinite.";
   }

   typedef advertised-received {
     type enumeration {
       enum advertised {
         description "Advertised information.";
       }
       enum received {
         description "Received information.";
       }
     }
     description
       "Received or advertised.";
   }

   typedef downstream-upstream {
     type enumeration {
       enum downstream {
         description "Downstream information.";
       }
       enum upstream {
         description "Upstream information.";
       }
     }
     description
       "Received or advertised.";
   }

   typedef label-adv-mode {
     type enumeration {
       enum downstream-unsolicited {
         description "Downstream Unsolicited.";
       }
       enum downstream-on-demand {
         description "Downstream on Demand.";
       }
     }
     description
       "Label Advertisement Mode.";
   }

   typedef mpls-interface-ref {
     type leafref {
       path "/rt:routing/mpls:mpls/mpls:interface/mpls:name";
     }
     description
       "This type is used by data models that need to reference
        mpls interfaces.";
   }

   typedef multipoint-type {
     type enumeration {
       enum p2mp {
         description "Point to multipoint.";
       }
       enum mp2mp {
         description "Multipoint to multipoint.";
       }
     }
     description
       "p2mp or mp2mp.";
   }

   typedef neighbor-list-ref {
     type string;
     description
       "A type for a reference to a neighbor list.";
   }

   typedef peer-list-ref {
     type string;
     description
       "A type for a reference to a peer list.";
   }

   typedef prefix-list-ref {
     type string;
     description
```

```
       "A type for a reference to a prefix list.";
   }

   typedef oper-status-event-type {
     type enumeration {
       enum up {
         value 1;
         description
           "Operational status changed to up.";
       }
       enum down {
         value 2;
         description
           "Operational status changed to down.";
       }
     }
     description "Operational status event type for notifications.";
   }

   typedef route-distinguisher {
     type string {
     }
     description
       "Type definition for route distinguisher.";
     reference
       "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
   }

   /*
    * Identities
    */
   identity adjacency-flag-base {
     description "Base type for adjacency flags.";
   }

   identity adjacency-flag-active {
     base "adjacency-flag-base";
     description
       "This adjacency is configured and actively created.";
   }

   identity adjacency-flag-passive {
     base "adjacency-flag-base";
     description
       "This adjacency is not configured and passively accepted.";
   }

   /*
```

```
       * Groupings
       */

      grouping adjacency-state-attributes {
        description
          "Adjacency state attributes.";

        leaf-list flag {
          type identityref {
            base "adjacency-flag-base";
          }
          description "Adjacency flags.";
        }
        container hello-holdtime {
          description "Hello holdtime state.";
          leaf adjacent {
            type uint16;
            units seconds;
            description "Peer holdtime.";
          }
          leaf negotiated {
            type uint16;
            units seconds;
            description "Negotiated holdtime.";
          }
          leaf remaining {
            type uint16;
            units seconds;
            description "Remaining holdtime.";
          }
        }

        leaf next-hello {
          type uint16;
          units seconds;
          description "Time to send the next hello message.";
        }

        container statistics {
          description
            "Statistics objects.";

          leaf discontinuity-time {
            type yang:date-and-time;
            mandatory true;
            description
              "The time on the most recent occasion at which any one or
               more of this interface’s counters suffered a
```

```
             discontinuity.  If no such discontinuities have occurred
             since the last re-initialization of the local management
             subsystem, then this node contains the time the local
             management subsystem re-initialized itself.";
        }

        leaf hello-received {
          type yang:counter64;
          description
            "The number of hello messages received.";
        }
        leaf hello-dropped {
          type yang:counter64;
          description
            "The number of hello messages received.";
        }
      } // statistics
    } // adjacency-state-attributes

    grouping basic-discovery-timers {
      description
        "Basic discovery timer attributes.";
      leaf hello-holdtime {
        type uint16 {
          range 15..3600;
        }
        units seconds;
        description
          "The time interval for which a LDP link Hello adjacency
           is maintained in the absence of link Hello messages from
           the LDP neighbor";
      }
      leaf hello-interval {
        type uint16 {
          range 5..1200;
        }
        units seconds;
        description
          "The interval between consecutive LDP link Hello messages
           used in basic LDP discovery";
      }
    } // basic-discovery-timers

    grouping binding-address-state-attributes {
      description
        "Address binding attributes";
      leaf advertisement-type {
        type advertised-received;
        description
          "Received or advertised.";
      }
      leaf peer {
        type leafref {
          path "../../../../../../../peers/peer/lsr-id";
        }
        must "../advertisement-type = 'received'" {
          description
            "Applicable for received address.";
        }
        description
          "LDP peer from which this address is received.";
      } // peer
    } // binding-address-state-attributes

    grouping binding-label-state-attributes {
      description
        "Label binding attributes";
      list peer {
        key "peer advertisement-type";
        description
          "List of advertised and received peers.";
        leaf peer {
          type leafref {
            path "../../../../../../../../peers/peer/lsr-id";
          }
          description
            "LDP peer from which this binding is received,
             or to which this binding is advertised.";
        }
        leaf advertisement-type {
          type advertised-received;
          description
            "Received or advertised.";
        }
        leaf label {
          type mpls:mpls-label;
          description
            "Advertised (outbound) or received (inbound)
             label.";
        }
        leaf used-in-forwarding {
          type boolean;
          description
            "'true' if the lable is used in forwarding.";
        }
      } // peer
```

```
   } // binding-label-state-attributes

   grouping extended-discovery-policy-attributes {
     description
       "LDP policy to control the acceptance of extended neighbor
        discovery hello messages.";
     container hello-accept {
       if-feature policy-extended-discovery-config;
       description
         "Extended discovery acceptance policies.";

       leaf enable {
         type boolean;
         description
           "'true' to accept; 'false' to deny.";
       }
       leaf neighbor-list {
         type neighbor-list-ref;
         description

           "The name of a peer ACL.";
       }
     } // hello-accept
   } // extended-discovery-policy-attributes

   grouping extended-discovery-timers {
     description
       "Extended discovery timer attributes.";
     leaf hello-holdtime {
       type uint16 {
         range 15..3600;
       }
       units seconds;
       description
         "The time interval for which LDP targeted Hello adjacency

          is maintained in the absence of targeted Hello messages
          from an LDP neighbor.";
     }
     leaf hello-interval {
       type uint16 {
         range 5..3600;
       }
       units seconds;
       description
         "The interval between consecutive LDP targeted Hello
          messages used in extended LDP discovery.";
     }
   } // extended-discovery-timers

   grouping global-attributes {
     description "Configuration attributes at global level.";

     uses instance-attributes;
   } // global-attributes

   grouping graceful-restart-attributes {
     description
       "Graceful restart configuration attributes.";
     container graceful-restart {
       description
         "Attributes for graceful restart.";
       leaf enable {
         type boolean;
         description
           "Enable or disable graceful restart.";
       }
       leaf helper-enable {
         if-feature graceful-restart-helper-mode;
         type boolean;
         description
           "Enable or disable graceful restart helper mode.";
       }
       leaf reconnect-time {
         type uint16 {
           range 10..1800;
         }
         units seconds;
         description
           "Specifies the time interval that the remote LDP peer
            must wait for the local LDP peer to reconnect after the
            remote peer detects the LDP communication failure.";
       }
       leaf recovery-time {
         type uint16 {
           range 30..3600;
         }
         units seconds;
         description
           "Specifies the time interval, in seconds, that the remote
            LDP peer preserves its MPLS forwarding state after
            receiving the Initialization message from the restarted
            local LDP peer.";
       }
       leaf forwarding-holdtime {
         type uint16 {
           range 30..3600;
         }
         units seconds;
         description
           "Specifies the time interval, in seconds, before the
            termination of the recovery phase.";
       }
     } // graceful-restart
   } // graceful-restart-attributes

   grouping graceful-restart-attributes-per-peer {
     description
       "Per peer graceful restart configuration attributes.";
     container graceful-restart {
       description
         "Attributes for graceful restart.";
       leaf enable {
         type boolean;
         description
           "Enable or disable graceful restart.";
       }
       leaf reconnect-time {
         type uint16 {
           range 10..1800;
         }
         units seconds;
         description
           "Specifies the time interval that the remote LDP peer
            must wait for the local LDP peer to reconnect after the
            remote peer detects the LDP communication failure.";
       }
       leaf recovery-time {
         type uint16 {
           range 30..3600;
         }
         units seconds;
         description
           "Specifies the time interval, in seconds, that the remote
            LDP peer preserves its MPLS forwarding state after
            receiving the Initialization message from the restarted
            local LDP peer.";
       }
     } // graceful-restart
   } // graceful-restart-attributes-per-peer

   grouping instance-attributes {
     description "Configuration attributes at instance level.";
     container capability {
       description "Configure capability.";
       container end-of-lib {
         if-feature capability-end-of-lib;
         description
           "Configure end-of-lib capability.";
         leaf enable {
           type boolean;
           description
             "Enable end-of-lib capability.";
         }
       }
       container typed-wildcard-fec {
         if-feature capability-typed-wildcard-fec;
         description
           "Configure typed-wildcard-fec capability.";
         leaf enable {
           type boolean;
           description
             "Enable typed-wildcard-fec capability.";
         }
       }
       container upstream-label-assignment {
         if-feature capability-upstream-label-assignment;
         description
           "Configure upstream label assignment capability.";
         leaf enable {
           type boolean;
           description
             "Enable upstream label assignment.";
         }
       }
       container mldp {
         if-feature mldp;

         description
           "Multipoint capabilities.";
         uses mldp-capabilities;
       }
     } // capability

     uses graceful-restart-attributes;

     leaf igp-synchronization-delay {
       type uint16 {
         range 3..60;
       }
       units seconds;
       description
         "Sets the interval that the LDP waits before notifying the
          Interior Gateway Protocol (IGP) that label exchange is
          completed so that IGP can start advertising the normal
          metric for the link.";
     }
     leaf lsr-id {
       type yang:dotted-quad;
       description "Router ID.";
     }
   } // instance-attributes

   grouping ldp-adjacency-ref {
     description
       "An absolute reference to an LDP adjacency.";
     choice hello-adjacency-type {
       description
         "Interface or targeted adjacency.";
       case targeted {
         container targeted {
           description "Targeted adjacency.";
           leaf target-address {
             type inet:ip-address;
             description
               "The target address.";
           }
         } // targeted
       }
       case link {
         container link {
           description "Link adjacency.";
           leaf next-hop-interface {
             type mpls-interface-ref;
             description
               "Interface connecting to next-hop.";
           }
           leaf next-hop-address {
             type inet:ip-address;
             must "../next-hop-interface" {
               description
                 "Applicable when interface is specified.";

             }
             description
               "IP address of next-hop.";
           }
         } // link
       }
     }
   } // ldp-adjacency-ref

   grouping ldp-fec-event {
     description
       "A LDP FEC event.";
     leaf prefix {
       type inet:ip-prefix;
       description
         "FEC.";
     }
   } // ldp-fec-event

   grouping ldp-peer-ref {
     description
       "An absolute reference to an LDP peer.";
     leaf peer-ref {
       type leafref {
         path "/rt:routing/rt:control-plane-protocols/mpls-ldp/"
           + "peers/peer/lsr-id";
       }
       description
         "Reference to an LDP peer.";
     }
   } // ldp-peer-ref

   grouping mldp-capabilities {
     description
       "mLDP capabilities.";
     container p2mp {
       description
         "Configure point-to-multipoint capability.";
       leaf enable {
         type boolean;
         description
           "Enable point-to-multipoint.";
       }
     }
     container mp2mp {
       description
         "Configure multipoint-to-multipoint capability.";
       leaf enable {
         type boolean;
         description
           "Enable multipoint-to-multipoint.";
       }
     }
     container make-before-break {
       description
         "Configure make-before-break capability.";
       leaf enable {
         type boolean;
         description
           "Enable make-before-break.";
       }
       leaf switchover-delay {
         type uint16;
         units seconds;
         description
           "Switchover delay in seconds.";
       }
       leaf timeout {
         type uint16;
         units seconds;
         description
           "Timeout in seconds.";
       }
     }
     container hub-and-spoke {
       if-feature capability-mldp-hsmp;
       description
         "Configure hub-and-spoke-multipoint capability.";
       reference
         "RFC7140: LDP Extensions for Hub and Spoke Multipoint
          Label Switched Path";
       leaf enable {
         type boolean;
         description
           "Enable hub-and-spoke-multipoint.";
       }
     }
     container node-protection {
       if-feature capability-mldp-node-protection;
       description
         "Configure node-protection capability.";
       reference
         "RFC7715: mLDP Node Protection.";
       leaf plr {
         type boolean;
         description
           "Point of Local Repair capable for MP LSP node
            protection.";
       }
       container merge-point {
         description
           "Merge Point capable for MP LSP node protection.";
```

```
         leaf enable {
           type boolean;
           description
             "Enable merge point capability.";
         }
         leaf targeted-session-teardown-delay {
           type uint16;
           units seconds;
           description
             "Targeted session teardown delay.";
         }
       } // merge-point
     }
   } // mldp-capabilities

   grouping  mldp-configured-lsp-roots {
     description
       "mLDP roots containers.";

     container roots-ipv4 {

       when "../../../af = 'ipv4'" {
         description
           "Only for IPv4.";
       }
       description
         "Configured IPv4 multicast LSPs.";
       list root {
         key "root-address";
         description
           "List of roots for configured multicast LSPs.";

         leaf root-address {
           type inet:ipv4-address;
           description
             "Root address.";
         }

         list lsp {
           must "(lsp-id = 0 and source-address != '0.0.0.0' and "
             + "group-address != '0.0.0.0') or "
             + "(lsp-id != 0 and source-address = '0.0.0.0' and "
             + "group-address = '0.0.0.0')" {
             description
               "A LSP can be identified by either <lsp-id> or
                <source-address, group-address>.";
           }
           key "lsp-id source-address group-address";
           description
             "List of LSPs.";
           leaf lsp-id {
             type uint16;
             description "ID to identify the LSP.";
           }
           leaf source-address {
             type inet:ipv4-address;
             description
               "Source address.";
           }
           leaf group-address {
             type inet:ipv4-address-no-zone;
             description
               "Group address.";
           }
         } // list lsp
       } // list root
     } // roots-ipv4

     container roots-ipv6 {

       when "../../../af = 'ipv6'" {
         description
           "Only for IPv6.";
       }
       description
         "Configured IPv6 multicast LSPs.";

       list root {
         key "root-address";
         description
           "List of roots for configured multicast LSPs.";

         leaf root-address {
           type inet:ipv6-address;
           description
             "Root address.";
         }

         list lsp {
           must "(lsp-id = 0 and source-address != '::' and "
             + "group-address != '::') or "
             + "(lsp-id != 0 and source-address = '::' and "
             + "group-address = '::')" {
             description
               "A LSP can be identified by either <lsp-id> or
                <source-address, group-address>.";
           }
           key "lsp-id source-address group-address";
           description
             "List of LSPs.";
           leaf lsp-id {
             type uint16;
             description "ID to identify the LSP.";
           }
           leaf source-address {
             type inet:ipv6-address;
             description
               "Source address.";
           }
           leaf group-address {
             type inet:ipv6-address-no-zone;
             description
               "Group address.";
           }
         } // list lsp
       } // list root
     } // roots-ipv6
   } // mldp-configured-lsp-roots

   grouping mldp-fec-event {
     description
       "A mLDP FEC event.";
     leaf tree-type {
       type multipoint-type;
       description
         "p2mp or mp2mp.";
     }
     leaf root {
       type inet:ip-address;
       description
         "Root address.";
     }
     choice lsp-key-type {
       description
         "LSP ID based or source-group based .";
       case lsp-id-based {
         leaf lsp-id {
           type uint16;
           description
             "ID to identify the LSP.";
         }
       }
       case source-group-based {
         leaf source-address {
           type inet:ip-address;
           description

             "LSP source address.";
         }
         leaf group-address {
           type inet:ip-address;
           description
             "Multicast group address.";
         }
       } // case source-group-based
     }
   } // mldp-fec-event

   grouping  mldp-binding-label-state-attributes {
     description
       "mLDP label binding attributes.";

     leaf multipoint-type {
       type multipoint-type;
       description
         "The type of mutipoint, p2mp or mp2mp.";
     }
     list peer {
       key "direction peer advertisement-type";
       description
         "List of advertised and received peers.";
       leaf direction {
         type downstream-upstream;
         description
           "Downstream or upstream.";
       }
       leaf peer {
         type leafref {
           path
             "../../../../../../../../../../../peers/peer/lsr-id";
         }
         description
           "LDP peer from which this binding is received,
            or to which this binding is advertised.";
       }
       leaf advertisement-type {
         type advertised-received;
         description
           "Advertised or received.";
       }
       leaf label {
         type mpls:mpls-label;
```

```
          description
            "Advertised (outbound) or received (inbound) label.";
        }
        leaf mbb-role {
          when "../direction = 'upstream'" {
            description
              "For upstream.";
          }
          type enumeration {
            enum none {
              description "MBB is not enabled.";
            }
            enum active {
              description "This LSP is active.";
            }
            enum inactive {
              description "This LSP is inactive.";
            }
          }
          description
            "The MBB status of this LSP.";
        }
        leaf mofrr-role {
          when "../direction = 'upstream'" {
            description
              "For upstream.";
          }
          type enumeration {
            enum none {
              description "MOFRR is not enabled.";
            }
            enum primary {
              description "This LSP is primary.";
            }
            enum backup {
              description "This LSP is backup.";
            }
          }
          description
            "The MOFRR status of this LSP.";
        }
      } // peer
    } // mldp-binding-label-state-attributes

    grouping peer-af-policy-container {
      description
        "LDP policy attribute container under peer address-family.";
      container label-policy {
        description
          "Label policy attributes.";
        container advertise {
          description
            "Label advertising policies.";
          leaf prefix-list {
            type prefix-list-ref;
              description
                "Applies the prefix list to outgoing label
                 advertisements.";
          }
        }
        container accept {
          description
            "Label advertisement acceptance policies.";
          leaf prefix-list {
            type prefix-list-ref;
            description
              "Applies the prefix list to incoming label
               advertisements.";
          }
        } // accept
      } // label-policy
    } // peer-af-policy-container

    grouping peer-attributes {
      description "Peer configuration attributes.";

      leaf session-ka-holdtime {
        type uint16 {
          range 45..3600;
        }
        units seconds;
        description
          "The time interval after which an inactive LDP session
           terminates and the corresponding TCP session closes.
           Inactivity is defined as not receiving LDP packets from the
           peer.";
      }
      leaf session-ka-interval {
        type uint16 {
          range 15..1200;
        }
        units seconds;
        description
          "The interval between successive transmissions of keepalive
           packets. Keepalive packets are only sent in the absence of
           other LDP packets transmitted over the LDP session.";
      }
    } // peer-attributes

    grouping peer-authentication {
      description
        "Peer authentication attributes.";
      leaf session-authentication-md5-password {
        type string {
          length "1..80";
        }
        description
          "Assigns an encrypted MD5 password to an LDP
           peer";
      } // md5-password
    } // peer-authentication

    grouping peer-state-derived {
      description "Peer derived state attributes.";

      container label-advertisement-mode {
        description "Label advertisement mode state.";
        leaf local {
          type label-adv-mode;
          description
            "Local Label Advertisement Mode.";
        }
        leaf peer {
          type label-adv-mode;
          description
            "Peer Label Advertisement Mode.";
        }
        leaf negotiated {
          type label-adv-mode;
          description
            "Negotiated Label Advertisement Mode.";
        }
      }
      leaf next-keep-alive {
        type uint16;
        units seconds;
        description "Time to send the next KeepAlive message.";
      }

      leaf peer-ldp-id {
        type yang:dotted-quad;
        description "Peer LDP ID.";
      }
      container received-peer-state {
        description "Peer features.";

        uses graceful-restart-attributes-per-peer;

        container capability {
          description "Configure capability.";
          container end-of-lib {
            description
              "Configure end-of-lib capability.";
            leaf enable {
              type boolean;
              description
                "Enable end-of-lib capability.";
            }
          }
          container typed-wildcard-fec {
            description
              "Configure typed-wildcard-fec capability.";
            leaf enable {
              type boolean;
              description
                "Enable typed-wildcard-fec capability.";
            }
          }
          container upstream-label-assignment {
            description
              "Configure upstream label assignment capability.";
            leaf enable {
              type boolean;
              description
                "Enable upstream label assignment.";
            }
          }
          container mldp {
            if-feature mldp;
            description
              "Multipoint capabilities.";

            container p2mp {
              description
                "Configure point-to-multipoint capability.";
              leaf enable {
                type boolean;
                description
                  "Enable point-to-multipoint.";
              }
            }
            container mp2mp {
              description
                "Configure multipoint-to-multipoint capability.";
              leaf enable {
                type boolean;
                description
                  "Enable multipoint-to-multipoint.";
              }
            }
            container make-before-break {
              description
                "Configure make-before-break capability.";
              leaf enable {
                type boolean;
                description
                  "Enable make-before-break.";
              }
            }
            container hub-and-spoke {
              description
                "Configure hub-and-spoke-multipoint capability.";
              reference
                "RFC7140: LDP Extensions for Hub and Spoke Multipoint
                 Label Switched Path";
              leaf enable {
                type boolean;
                description
                  "Enable hub-and-spoke-multipoint.";
              }
            }
            container node-protection {
              description
                "Configure node-protection capability.";
              reference
                "RFC7715: mLDP Node Protection.";
              leaf plr {
                type boolean;
                description
                  "Point of Local Repair capable for MP LSP node
                   protection.";
              }
              leaf merge-point {
                type boolean;
                description
                  "Merge Point capable for MP LSP node protection.";
              } // merge-point
            } // node-protection
          } // mldp
```

```
        } // capability
      } // received-peer-state

      container session-holdtime {
        description "Session holdtime state.";
        leaf peer {
          type uint16;
          units seconds;
          description "Peer holdtime.";
        }
        leaf negotiated {
          type uint16;
          units seconds;
          description "Negotiated holdtime.";
        }
        leaf remaining {
          type uint16;
          units seconds;
          description "Remaining holdtime.";
        }
      } // session-holdtime

      leaf session-state {
        type enumeration {
          enum non-existent {
            description "NON EXISTENT state. Transport disconnected.";
          }
          enum initialized {
            description "INITIALIZED state.";
          }
          enum openrec {
            description "OPENREC state.";
          }
          enum opensent {
            description "OPENSENT state.";
          }
          enum operational {
            description "OPERATIONAL state.";
          }
        }
        description
          "Representing the operational status.";
      }

      container tcp-connection {
        description "TCP connection state.";
        leaf local-address {
          type inet:ip-address;
```

```
         description "Local address.";
       }
       leaf local-port {
         type inet:port-number;
         description "Local port.";
       }
       leaf remote-address {
         type inet:ip-address;
         description "Remote address.";
       }
       leaf remote-port {
         type inet:port-number;
         description "Remote port.";
       }
     } // tcp-connection

     leaf up-time {
       type string;
       description "Up time. The interval format in ISO 8601.";
     }

     container statistics {
       description
         "Statistics objects.";

       leaf discontinuity-time {
         type yang:date-and-time;
         mandatory true;
         description
           "The time on the most recent occasion at which any one or
            more of this interface’s counters suffered a
            discontinuity.  If no such discontinuities have occurred
            since the last re-initialization of the local management
            subsystem, then this node contains the time the local
            management subsystem re-initialized itself.";
       }

       container received {
         description "Inbound statistics.";
         uses statistics-peer-received-sent;
       }
       container sent {
         description "Outbound statistics.";
         uses statistics-peer-received-sent;
       }

       leaf total-addresses {
         type uint32;
         description
           "The number of learned addresses.";
       }
       leaf total-labels {
         type uint32;
         description
           "The number of learned labels.";
       }
       leaf total-fec-label-bindings {
         type uint32;
         description
           "The number of learned label-address bindings.";
       }
     } // statistics
   } // peer-state-derived

   grouping policy-container {
     description
       "LDP policy attributes.";
     container label-policy {
       description
         "Label policy attributes.";
       container independent-mode {
         description
           "Independent label policy attributes.";
         container assign {

           if-feature policy-label-assignment-config;
           description
             "Label assignment policies";
           choice prefix-option {
             description
               "Use either prefix-list or host-routes-only.";
             case prefix-list {
               leaf prefix-list {
                 type prefix-list-ref;
                 description
                   "Assign labels according to certain prefixes.";
               }
             }
             case host-routes-only {
               leaf host-routes-only {
                 type boolean;
                 description
                   "'true' to apply host routes only.";
               }
             }
           } // prefix-option
         }
         container advertise {
           description
             "Label advertising policies.";
           container explicit-null {
             description
               "Enables an egress router to advertise an
                explicit null label (value 0) in place of an
                implicit null label (value 3) to the
                penultimate hop router.";
             leaf enable {
               type boolean;
               description
                 "'true' to enable explicit null.";
             }
             leaf prefix-list {
               type prefix-list-ref;
               description
                 "Prefix list name. Applies the filters in the
                  specified prefix list to label
                  advertisements.
                  If the prefix list is not specified, explicit
                  null label advertisement is enabled for all
                  directly connected prefixes.";
             }
           }
           leaf prefix-list {
             type prefix-list-ref;
             description
               "Applies the prefix list to outgoing label
                advertisements.";

           }
         }
         container accept {
           description
             "Label advertisement acceptance policies.";
           leaf prefix-list {
             type prefix-list-ref;
             description
               "Applies the prefix list to incoming label
                advertisements.";
           }
         }
       } // independent-mode
       container ordered-mode {
         if-feature policy-ordered-label-config;
         description
```

```
           "Ordered label policy attributes.";
         container egress-lsr {
           description
             "Egress LSR label assignment policies";
           leaf prefix-list {
             type prefix-list-ref;
             description
               "Assign labels according to certain prefixes.";
           }
         }
         container advertise {
           description
             "Label advertising policies.";
           leaf prefix-list {
             type prefix-list-ref;
             description
               "Applies the prefix list to outgoing label
                advertisements.";
           }
         }
         container accept {
           description
             "Label advertisement acceptance policies.";
           leaf prefix-list {
             type prefix-list-ref;
             description
               "Applies the prefix list to incoming label
                advertisements.";
           }
         }
       } // ordered-mode
     } // label-policy
   } // policy-container

   grouping statistics-peer-received-sent {
     description
       "Inbound and outbound statistic counters.";
     leaf total-octets {
       type yang:counter64;
       description
         "The total number of octets sent or received.";
     }
     leaf total-messages {
       type yang:counter64;
       description
         "The number of messages sent or received.";
     }
     leaf address {
       type yang:counter64;
       description
         "The number of address messages sent or received.";
     }
     leaf address-withdraw {
       type yang:counter64;
       description
         "The number of address-withdraw messages sent or received.";
     }
     leaf initialization {
       type yang:counter64;
       description
         "The number of initialization messages sent or received.";
     }
     leaf keepalive {
       type yang:counter64;
       description
         "The number of keepalive messages sent or received.";
     }
     leaf label-abort-request {
       type yang:counter64;
       description
         "The number of label-abort-request messages sent or
          received.";
     }
     leaf label-mapping {
       type yang:counter64;
       description
         "The number of label-mapping messages sent or received.";
     }
     leaf label-release {
       type yang:counter64;
       description
         "The number of label-release messages sent or received.";
     }
     leaf label-request {
       type yang:counter64;
       description
         "The number of label-request messages sent or received.";
     }
     leaf label-withdraw {
       type yang:counter64;
       description
         "The number of label-withdraw messages sent or received.";
     }
     leaf notification {
       type yang:counter64;
       description
```

```
         "The number of messages sent or received.";
     }
   } // statistics-peer-received-sent

   /*
    * Configuration data nodes
    */

   augment "/rt:routing/rt:control-plane-protocols" {
     description "LDP augmentation.";

     container mpls-ldp {
       presence "Container for LDP protocol.";
       description
         "Container for LDP protocol.";

       container global {
         description
           "Global attributes for LDP.";
         container config {
           description
             "Configuration data.";
           uses global-attributes;
         }
         container state {
           config false;
           description
             "Operational state data.";
           uses global-attributes;
         }

         container mldp {
           if-feature mldp;
           description
             "mLDP attributes at per instance level. Defining
              attributes here does not enable any MP capabilities.
              MP capabilities need to be explicitly enabled under
              container capability.";

           container config {
             description
               "Configuration data.";
             leaf enable {
               type boolean;
               description
                 "Enable mLDP.";
             }
           }
           container state {
             config false;
             description

               "Operational state data.";
             leaf enable {
               type boolean;
               description
                 "Enable mLDP.";
             }
           }

           list address-family {
             key "afi";
             description
               "Per-af params.";
             leaf afi {
               type ldp-address-family;
               description
                 "Address family type value.";
             }

             container config {
               description
                 "Configuration data.";
               container multicast-only-frr {
                 if-feature mldp-mofrr;
                 description
                   "Multicast only FRR (MoFRR) policy.";
                 leaf prefix-list {
                   type prefix-list-ref;
                   description
                     "Enables MoFRR for the specified access list.";
                 }
               } // multicast-only-frr
               container recursive-fec {
                 description
                   "Recursive FEC policy.";
                 leaf prefix-list {
                   type prefix-list-ref;
                   description
                     "Enables recursive FEC for the specified access
                      list.";
                 }
               } // recursive-for
             }
             container state {
               config false;
               description
                 "Operational state data.";
               container multicast-only-frr {
                 if-feature mldp-mofrr;

                 description
                   "Multicast only FRR (MoFRR) policy.";
                 leaf prefix-list {
                   type prefix-list-ref;
                   description
                     "Enables MoFRR for the specified access list.";
                 }
               } // multicast-only-frr
               container recursive-fec {
                 description
                   "Recursive FEC policy.";
                 leaf prefix-list {
                   type prefix-list-ref;
                   description
                     "Enables recursive FEC for the specified access
                      list.";
                 }
               } // recursive-fec

               container ipv4 {
                 when "../../afi = 'ipv4'" {
                   description
                     "Only for IPv4.";
                 }
                 description
                   "IPv4 state information.";
                 container roots {
                   description
                     "IPv4 multicast LSP roots.";
                   list root {
                     key "root-address";
                     description
                       "List of roots for configured multicast LSPs.";

                     leaf root-address {
                       type inet:ipv4-address;
                       description
                         "Root address.";
                     }

                     leaf is-self {
                       type boolean;
                       description
```

```
                    "This is the root.";
                }

                list reachability {
                  key "address interface";
                  description
                    "A next hop for reachability to root,
                     as a RIB view.";
                  leaf address {
                    type inet:ipv4-address;
                    description
                      "The next hop address to reach root.";
                  }
                  leaf interface {
                    type mpls-interface-ref;
                    description
                      "Interface connecting to next-hop.";
                  }
                  leaf peer {
                    type leafref {
                      path
                        "../../../../../../../../../../peers/peer/"
                        + "lsr-id";
                    }
                    description
                      "LDP peer from which this next hop can be
                       reached.";
                  }
                }
              } // list root
            } // roots
            container bindings {
              description
                "mLDP FEC to label bindings.";
              container opaque-type-lspid {
                description
                  "The type of opaque value element is
                   the generic LSP identifier";
                reference
                  "RFC6388: Label Distribution Protocol
                   Extensions for Point-to-Multipoint and
                   Multipoint-to-Multipoint Label Switched
                   Paths.";
                list fec-label {
                  key
                    "root-address lsp-id "
                    + "recur-root-address recur-rd";
                  description
```

```
                        "List of FEC to label bindings.";
                      leaf root-address {
                        type inet:ipv4-address;
                        description
                          "Root address.";
                      }
                      leaf lsp-id {
                        type uint32;
                        description "ID to identify the LSP.";
                      }
                      leaf recur-root-address {
                        type inet:ip-address;
                        description
                          "Recursive root address.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      leaf recur-rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher in the VPN-Recursive
                           Opaque Value.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      uses mldp-binding-label-state-attributes;
                    } // fec-label
                  } // opaque-type-lspid

                  container opaque-type-src {
                    description
                      "The type of opaque value element is
                       the transit source TLV";
                    reference
                      "RFC6826: Multipoint LDP In-Band Signaling for
                       Point-to-Multipoint and
                       Multipoint-to-Multipoint Label Switched
                       Paths.";
                    list fec-label {
                      key
                        "root-address source-address group-address "
                        + "rd recur-root-address recur-rd";
                      description
                        "List of FEC to label bindings.";
                      leaf root-address {
                        type inet:ipv4-address;
                        description
                          "Root address.";
                      }
                      leaf source-address {
                        type inet:ip-address;
                        description
                          "Source address.";
                      }
                      leaf group-address {
                        type inet:ip-address-no-zone;
                        description
                          "Group address.";
                      }
                      leaf rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher.";
                        reference
                          "RFC7246: Multipoint Label Distribution
                           Protocol In-Band Signaling in a Virtual
                           Routing and Forwarding (VRF) Table
                           Context.";
                      }
                      leaf recur-root-address {
                        type inet:ip-address;
                        description
                          "Recursive root address.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      leaf recur-rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher in the VPN-Recursive
                           Opaque Value.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      uses mldp-binding-label-state-attributes;
                    } // fec-label
                  } // opaque-type-src

                  container opaque-type-bidir {
                    description
                      "The type of opaque value element is
                       the generic LSP identifier";
```

```
                      reference
                        "RFC6826: Multipoint LDP In-Band Signaling for
                         Point-to-Multipoint and
                         Multipoint-to-Multipoint Label Switched
                         Paths.";
                      list fec-label {
                        key
                          "root-address rp group-address "
                          + "rd recur-root-address recur-rd";
                        description
                          "List of FEC to label bindings.";
                        leaf root-address {
                          type inet:ipv4-address;
                          description
                            "Root address.";
                        }
                        leaf rp {
                          type inet:ip-address;
                          description
                            "RP address.";
                        }
                        leaf group-address {
                          type inet:ip-address-no-zone;
                          description
                            "Group address.";
                        }
                        leaf rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher.";
                          reference
                            "RFC7246: Multipoint Label Distribution
                             Protocol In-Band Signaling in a Virtual
                             Routing and Forwarding (VRF) Table
                             Context.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
                            "Recursive root address.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        leaf recur-rd {
                          type route-distinguisher;
                          description
                            "Route Distinguisher in the VPN-Recursive
                             Opaque Value.";
                          reference
                            "RFC6512: Using Multipoint LDP When the
                             Backbone Has No Route to the Root";
                        }
                        uses mldp-binding-label-state-attributes;
                      } // fec-label
                    } // opaque-type-bidir
                  } // bindings
                } // ipv4

                container ipv6 {
                  when "../../afi = 'ipv6'" {
                    description
                      "Only for IPv6.";
                  }
                  description
                    "IPv6 state information.";
                  container roots {
                    description
                      "IPv6 multicast LSP roots.";
                    list root {
                      key "root-address";
                      description
                        "List of roots for configured multicast LSPs.";

                      leaf root-address {
                        type inet:ipv6-address;
                        description
                          "Root address.";
                      }

                      leaf is-self {
                        type boolean;
                        description
                          "This is the root.";
                      }

                      list reachability {
                        key "address interface";
                        description
                          "A next hop for reachability to root,
                           as a RIB view.";
                        leaf address {
                          type inet:ipv6-address;
                          description
                            "The next hop address to reach root.";
                        }
```

```
                        leaf interface {
                          type mpls-interface-ref;
                          description
                            "Interface connecting to next-hop.";
                        }
                        leaf peer {
                          type leafref {
                            path
                              "../../../../../../../../../../peers/peer/"
                              + "lsr-id";
                          }
                          description
                            "LDP peer from which this next hop can be
                             reached.";
                        }
                      }
                    } // list root
                  } // roots
                  container bindings {
                    description
                      "mLDP FEC to label bindings.";
                    container opaque-type-lspid {
                      description
                        "The type of opaque value element is
                         the generic LSP identifier";
                      reference
                        "RFC6388: Label Distribution Protocol
                         Extensions for Point-to-Multipoint and
                         Multipoint-to-Multipoint Label Switched
                         Paths.";
                      list fec-label {
                        key
                          "root-address lsp-id "
                          + "recur-root-address recur-rd";
                        description
                          "List of FEC to label bindings.";
                        leaf root-address {
                          type inet:ipv6-address;
                          description
                            "Root address.";
                        }
                        leaf lsp-id {
                          type uint32;
                          description "ID to identify the LSP.";
                        }
                        leaf recur-root-address {
                          type inet:ip-address;
                          description
```

```
                   "Recursive root address.";
                 reference
                   "RFC6512: Using Multipoint LDP When the
                    Backbone Has No Route to the Root";
               }
               leaf recur-rd {
                 type route-distinguisher;
                 description
                   "Route Distinguisher in the VPN-Recursive
                    Opaque Value.";
                 reference
                   "RFC6512: Using Multipoint LDP When the
                    Backbone Has No Route to the Root";
               }
               uses mldp-binding-label-state-attributes;
             } // fec-label
           } // opaque-type-lspid

           container opaque-type-src {
             description
               "The type of opaque value element is
                the transit Source TLV";
             reference
               "RFC6826: Multipoint LDP In-Band Signaling for
                Point-to-Multipoint and
                Multipoint-to-Multipoint Label Switched
                Paths.";
             list fec-label {
               key
                 "root-address source-address group-address "
                 + "rd recur-root-address recur-rd";
               description
                 "List of FEC to label bindings.";
               leaf root-address {
                 type inet:ipv6-address;
                 description
                   "Root address.";
               }
               leaf source-address {
                 type inet:ip-address;
                 description
                   "Source address.";
               }
               leaf group-address {
                 type inet:ip-address-no-zone;
                 description
                   "Group address.";
               }
```

```
                   leaf rd {
                     type route-distinguisher;
                     description
                       "Route Distinguisher.";
                     reference
                       "RFC7246: Multipoint Label Distribution
                        Protocol In-Band Signaling in a Virtual
                        Routing and Forwarding (VRF) Table
                        Context.";
                   }
                   leaf recur-root-address {
                     type inet:ip-address;
                     description
                       "Recursive root address.";
                     reference
                       "RFC6512: Using Multipoint LDP When the
                        Backbone Has No Route to the Root";
                   }
                   leaf recur-rd {
                     type route-distinguisher;
                     description
                       "Route Distinguisher in the VPN-Recursive
                        Opaque Value.";
                     reference
                       "RFC6512: Using Multipoint LDP When the
                        Backbone Has No Route to the Root";
                   }
                   uses mldp-binding-label-state-attributes;
                 } // fec-label
               } // opaque-type-src

               container opaque-type-bidir {
                 description
                   "The type of opaque value element is
                    the generic LSP identifier";
                 reference
                   "RFC6826: Multipoint LDP In-Band Signaling for
                    Point-to-Multipoint and
                    Multipoint-to-Multipoint Label Switched
                    Paths.";
                 list fec-label {
                   key
                     "root-address rp group-address "
                     + "rd recur-root-address recur-rd";
                   description
                     "List of FEC to label bindings.";
                   leaf root-address {
                     type inet:ipv6-address;
```

```
                        description
                          "Root address.";
                      }
                      leaf rp {
                        type inet:ip-address;
                        description
                          "RP address.";
                      }
                      leaf group-address {
                        type inet:ip-address-no-zone;
                        description
                          "Group address.";
                      }
                      leaf rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher.";
                        reference
                          "RFC7246: Multipoint Label Distribution
                           Protocol In-Band Signaling in a Virtual
                           Routing and Forwarding (VRF) Table
                           Context.";
                      }
                      leaf recur-root-address {
                        type inet:ip-address;
                        description
                          "Recursive root address.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      leaf recur-rd {
                        type route-distinguisher;
                        description
                          "Route Distinguisher in the VPN-Recursive
                           Opaque Value.";
                        reference
                          "RFC6512: Using Multipoint LDP When the
                           Backbone Has No Route to the Root";
                      }
                      uses mldp-binding-label-state-attributes;
                    } // fec-label
                  } // opaque-type-bidir
                } // bindings
              } // ipv6
            } // state

            container configured-leaf-lsps {
              description
                "Configured multicast LSPs.";

              container p2mp {
                description
                  "Configured point-to-multipoint LSPs.";
                uses mldp-configured-lsp-roots;
              }
              container mp2mp {
                description
                  "Configured multipoint-to-multipoint LSPs.";
                uses mldp-configured-lsp-roots;
              }
            } // configured-leaf-lsps
          } // list address-family
        } // mldp

        list address-family {
          key "afi";
          description
            "Per-vrf per-af params.";
          leaf afi {
            type ldp-address-family;
            description
              "Address family type value.";
          }

          container config {
            description
              "Configuration data.";
            leaf enable {
              type boolean;
              description
                "'true' to enable the address family.";
            }
            uses policy-container;

            container ipv4 {
              when "../../afi = 'ipv4'" {
                description
                  "Only for IPv4.";
              }
              description
                "IPv4 address family.";
              leaf transport-address {
                type inet:ipv4-address;
                description
                  "The transport address advertised in LDP Hello
```

```
                    messages.";
                }
              } // ipv4
              container ipv6 {
                when "../../afi = 'ipv6'" {
                  description
                    "Only for IPv6.";
                }
                description
                  "IPv6 address family.";
                leaf transport-address {
                  type inet:ipv6-address;
                  description
                    "The transport address advertised in LDP Hello
                     messages.";
                }
              } // ipv6
            }
            container state {
              config false;
              description
                "Operational state data.";
              leaf enable {
                type boolean;
                description
                  "'true' to enable the address family.";
              }

              uses policy-container;

              container ipv4 {
                when "../../afi = 'ipv4'" {
                  description
                    "Only for IPv4.";
                }
                description
                  "IPv4 address family.";
                leaf transport-address {
                  type inet:ipv4-address;
                  description
                    "The transport address advertised in LDP Hello
                     messages.";
                }

                container bindings {
                  description
                    "LDP address and label binding information.";
                  list address {
                    key "address";
                    description
                      "List of address bindings.";
                    leaf address {
                      type inet:ipv4-address;
                      description
                        "Binding address.";
                    }
                    uses binding-address-state-attributes;
                  } // binding-address

                  list fec-label {
                    key "fec";
                    description
                      "List of label bindings.";
                    leaf fec {
                      type inet:ipv4-prefix;
                      description
                        "Prefix FEC.";
                    }
                    uses binding-label-state-attributes;
                  } // fec-label
                } // binding
              } // ipv4
              container ipv6 {
                when "../../afi = 'ipv6'" {
                  description
                    "Only for IPv6.";
                }
                description
                  "IPv6 address family.";
                leaf transport-address {
                  type inet:ipv6-address;
                  description
                    "The transport address advertised in LDP Hello
                     messages.";
                }

                container binding {
                  description
                    "LDP address and label binding information.";
                  list address {
                    key "address";
                    description
                      "List of address bindings.";
                    leaf address {
                      type inet:ipv6-address;
                      description
```

```
                      "Binding address.";
                  }
                  uses binding-address-state-attributes;
                } // binding-address

                list fec-label {
                  key "fec";
                  description
                    "List of label bindings.";
                  leaf fec {
                    type inet:ipv6-prefix;
                    description
                      "Prefix FEC.";
                  }
                  uses binding-label-state-attributes;
                } // fec-label
              } // binding
            } // ipv6
          } // state
        } // address-family

        container discovery {
          description
            "Neibgbor discovery configuration.";

          container interfaces {
            description
              "A list of interfaces for basic descovery.";
            container config {
              description
                "Configuration data.";
              uses basic-discovery-timers;
            }
            container state {
              config false;
              description

                "Operational state data.";
              uses basic-discovery-timers;
            }

            list interface {
              key "interface";
              description
                "List of LDP interfaces.";
              leaf interface {
                type mpls-interface-ref;
                description
```

```
                   "Interface.";
               }
               container config {
                 description
                   "Configuration data.";
                 uses basic-discovery-timers {
                   if-feature per-interface-timer-config;
                 }
                 leaf igp-synchronization-delay {
                   if-feature per-interface-timer-config;
                   type uint16 {
                     range 3..60;
                   }
                   units seconds;
                   description
                     "Sets the interval that the LDP waits before
                      notifying the Interior Gateway Protocol (IGP)
                      that label exchange is completed so that IGP
                      can start advertising the normal metric for
                      the link.";
                 }
               }
               container state {
                 config false;
                 description
                   "Operational state data.";
                 uses basic-discovery-timers {
                   if-feature per-interface-timer-config;
                 }
                 leaf igp-synchronization-delay {
                   if-feature per-interface-timer-config;
                   type uint16 {
                     range 3..60;
                   }
                   units seconds;
                   description
                     "Sets the interval that the LDP waits before
                      notifying the Interior Gateway Protocol (IGP)
                      that label exchange is completed so that IGP
                      can start advertising the normal metric for
                      the link.";
                 }
                 leaf next-hello {
                   type uint16;
                   units seconds;
                   description "Time to send the next hello message.";
                 }
               } // state
               list address-family {
                 key "afi";
                 description
                   "Per-vrf per-af params.";
                 leaf afi {
                   type ldp-address-family;
                   description
                     "Address family type value.";
                 }
                 container config {
                   description
                     "Configuration data.";
                   leaf enable {
                     type boolean;
                     description
                       "Enable the address family on the interface.";
                   }

                   container ipv4 {
                     must "/if:interfaces/if:interface"
                       + "[name = current()/../../../interface]/"
                       + "ip:ipv4" {
                       description
                         "Only if IPv4 is enabled on the interface.";
                     }
                     description
                       "IPv4 address family.";
                     leaf transport-address {
                       type union {
                         type enumeration {
                           enum "use-interface-address" {
                             description
                             "Use interface address as the transport
                              address.";
                           }
                         }
                         type inet:ipv4-address;
                       }
                       description
                         "IP address to be advertised as the LDP
                          transport address.";
                     }
                   }

                   container ipv6 {
                     must "/if:interfaces/if:interface"
                       + "[name = current()/../../../interface]/"
                       + "ip:ipv6" {
```

```
                  description
                    "Only if IPv6 is enabled on the interface.";
                }
                description
                  "IPv6 address family.";
                leaf transport-address {
                  type union {
                    type enumeration {
                      enum "use-interface-address" {
                        description
                          "Use interface address as the transport
                           address.";
                      }
                    }
                    type inet:ipv4-address;
                  }
                  description
                    "IP address to be advertised as the LDP
                     transport address.";
                }
              } // ipv6
            }
            container state {
              config false;
              description
                "Operational state data.";
              leaf enable {
                type boolean;
                description
                  "Enable the address family on the interface.";
              }

              container ipv4 {
                must "/if:interfaces/if:interface"
                  + "[name = current()/../../../interface]/"
                  + "ip:ipv4" {
                  description
                    "Only if IPv4 is enabled on the interface.";
                }
                description
                  "IPv4 address family.";
                leaf transport-address {
                  type union {
                    type enumeration {

                      enum "use-interface-address" {
                        description
                          "Use interface address as the transport
```

```
                             address.";
                        }
                      }
                      type inet:ipv4-address;
                    }
                    description
                      "IP address to be advertised as the LDP
                       transport address.";
                  }

                  list hello-adjacencies {
                    key "adjacent-address";
                    description "List of hello adjacencies.";

                    leaf adjacent-address {
                      type inet:ipv4-address;
                      description
                        "Neighbor address of the hello adjacency.";
                    }

                    uses adjacency-state-attributes;

                    leaf peer {
                      type leafref {
                        path "../../../../../../../../../../peers/peer/"
                          + "lsr-id";
                      }
                      description
                        "LDP peer from this adjacency.";
                    }
                  } // hello-adjacencies
                }
                container ipv6 {
                  must "/if:interfaces/if:interface"
                    + "[name = current()/../../../interface]/"
                    + "ip:ipv6" {
                    description
                      "Only if IPv6 is enabled on the interface.";
                  }
                  description
                    "IPv6 address family.";
                  leaf transport-address {
                    type union {
                      type enumeration {
                        enum "use-interface-address" {
                          description
                          "Use interface address as the transport
                           address.";
                        }
                      }
                      type inet:ipv4-address;
                    }
                    description
                      "IP address to be advertised as the LDP
                       transport address.";
                  }

                  list hello-adjacencies {
                    key "adjacent-address";
                    description "List of hello adjacencies.";

                    leaf adjacent-address {
                      type inet:ipv6-address;
                      description
                        "Neighbor address of the hello adjacency.";
                    }

                    uses adjacency-state-attributes;

                    leaf peer {
                      type leafref {
                        path "../../../../../../../../../../peers/peer/"
                          + "lsr-id";
                      }
                      description
                        "LDP peer from this adjacency.";
                    }
                  } // hello-adjacencies
                } // ipv6
              }
            } // address-family
          } // list interface
        } // interfaces

        container targeted
        {
          description
            "A list of targeted neighbors for extended discovery.";
          container config {

            description
              "Configuration data.";
            uses extended-discovery-timers;
            uses extended-discovery-policy-attributes;
          }
          container state {
            config false;
            description
              "Operational state data.";
            uses extended-discovery-timers;
            uses extended-discovery-policy-attributes;
          }

          list address-family {
            key "afi";
            description
              "Per-af params.";
            leaf afi {
              type ldp-address-family;
              description
                "Address family type value.";
            }

            container state {
              config false;
              description
                "Operational state data.";

              container ipv4 {
                when "../../afi = 'ipv4'" {
                  description
                    "For IPv4.";
                }
                description
                  "IPv4 address family.";
                list hello-adjacencies {
                  key "local-address adjacent-address";
                  description "List of hello adjacencies.";

                  leaf local-address {
                    type inet:ipv4-address;
                    description
                      "Local address of the hello adjacency.";
                  }
                  leaf adjacent-address {
                    type inet:ipv4-address;
                    description
                      "Neighbor address of the hello adjacency.";
                  }

                  uses adjacency-state-attributes;

                  leaf peer {
                    type leafref {
```

```
                   path "../../../../../../../../peers/peer/"
                     + "lsr-id";
                 }
                 description
                   "LDP peer from this adjacency.";
               }
             } // hello-adjacencies
           } // ipv4

           container ipv6 {
             when "../../afi = 'ipv6'" {
               description
                 "For IPv6.";
             }
             description
               "IPv6 address family.";
             list hello-adjacencies {
               key "local-address adjacent-address";
               description "List of hello adjacencies.";

               leaf local-address {
                 type inet:ipv6-address;
                 description
                   "Local address of the hello adjacency.";
               }
               leaf adjacent-address {
                 type inet:ipv6-address;
                 description
                   "Neighbor address of the hello adjacency.";
               }

               uses adjacency-state-attributes;

               leaf peer {
                 type leafref {
                   path "../../../../../../../../peers/peer/"
                     + "lsr-id";
                 }
                 description
                   "LDP peer from this adjacency.";
               }
             } // hello-adjacencies
           } // ipv6
         } // state

         container ipv4 {
           when "../afi = 'ipv4'" {
             description
```

```
                    "For IPv4.";
                }
                description
                  "IPv4 address family.";
                list target {
                  key "adjacent-address";
                  description
                    "Targeted discovery params.";

                  leaf adjacent-address {
                    type inet:ipv4-address;
                    description
                      "Configures a remote LDP neighbor and enables
                       extended LDP discovery of the specified
                       neighbor.";
                  }
                  container config {
                    description
                      "Configuration data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv4-address;
                      description
                        "The local address.";
                    }
                  }
                  container state {
                    config false;
                    description
                      "Operational state data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv4-address;
                      description
                        "The local address.";
                    }
                  } // state
                }
              } // ipv4
              container ipv6 {
                when "../afi = 'ipv6'" {
                  description
                    "For IPv6.";
                }
                description
                  "IPv6 address family.";
                list target {
                  key "adjacent-address";
                  description
                    "Targeted discovery params.";

                  leaf adjacent-address {
                    type inet:ipv6-address;
                    description
                      "Configures a remote LDP neighbor and enables
                       extended LDP discovery of the specified
                       neighbor.";
                  }
                  container config {
                    description
                      "Configuration data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv6-address;
                      description
                        "The local address.";
                    }
                  }
                  container state {
                    config false;
                    description
                      "Operational state data.";
                    leaf enable {
                      type boolean;
                      description
                        "Enable the target.";
                    }
                    leaf local-address {
                      type inet:ipv6-address;
                      description
                        "The local address.";
                    }
                  } // state
                }
```

```
              } // ipv6
            } // address-family
          } // targeted
        } // discovery

        container forwarding-nexthop {
          if-feature forwarding-nexthop-config;
          description
            "Configuration for forwarding nexthop.";

          container interfaces {
            description
              "A list of interfaces on which forwarding is
               disabled.";

            list interface {
              key "interface";
              description
                "List of LDP interfaces.";
              leaf interface {
                type mpls-interface-ref;
                description
                  "Interface.";
              }
              list address-family {
                key "afi";
                description
                  "Per-vrf per-af params.";
                leaf afi {
                  type ldp-address-family;
                  description
                    "Address family type value.";
                }
                container config {
                  description
                    "Configuration data.";
                  leaf ldp-disable {
                    type boolean;
                    description
                      "Disable LDP forwarding on the interface.";
                  }
                  leaf mldp-disable {
                    if-feature mldp;
                    type boolean;
                    description
                      "Disable mLDP forwarding on the interface.";
                  }
                }
```

```
                 container state {
                   config false;
                   description
                     "Operational state data.";
                   leaf ldp-disable {
                     type boolean;
                     description
                       "Disable LDP forwarding on the interface.";
                   }
                   leaf mldp-disable {
                     if-feature mldp;

                     type boolean;
                     description
                       "Disable mLDP forwarding on the interface.";
                   }
                 }
               } // address-family
             } // list interface
           } // interfaces
         } // forwarding-nexthop
         uses policy-container {
           if-feature all-af-policy-config;
         }
       } // global

       container peers {
         description
           "Peers configuration attributes.";

         container config {
           description
             "Configuration data.";
           uses peer-authentication {
             if-feature global-session-authentication;
           }
           uses peer-attributes;

           container session-downstream-on-demand {
             if-feature session-downstream-on-demand-config;
             description
               "Session downstream-on-demand attributes.";
             leaf enable {
               type boolean;
               description
                 "'true' if session downstream-on-demand is enabled.";
             }
             leaf peer-list {
               type peer-list-ref;
               description
                 "The name of a peer ACL.";
             }
           }
         }
         container state {
           config false;
           description
             "Operational state data.";
           uses peer-authentication {
             if-feature global-session-authentication;
           }
           uses peer-attributes;

           container session-downstream-on-demand {
             if-feature session-downstream-on-demand-config;
             description
               "Session downstream-on-demand attributes.";
             leaf enable {
               type boolean;
               description
                 "'true' if session downstream-on-demand is enabled.";
             }
             leaf peer-list {
               type peer-list-ref;
               description
                 "The name of a peer ACL.";
             }
           }
         }

         list peer {
           key "lsr-id";
           description
             "List of peers.";

           leaf lsr-id {
             type yang:dotted-quad;
             description "LSR ID.";
           }

           container config {
             description
               "Configuration data.";
             leaf admin-down {
               type boolean;
               default false;
```

```
               description
                 "’true’ to disable the peer.";
             }

             container capability {
               description
                 "Per peer capability";
               container mldp {
                 if-feature mldp;
                 description
                   "mLDP capabilities.";
                 uses mldp-capabilities;
               }
             }

             uses peer-af-policy-container {
               if-feature all-af-policy-config;
             }

             uses peer-authentication;

             uses graceful-restart-attributes-per-peer {
               if-feature per-peer-graceful-restart-config;
             }

             uses peer-attributes {
               if-feature per-peer-session-attributes-config;
             }

             container address-family {
               description
                 "Per-vrf per-af params.";
               container ipv4 {
                 description
                   "IPv4 address family.";
                 uses peer-af-policy-container;
               }
               container ipv6 {
                 description
                   "IPv6 address family.";
                 uses peer-af-policy-container;
               } // ipv6
             } // address-family
           }
           container state {
             config false;
             description
               "Operational state data.";
             leaf admin-down {
               type boolean;
               default false;
               description
                 "’true’ to disable the peer.";
             }

             container capability {
               description
                 "Per peer capability";
               container mldp {
                 if-feature mldp;
                 description
                   "mLDP capabilities.";
                 uses mldp-capabilities;
               }
             }

             uses peer-af-policy-container {
               if-feature all-af-policy-config;
             }

             uses peer-authentication;

             uses graceful-restart-attributes-per-peer {
               if-feature per-peer-graceful-restart-config;
             }

             uses peer-attributes {
               if-feature per-peer-session-attributes-config;
             }

             container address-family {
               description
                 "Per-vrf per-af params.";
               container ipv4 {
                 description
                   "IPv4 address family.";
                 uses peer-af-policy-container;

                 list hello-adjacencies {
                   key "local-address adjacent-address";
                   description "List of hello adjacencies.";

                   leaf local-address {
                     type inet:ipv4-address;
                     description
                       "Local address of the hello adjacency.";
                   }
                   leaf adjacent-address {
                     type inet:ipv4-address;
                     description
                       "Neighbor address of the hello adjacency.";
                   }

                   uses adjacency-state-attributes;

                   leaf interface {
                     type mpls-interface-ref;
                     description "Interface for this adjacency.";
                   }
                 } // hello-adjacencies
               } // ipv4
               container ipv6 {
                 description
                   "IPv6 address family.";
                 uses peer-af-policy-container;

                 list hello-adjacencies {
                   key "local-address adjacent-address";
                   description "List of hello adjacencies.";

                   leaf local-address {
                     type inet:ipv6-address;
                     description
                       "Local address of the hello adjacency.";
                   }
                   leaf adjacent-address {
                     type inet:ipv6-address;
                     description
                       "Neighbor address of the hello adjacency.";
                   }

                   uses adjacency-state-attributes;

                   leaf interface {
                     type mpls-interface-ref;
                     description "Interface for this adjacency.";
                   }
                 } // hello-adjacencies
               } // ipv6
             } // address-family

             uses peer-state-derived;
           } // state
         } // list peer
```

```
        } // peers
      } // container mpls-ldp
    }

    /*
     * RPCs
     */
    rpc mpls-ldp-clear-peer {
      description
        "Clears the session to the peer.";
      input {
        leaf lsr-id {
          type union {
            type yang:dotted-quad;
            type uint32;
          }
          description
            "LSR ID of peer to be cleared. If this is not provided
             then all peers are cleared";
        }
      }
    }

    rpc mpls-ldp-clear-hello-adjacency {
      description
        "Clears the hello adjacency";
      input {
        container hello-adjacency {
          description
            "Link adjacency or targettted adjacency. If this is not
             provided then all hello adjacencies are cleared";
          choice hello-adjacency-type {
            description "Adjacency type.";
            case targeted {
              container targeted {
                presence "Present to clear targeted adjacencies.";
                description
                  "Clear targeted adjacencies.";
                leaf target-address {
                  type inet:ip-address;
                  description
                    "The target address. If this is not provided then
                     all targeted adjacencies are cleared";
                }
              } // targeted
            }
            case link {
              container link {
                presence "Present to clear link adjacencies.";
                description
                  "Clear link adjacencies.";
                leaf next-hop-interface {
                  type mpls-interface-ref;
                  description

                    "Interface connecting to next-hop. If this is not
                     provided then all link adjacencies are cleared.";
                }
                leaf next-hop-address {
                  type inet:ip-address;
                  must "../next-hop-interface" {
                    description
                      "Applicable when interface is specified.";
                  }
                  description
                    "IP address of next-hop. If this is not provided
                     then adjacencies to all next-hops on the given
                     interface are cleared.";
                } // next-hop-address
              } // link
            }
          }
        }
      }
    }

    rpc mpls-ldp-clear-peer-statistics {
      description
        "Clears protocol statistics (e.g. sent and received
         counters).";
      input {
        leaf lsr-id {
          type union {
            type yang:dotted-quad;
            type uint32;
          }
          description
            "LSR ID of peer whose statistic are to be cleared.
             If this is not provided then all peers statistics are
             cleared";
        }
      }
    }

    /*
     * Notifications
     */
    notification mpls-ldp-peer-event {

      description
        "Notification event for a change of LDP peer operational
         status.";
      leaf event-type {
        type oper-status-event-type;
        description "Event type.";
      }
      uses ldp-peer-ref;
    }

    notification mpls-ldp-hello-adjacency-event {
      description
        "Notification event for a change of LDP adjacency operational
         status.";
      leaf event-type {
        type oper-status-event-type;
        description "Event type.";
      }
      uses ldp-adjacency-ref;
    }

    notification mpls-ldp-fec-event {
      description
        "Notification event for a change of FEC status.";
      leaf event-type {
        type oper-status-event-type;
        description "Event type.";
      }
      uses ldp-fec-event;
    }

    notification mpls-mldp-fec-event {
      description
        "Notification event for a change of FEC status.";
      leaf event-type {
        type oper-status-event-type;
        description "Event type.";
      }
      uses mldp-fec-event;
    }
  }

  <CODE ENDS>
```

Figure 22

7. Security Considerations

The configuration, state, action and notification data defined using YANG data models in this document are likely to be accessed via the protocols such as NETCONF [RFC6241] etc.

Hence, YANG implementations MUST comply with the security requirements specified in section 15 of [RFC6020]. Additionally, NETCONF implementations MUST comply with the security requirements specified in sections 2.2, 2.3 and 9 of [RFC6241] as well as section 3.7 of [RFC6536].

8. IANA Considerations

This document does not extend LDP or mLDP base protocol specifiction and hence there are no IANA considerations.

Note to the RFC Editor: Please remove IANA section before the publication.

9. Acknowledgments

The authors would like to acknowledge Eddie Chami, Nagendra Kumar, Mannan Venkatesan, Pavan Beeram for their contribution to this document. We also acknowledge Ladislav Lhotka for his useful comments as the YANG Doctor.

10. References

10.1. Normative References

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

[I-D.rtgyangdt-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., and D. Bogdanovic, "Network Instance Model", draft-rtgyangdt-rtgwg-ni-model-00 (work in progress), May 2016.

[I-D.saad-mpls-base-yang]
Raza, K., Gandhi, R., Liu, X., Beeram, V., Saad, T., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for MPLS Base", draft-saad-mpls-base-yang-00 (work in progress), May 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3478] Leelanivas, M., Rekhter, Y., and R. Aggarwal, "Graceful Restart Mechanism for Label Distribution Protocol", RFC 3478, DOI 10.17487/RFC3478, February 2003, <http://www.rfc-editor.org/info/rfc3478>.

[RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <http://www.rfc-editor.org/info/rfc5036>.

[RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream Label Assignment and Context-Specific Label Space", RFC 5331, DOI 10.17487/RFC5331, August 2008, <http://www.rfc-editor.org/info/rfc5331>.

[RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, DOI 10.17487/RFC5561, July 2009, <http://www.rfc-editor.org/info/rfc5561>.

[RFC5918] Asati, R., Minei, I., and B. Thomas, "Label Distribution Protocol (LDP) ’Typed Wildcard’ Forward Equivalence Class (FEC)", RFC 5918, DOI 10.17487/RFC5918, August 2010, <http://www.rfc-editor.org/info/rfc5918>.

[RFC5919] Asati, R., Mohapatra, P., Chen, E., and B. Thomas, "Signaling LDP Label Advertisement Completion", RFC 5919, DOI 10.17487/RFC5919, August 2010, <http://www.rfc-editor.org/info/rfc5919>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <http://www.rfc-editor.org/info/rfc6388>.

[RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", RFC 6389, DOI 10.17487/RFC6389, November 2011, <http://www.rfc-editor.org/info/rfc6389>.

[RFC6512] Wijnands, IJ., Rosen, E., Napierala, M., and N. Leymann, "Using Multipoint LDP When the Backbone Has No Route to the Root", RFC 6512, DOI 10.17487/RFC6512, February 2012, <http://www.rfc-editor.org/info/rfc6512>.

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

[RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <http://www.rfc-editor.org/info/rfc6826>.

[RFC7060] Napierala, M., Rosen, E., and IJ. Wijnands, "Using LDP Multipoint Extensions on Targeted LDP Sessions", RFC 7060, DOI 10.17487/RFC7060, November 2013, <http://www.rfc-editor.org/info/rfc7060>.

[RFC7140] Jin, L., Jounay, F., Wijnands, IJ., and N. Leymann, "LDP Extensions for Hub and Spoke Multipoint Label Switched Path", RFC 7140, DOI 10.17487/RFC7140, March 2014, <http://www.rfc-editor.org/info/rfc7140>.

[RFC7246] Wijnands, IJ., Ed., Hitchen, P., Leymann, N., Henderickx, W., Gulko, A., and J. Tantsura, "Multipoint Label Distribution Protocol In-Band Signaling in a Virtual Routing and Forwarding (VRF) Table Context", RFC 7246, DOI 10.17487/RFC7246, June 2014, <http://www.rfc-editor.org/info/rfc7246>.

[RFC7438] Wijnands, IJ., Ed., Rosen, E., Gulko, A., Joorde, U., and J. Tantsura, "Multipoint LDP (mLDP) In-Band Signaling with Wildcards", RFC 7438, DOI 10.17487/RFC7438, January 2015, <http://www.rfc-editor.org/info/rfc7438>.

[RFC7552] Asati, R., Pignataro, C., Raza, K., Manral, V., and R. Papneja, "Updates to LDP for IPv6", RFC 7552, DOI 10.17487/RFC7552, June 2015, <http://www.rfc-editor.org/info/rfc7552>.

[RFC7715] Wijnands, IJ., Ed., Raza, K., Atlas, A., Tantsura, J., and Q. Zhao, "Multipoint LDP (mLDP) Node Protection", RFC 7715, DOI 10.17487/RFC7715, January 2016, <http://www.rfc-editor.org/info/rfc7715>.

10.2. Informative References

[I-D.ietf-rtgwg-policy-model] Shaikh, A., Shakir, R., D’Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-ietf-rtgwg-policy-model-01 (work in progress), April 2016.

[I-D.iwijnand-mpls-mldp-multi-topology] Wijnands, I. and K. Raza, "mLDP Extensions for Multi Topology Routing", draft-iwijnand-mpls-mldp-multi-topology-03 (work in progress), June 2013.

[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.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

[RFC7307] Zhao, Q., Raza, K., Zhou, C., Fang, L., Li, L., and D. King, "LDP Extensions for Multi-Topology", RFC 7307, DOI 10.17487/RFC7307, July 2014, <http://www.rfc-editor.org/info/rfc7307>.

Appendix A. Additional Contributors

Stephane Litkowski
Orange.
Email: stephane.litkowski@orange.com

Reshad Rahman
Cisco Systems Inc.
Email: rrahman@cisco.com

Danial Johari
Cisco Systems Inc.
Email: dajohari@cisco.com

Authors' Addresses

Kamran Raza
Cisco Systems, Inc.
Email: skraza@cisco.com

Rajiv Asati
Cisco Systems, Inc.
Email: rajiva@cisco.com

Sowmya Krishnaswamy
Cisco Systems, Inc.
Email: sowkrish@cisco.com

Xufeng Liu
Ericsson
Email: xliu@kuatrotech.com

Jeff Tantsura
Ericsson
Email: jeff.tantsura@ericsson.com

Santosh Esale
Juniper Networks
Email: sesale@juniper.net

Xia Chen
Huawei Technologies
Email: jescia.chenxia@huawei.com

Loa Andersson
Huawei Technologies
Email: loa@pi.nu

Himanshu Shah
Ciena Corporation
Email: hshah@ciena.com

Matthew Bocci
Alcatel-Lucent
Email: matthew.bocci@alcatel-lucent.com