

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

K. Raza
R. Asati
Cisco Systems, Inc.

X. Liu
Jabil

S. Esale
Juniper Networks

X. Chen
Huawei Technologies

H. Shah
Ciena Corporation

March 5, 2018

YANG Data Model for MPLS LDP
draft-ietf-mpls-ldp-yang-04

Abstract

This document describes a YANG data model for Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP). This model also serves as the base model that is augmented to define Multipoint LDP (mLDP) model.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Base and Extended	3
2.	Specification of Requirements	4
3.	Overview	4
4.	Consolidated Tree	7
5.	Configuration	16
5.1.	Configuration Tree	19
5.1.1.	Base	19
5.1.2.	Extended	20
5.2.	Configuration Hierarchy	23
5.2.1.	Per-VRF parameters	23
6.	Operational State	25
6.1.	Operational Tree	25
6.1.1.	Base	25
6.1.2.	Extended	29
6.2.	States	31
6.2.1.	Adjacency state	31
6.2.2.	Peer state	32
6.2.3.	Bindings state	33
6.2.4.	Capabilities state	35
7.	Notifications	35
8.	Actions	36
9.	YANG Specification	37
9.1.	Base	37
9.2.	Extended	65
10.	Security Considerations	85
11.	IANA Considerations	85
12.	Acknowledgments	85
13.	References	85
13.1.	Normative References	85
13.2.	Informative References	87

Appendix A. Additional Contributors	88
Authors' Addresses	88

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](#)]. This model 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 in the sequence as listed above.

1.1. Base and Extended

The configuration and state items are divided into following two broad categories:

- o Base
- o Extended

The "base" category contains the basic and fundamental features that are covered in LDP base specification [[RFC5036](#)] and constitute the minimum requirements for a typical base LDP deployment. Whereas, the "extended" category contains all other non-base features. All the items in a base category are mandatory and hence no "if-feature" is allowed under the "base" category model. The base and extended categories are defined in their own modules as described later.

The example of base feature includes the configuration of LDP lsr-id, enabling LDP interfaces, setting password for LDP session etc.,

whereas the examples of extended feature include inbound/outbound label policies, igp sync, downstream-on-demand etc. This is worth highlighting that LDP IPv6 [[RFC7552](#)] is also categorized as an extended feature.

While "base" model support will suffice for small deployments, it is expected that large deployments will require not only the "base" module support from the vendors but also the support for "extended" model for some extended feature(s) of interest.

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. Overview

This document defines two new modules for LDP YANG support:

- o "ietf-mpls-ldp" module that models the base LDP features and augments /rt:routing/rt:control-plane-protocols defined in [[RFC8022](#)]
- o extended "ietf-mpls-ldp-extended" module that models the extended LDP features and augments the base LDP

It is to be noted that mLDP data model [[I-D.ietf-mpls-mldp-yang](#)] augments LDP base and extended models to model the base and extended mLDP features respectively.

There are four main containers in our module(s):

- o Read-Write parameters for configuration (Discussed in [Section 5](#))
- o Read-only parameters for operational state (Discussed in [Section 6](#))
- o Notifications for events (Discussed in [Section 7](#))
- o RPCs for executing commands to perform some action (Discussed in [Section 8](#))

The modeling in this document complies with the Network Management Datastore Architecture (NMDA) [[I-D.ietf-netmod-revised-datastores](#)]. The operational state data is combined with the associated configuration data in the same hierarchy [[I-D.ietf-netmod-rfc6087bis](#)]. When protocol states are retrieved from the NMDA operational state datastore, the returned states cover all "config true" (rw) and "config false" (ro) nodes defined in the schema.

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

```

+-- rw routing
  +-- rw control-plane-protocols
    +-- rw mpls-ldp
      +-- rw ...
        +-- rw ... // base
        | +-- rw ...
        | +-- ro ...
        | +--
        +-- ro ...
        | +-- ro ...
        | +-- ro ...
        | +--
        +-- rw ldp-ext: .... // extended
        | +-- rw ...
        | +-- ro ...
        | +--
        +-- ro ...
          +-- ro ...
          +-- ro ...

```

rpcs:

```

+-- x mpls-ldp-some_action
+-- x . . . . .

```

notifications:

```

+--- n mpls-ldp-some_event
+--- 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 parameters as per RFC specification, as well as some widely 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](#)] is beyond the scope of this document.
- o This module does not cover any applications running on top of LDP, nor does it cover any OAM procedures for LDP.
- 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.ietf-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 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 belonging to a "stale" peer -- i.e. keeping peer bindings from a peer with whom currently there is either no connection established or connection is established but GR session is in recovery state. When used in this document, the above terms will refer strictly to the semantics and definitions defined for them.

A graphical tree representation of full LDP YANG data model is presented in Figure 2, whereas LDP configuration (base and extended), state (base and extended), notification, and rpc are graphically represented in Figure 5, Figure 6, Figure 8, Figure 9, Figure 15, and Figure 16 respectively. The actual base and extended model definition in YANG is captured in [Section 9](#).

While presenting the YANG tree view and actual .yang specification, this document assumes readers' familiarity with the concepts of YANG modeling, its presentation and its compilation.

[4. Consolidated Tree](#)

Following is a consolidated tree representation of configuration, state, notification, and rpc items under LDP base and extended.

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
    +--rw global
      | +--rw capability
      | | +--rw ldp-ext:end-of-lib {capability-end-of-lib}?
      | | | +--rw ldp-ext:enable? boolean
      | | +--rw ldp-ext:typed-wildcard-fec
      | | | {capability-typed-wildcard-fec}?
      | | | +--rw ldp-ext:enable? boolean
      | | +--rw ldp-ext:upstream-label-assignment
      | | | {capability-upstream-label-assignment}?
      | | +--rw ldp-ext:enable? boolean
      | +--rw graceful-restart
      | | +--rw enable? boolean
      | | +--rw reconnect-time? uint16
      | | +--rw recovery-time? uint16
      | | +--rw forwarding-holdtime? uint16
      | | +--rw ldp-ext:helper-enable? boolean
      | | | {graceful-restart-helper-mode}?
      | +--rw lsr-id? rt-types:router-id
      | +--rw address-families
```



```

| | +--rw ipv4!
| | | +--rw enable? boolean
| | | +--ro label-distribution-controlmode? enumeration
| | | +--ro bindings
| | | | +--ro address* [address]
| | | | | +--ro address inet:ipv4-address
| | | | | +--ro advertisement-type? advertised-received
| | | | | +--ro peer
| | | | | | +--ro lsr-id? leafref
| | | | | | +--ro label-space-id? leafref
| | | | +--ro fec-label* [fec]
| | | | | +--ro fec inet:ipv4-prefix
| | | | | +--ro peer*
| | | | | [lsr-id label-space-id advertisement-type]
| | | | | +--ro lsr-id leafref
| | | | | +--ro label-space-id leafref
| | | | | +--ro advertisement-type advertised-received
| | | | | +--ro label? rt-types:mpls-label
| | | | | +--ro used-in-forwarding? boolean
| | | +--rw ldp-ext:label-policy
| | | | +--rw ldp-ext:advertise
| | | | | +--rw ldp-ext:egress-explicit-null
| | | | | | +--rw ldp-ext:enable? boolean
| | | | | | +--rw ldp-ext:prefix-list?
| | | | | | | prefix-list-ref
| | | | +--rw ldp-ext:accept
| | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | +--rw ldp-ext:assign
| | | | | {policy-label-assignment-config}?
| | | | | +--rw ldp-ext:independent-mode
| | | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | | +--rw ldp-ext:ordered-mode
| | | | | | {policy-ordered-label-config}?
| | | | | | +--rw ldp-ext:egress-prefix-list?
| | | | | | | prefix-list-ref
| | | | +--rw ldp-ext:transport-address? inet:ipv4-address
| | +--rw ldp-ext:ipv6!
| | | +--rw ldp-ext:enable? boolean
| | | +--rw ldp-ext:label-policy
| | | | +--rw ldp-ext:advertise
| | | | | +--rw ldp-ext:egress-explicit-null
| | | | | | +--rw ldp-ext:enable? boolean
| | | | | | +--rw ldp-ext:prefix-list?
| | | | | | | prefix-list-ref
| | | | +--rw ldp-ext:accept
| | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | +--rw ldp-ext:assign
| | | | | {policy-label-assignment-config}?

```



```

| | | +--rw ldp-ext:independent-mode
| | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | +--rw ldp-ext:ordered-mode
| | | | {policy-ordered-label-config}?
| | | | +--rw ldp-ext:egress-prefix-list?
| | | | prefix-list-ref
| | +--rw ldp-ext:transport-address
| | | inet:ipv6-address
| | +--ro ldp-ext:label-distribution-controlmode?
| | | enumeration
| | +--ro ldp-ext:bindings
| | | +--ro ldp-ext:address* [address]
| | | | +--ro ldp-ext:address
| | | | | inet:ipv6-address
| | | | +--ro ldp-ext:advertisement-type?
| | | | | advertised-received
| | | | +--ro ldp-ext:peer
| | | | | +--ro ldp-ext:lsr-id? leafref
| | | | | +--ro ldp-ext:label-space-id? leafref
| | | +--ro ldp-ext:fec-label* [fec]
| | | | +--ro ldp-ext:fec inet:ipv6-prefix
| | | | +--ro ldp-ext:peer*
| | | | | [lsr-id label-space-id advertisement-type]
| | | | +--ro ldp-ext:lsr-id leafref
| | | | +--ro ldp-ext:label-space-id leafref
| | | | +--ro ldp-ext:advertisement-type
| | | | | advertised-received
| | | | +--ro ldp-ext:label?
| | | | | rt-types:mpls-label
| | | | +--ro ldp-ext:used-in-forwarding? boolean
| +--rw ldp-ext:forwarding-nexthop
| | {forwarding-nexthop-config}?
| | +--rw ldp-ext:interfaces
| | | +--rw ldp-ext:interface* [name]
| | | | +--rw ldp-ext:name if:interface-ref
| | | | +--rw ldp-ext:address-family* [afi]
| | | | | +--rw ldp-ext:afi ldp:ldp-address-family
| | | | | +--rw ldp-ext:ldp-disable? boolean
| | +--rw ldp-ext:igp-synchronization-delay? uint16
+--rw discovery
| +--rw interfaces
| | +--rw hello-holdtime? uint16
| | +--rw hello-interval? uint16
| | +--rw interface* [name]
| | | +--rw name
| | | | if:interface-ref
| | | +--ro next-hello? uint16
| | +--rw address-families

```



```

| | | +--rw ipv4!
| | | | +--rw enable? boolean
| | | | +--ro hello-adjacencies
| | | | | +--ro hello-adjacency* [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
| | | | | +--ro lsr-id? leafref
| | | | | +--ro label-space-id? leafref
| | | | +--rw ldp-ext:transport-address? union
| | | +--rw ldp-ext:ipv6!
| | | | +--rw ldp-ext:enable? boolean
| | | | +--ro ldp-ext:hello-adjacencies
| | | | | +--ro ldp-ext:hello-adjacency*
| | | | | | [adjacent-address]
| | | | | | +--ro ldp-ext:adjacent-address
| | | | | | | inet:ipv6-address
| | | | | | +--ro ldp-ext:flag* identityref
| | | | | +--ro ldp-ext:hello-holdtime
| | | | | | +--ro ldp-ext:adjacent? uint16
| | | | | | +--ro ldp-ext:negotiated? uint16
| | | | | | +--ro ldp-ext:remaining? uint16
| | | | | +--ro ldp-ext:next-hello? uint16
| | | | +--ro ldp-ext:statistics
| | | | | +--ro ldp-ext:discontinuity-time
| | | | | | yang:date-and-time
| | | | | +--ro ldp-ext:hello-received?
| | | | | | yang:counter64
| | | | | +--ro ldp-ext:hello-dropped?
| | | | | | yang:counter64
| | | | +--ro ldp-ext:peer
| | | | | +--ro ldp-ext:lsr-id? leafref
| | | | | +--ro ldp-ext:label-space-id? leafref
| | | | +--rw ldp-ext:transport-address? union
| | | +--rw ldp-ext:hello-holdtime? uint16
| | | +--rw ldp-ext:hello-interval? uint16
| | | +--rw ldp-ext:igp-synchronization-delay? uint16
| | | {per-interface-timer-config}?

```



```

|   +--rw targeted
|   |   +--rw hello-holdtime?      uint16
|   |   +--rw hello-interval?     uint16
|   |   +--rw hello-accept
|   |   |   +--rw enable?          boolean
|   |   |   +--rw ldp-ext:neighbor-list? neighbor-list-ref
|   |   |       {policy-targeted-discovery-config}?
|   +--rw address-families
|   |   +--rw ipv4!
|   |   |   +--ro hello-adjacencies
|   |   |   |   +--ro hello-adjacency*
|   |   |   |   |       [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
|   |   |   |   |       |   +--ro lsr-id?         leafref
|   |   |   |   |       |   +--ro label-space-id? leafref
|   |   |   +--rw target* [adjacent-address]
|   |   |   |   +--rw adjacent-address   inet:ipv4-address
|   |   |   |   +--rw enable?           boolean
|   |   |   |   +--rw local-address?     inet:ipv4-address
|   +--rw ldp-ext:ipv6!
|   |   +--ro ldp-ext:hello-adjacencies
|   |   |   +--ro ldp-ext:hello-adjacency*
|   |   |   |       [local-address adjacent-address]
|   |   |   |       +--ro ldp-ext:local-address
|   |   |   |       |       inet:ipv6-address
|   |   |   |       +--ro ldp-ext:adjacent-address
|   |   |   |       |       inet:ipv6-address
|   |   |   |       +--ro ldp-ext:flag*          identityref
|   |   |   |       +--ro ldp-ext:hello-holdtime
|   |   |   |       |   +--ro ldp-ext:adjacent?    uint16
|   |   |   |       |   +--ro ldp-ext:negotiated?  uint16
|   |   |   |       |   +--ro ldp-ext:remaining?   uint16
|   |   |   |       +--ro ldp-ext:next-hello?      uint16
|   |   |   |       +--ro ldp-ext:statistics
|   |   |   |       |   +--ro ldp-ext:discontinuity-time

```



```

| | | yang:date-and-time
| | | +--ro ldp-ext:hello-received?
| | | | yang:counter64
| | | +--ro ldp-ext:hello-dropped?
| | | | yang:counter64
| | | +--ro ldp-ext:peer
| | | | +--ro ldp-ext:lsr-id? leafref
| | | | +--ro ldp-ext:label-space-id? leafref
| | | +--rw ldp-ext:target* [adjacent-address]
| | | | +--rw ldp-ext:adjacent-address inet:ipv6-address
| | | | +--rw ldp-ext:enable? boolean
| | | | +--rw ldp-ext:local-address? inet:ipv6-address
+--rw peers
+--rw authentication
| +--rw (auth-type-selection)?
| | +--:(auth-key)
| | | +--rw md5-key? string
| | | +--:(ldp-ext:auth-key-chain) {key-chain}?
| | | +--rw ldp-ext:key-chain? key-chain:key-chain-ref
+--rw session-ka-holdtime? uint16
+--rw session-ka-interval? uint16
+--rw peer* [lsr-id label-space-id]
| +--rw lsr-id rt-types:router-id
| +--rw label-space-id uint16
| +--rw authentication
| | +--rw (auth-type-selection)?
| | | +--:(auth-key)
| | | | +--rw md5-key? string
| | | | +--:(ldp-ext:auth-key-chain) {key-chain}?
| | | | +--rw ldp-ext:key-chain? key-chain:key-chain-ref
+--rw capability
+--rw address-families
| | +--rw ipv4!
| | | +--ro hello-adjacencies
| | | | +--ro hello-adjacency*
| | | | | [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? if:interface-ref
| | | +-rw ldp-ext:label-policy
| | | +-rw ldp-ext:advertise
| | | | +-rw ldp-ext:prefix-list? prefix-list-ref
| | | +-rw ldp-ext:accept
| | | +-rw ldp-ext:prefix-list? prefix-list-ref
| | +-rw ldp-ext:ipv6!
| | +-ro ldp-ext:hello-adjacencies
| | | +-ro ldp-ext:hello-adjacency*
| | | | [local-address adjacent-address]
| | | +-ro ldp-ext:local-address
| | | | inet:ipv6-address
| | | +-ro ldp-ext:adjacent-address
| | | | inet:ipv6-address
| | | +-ro ldp-ext:flag* identityref
| | | +-ro ldp-ext:hello-holdtime
| | | | +-ro ldp-ext:adjacent? uint16
| | | | +-ro ldp-ext:negotiated? uint16
| | | | +-ro ldp-ext:remaining? uint16
| | | +-ro ldp-ext:next-hello? uint16
| | | +-ro ldp-ext:statistics
| | | | +-ro ldp-ext:discontinuity-time
| | | | | yang:date-and-time
| | | | +-ro ldp-ext:hello-received?
| | | | | yang:counter64
| | | | +-ro ldp-ext:hello-dropped?
| | | | | yang:counter64
| | | +-ro ldp-ext:interface?
| | | | if:interface-ref
| | +-rw ldp-ext:label-policy
| | +-rw ldp-ext:advertise
| | | +-rw ldp-ext:prefix-list? prefix-list-ref
| | +-rw ldp-ext:accept
| | | +-rw ldp-ext:prefix-list? prefix-list-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 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
| +--rw ldp-ext:admin-down?    boolean
| |      {per-peer-admin-down}?

```



```

    | +--rw ldp-ext:graceful-restart
    | | +--rw ldp-ext:enable?          boolean
    | | +--rw ldp-ext:reconnect-time?  uint16
    | | +--rw ldp-ext:recovery-time?   uint16
    | +--rw ldp-ext:session-ka-holdtime? uint16
    | +--rw ldp-ext:session-ka-interval? uint16
+--rw ldp-ext:session-downstream-on-demand
|   {session-downstream-on-demand-config}?
| +--rw ldp-ext:enable?          boolean
| +--rw ldp-ext:peer-list?       peer-list-ref
+--rw ldp-ext:dual-stack-transport-pereference {dual-stack-transport-
pereference}?
    +--rw ldp-ext:max-wait?          uint16
    +--rw ldp-ext:prefer-ipv4!
        +--rw ldp-ext:peer-list?     peer-list-ref

```

rpcs:

```

+---x mpls-ldp-clear-peer
| +---w input
|   +---w lsr-id?          leafref
|   +---w label-space-id? leafref
+---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? leafref
|             +---w next-hop-address?   inet:ip-address
+---x mpls-ldp-clear-peer-statistics
    +---w input
        +---w lsr-id?          leafref
        +---w label-space-id? leafref

```

notifications:

```

+---n mpls-ldp-peer-event
| +--ro event-type?  oper-status-event-type
| +--ro peer
|   +--ro lsr-id?          leafref
|   +--ro label-space-id? 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?   if: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 2

5. 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 model augments `/rt:routing/rt:control-plane-protocols` that is defined in [RFC8022] and follows NMDA as mentioned earlier.

Following is the high-level configuration organization for base LDP:


```

augment /rt:routing/rt:control-plane-protocols:
  +-- mpls-ldp
    +-- global
      |   +-- ...
      |   +-- ...
      |   +-- address-families
      |     |   +-- ipv4
      |     |     +-- ...
      |     |     +-- ...
      |   +-- capability
      |     +-- ...
      |     +-- ...
    +-- discovery
      |   +-- interfaces
      |     |   +-- ...
      |     |   +-- ...
      |     |   +-- interface* [interface]
      |     |     +-- ...
      |     |     +-- address-families
      |     |       +-- ipv4
      |     |       +-- ...
      |     |       +-- ...
      |   +-- targetted
      |     +-- ...
      |     +-- address-families
      |       +-- ipv4
      |         +-- target* [adjacent-address]
      |           +- ...
      |           +- ...
    +-- peers
      +-- ...
      +-- ...
      +-- peer* [lsr-id label-space-id]
        +-- ...
        +-- ...

```

Figure 3

Following is the high-level configuration organization for extended LDP:

```

augment /rt:routing/rt:control-plane-protocols:
  +-- mpls-ldp
    +-- global
      |   +-- ...
      |   +-- ...

```



```

|   +-- address-families
|   |   +-- ipv4
|   |   |   +-- . . .
|   |   |   +-- . . .
|   |   |   +-- label-policy
|   |   |       +-- ...
|   |   |       +-- ...
|   |   +-- ipv6
|   |       +-- . . .
|   |       +-- . . .
|   |       +-- label-policy
|   |           +-- ...
|   |           +-- ...
|   +-- capability
|   |   +-- ...
|   |   +-- ...
|   +-- discovery
|   |   +-- interfaces
|   |   |   +-- ...
|   |   |   +-- ...
|   |   |   +-- interface* [interface]
|   |   |       +-- ...
|   |   |       +-- address-families
|   |   |           +-- ipv4
|   |   |               | +-- ...
|   |   |               | +-- ...
|   |   |           +-- ipv6
|   |   |               +-- ...
|   |   |               +-- ...
|   |   +-- targetted
|   |       +-- ...
|   |       +-- address-families
|   |           +-- ipv6
|   |               +- target* [adjacent-address]
|   |                   +- ...
|   |                   +- ...
+-- forwarding-nextthop
|   +-- ...
|   +-- ...
+-- peers
|   +-- ...
|   +-- ...
|   +-- peer*
|       +-- ...
|       +-- ...
|       +-- label-policy
|           | +-- ..
+-- address-families

```



```

+-- ipv4
|   +-- ...
+-- ipv6
    +-- ...

```

Figure 4

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.

5.1. Configuration Tree

5.1.1. Base

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

```

module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
+--rw mpls-ldp!
  +--rw global
  |   +--rw graceful-restart
  |   |   +--rw enable?                boolean
  |   |   +--rw reconnect-time?        uint16
  |   |   +--rw recovery-time?         uint16
  |   |   +--rw forwarding-holdtime?    uint16
  |   +--rw lsr-id?                    rt-types:router-id
  |   +--rw address-families
  |   |   +--rw ipv4!
  |   |   |   +--rw enable?            boolean
  +--rw discovery
  |   +--rw interfaces
  |   |   +--rw hello-holdtime?    uint16
  |   |   +--rw hello-interval?   uint16
  |   |   +--rw interface* [name]
  |   |   |   +--rw name            if:interface-ref
  |   |   |   +--rw address-families
  |   |   |   |   +--rw ipv4!
  |   |   |   |   |   +--rw enable?    boolean
  |   +--rw targeted
  |   |   +--rw hello-holdtime?    uint16

```



```

|   +--rw hello-interval?    uint16
|   +--rw hello-accept
|   |   +--rw enable?    boolean
|   +--rw address-families
|   |   +--rw ipv4!
|   |   +--rw target* [adjacent-address]
|   |       +--rw adjacent-address    inet:ipv4-address
|   |       +--rw enable?            boolean
|   |       +--rw local-address?      inet:ipv4-address
+--rw peers
    +--rw authentication
    |   +--rw (auth-type-selection)?
    |   |   +--:(auth-key)
    |   |   +--rw md5-key?    string
+--rw session-ka-holdtime?    uint16
+--rw session-ka-interval?    uint16
+--rw peer* [lsr-id label-space-id]
    +--rw lsr-id                rt-types:router-id
    +--rw label-space-id        uint16
    +--rw authentication
    |   +--rw (auth-type-selection)?
    |   |   +--:(auth-key)
    |   |   +--rw md5-key?    string

```

Figure 5

5.1.2. Extended

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

```

module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
  +--rw global
  |   +--rw capability
  |   |   +--rw ldp-ext:end-of-lib {capability-end-of-lib}?
  |   |   |   +--rw ldp-ext:enable?    boolean
  |   |   +--rw ldp-ext:typed-wildcard-fec {capability-typed-wildcard-fec}?
  |   |   |   +--rw ldp-ext:enable?    boolean
  |   |   +--rw ldp-ext:upstream-label-assignment {capability-upstream-
label-assignment}?
  |   |       +--rw ldp-ext:enable?    boolean
  |   +--rw graceful-restart
  |   |   +--rw ldp-ext:helper-enable?    boolean {graceful-restart-helper-

```

```
mode}?  
|  +--rw address-families
```

```

| | +--rw ipv4!
| | | +--rw ldp-ext:label-policy
| | | | +--rw ldp-ext:advertise
| | | | | +--rw ldp-ext:egress-explicit-null
| | | | | | +--rw ldp-ext:enable? boolean
| | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | +--rw ldp-ext:accept
| | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | +--rw ldp-ext:assign {policy-label-assignment-config}?
| | | | | +--rw ldp-ext:independent-mode
| | | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | | +--rw ldp-ext:ordered-mode {policy-ordered-label-config}?
| | | | | +--rw ldp-ext:egress-prefix-list? prefix-list-ref
| | | +--rw ldp-ext:transport-address? inet:ipv4-address
| | +--rw ldp-ext:ipv6!
| | | +--rw ldp-ext:enable? boolean
| | | +--rw ldp-ext:label-policy
| | | | +--rw ldp-ext:advertise
| | | | | +--rw ldp-ext:egress-explicit-null
| | | | | | +--rw ldp-ext:enable? boolean
| | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | +--rw ldp-ext:accept
| | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | +--rw ldp-ext:assign {policy-label-assignment-config}?
| | | | | +--rw ldp-ext:independent-mode
| | | | | | +--rw ldp-ext:prefix-list? prefix-list-ref
| | | | | +--rw ldp-ext:ordered-mode {policy-ordered-label-config}?
| | | | | +--rw ldp-ext:egress-prefix-list? prefix-list-ref
| | | +--rw ldp-ext:transport-address inet:ipv6-address
| +--rw ldp-ext:forwarding-nexthop {forwarding-nexthop-config}?
| | +--rw ldp-ext:interfaces
| | | +--rw ldp-ext:interface* [name]
| | | | +--rw ldp-ext:name if:interface-ref
| | | | +--rw ldp-ext:address-family* [afi]
| | | | | +--rw ldp-ext:afi ldp:ldp-address-family
| | | | | +--rw ldp-ext:ldp-disable? boolean
| | +--rw ldp-ext:igp-synchronization-delay? uint16
+--rw discovery
| +--rw interfaces
| | +--rw interface* [name]
| | | +--rw name if:interface-ref
| | | +--rw address-families
| | | | +--rw ipv4!
| | | | | +--rw ldp-ext:transport-address? union
| | | | | +--rw ldp-ext:ipv6!
| | | | | +--rw ldp-ext:enable? boolean
| | | | | +--rw ldp-ext:transport-address? union
| | | +--rw ldp-ext:hello-holdtime? uint16

```



```

    | |      +--rw ldp-ext:hello-interval?          uint16
    | |      +--rw ldp-ext:igp-synchronization-delay? uint16 {per-
interface-timer-config}?
    | +--rw targeted
    |   +--rw hello-accept
    |     | +--rw ldp-ext:neighbor-list?  neighbor-list-ref {policy-
targeted-discovery-config}?
    |     +--rw address-families
    |       +--rw ldp-ext:ipv6!
    |         +--rw ldp-ext:target* [adjacent-address]
    |           +--rw ldp-ext:adjacent-address  inet:ipv6-address
    |           +--rw ldp-ext:enable?           boolean
    |           +--rw ldp-ext:local-address?    inet:ipv6-address
+--rw peers
  +--rw authentication
  | +--rw (auth-type-selection)?
  |   +--:(ldp-ext:auth-key-chain) {key-chain}?
  |     +--rw ldp-ext:key-chain?  key-chain:key-chain-ref
+--rw peer* [lsr-id label-space-id]
  | +--rw lsr-id                    rt-types:router-id
  | +--rw label-space-id            uint16
  | +--rw authentication
  | | +--rw (auth-type-selection)?
  | |   +--:(ldp-ext:auth-key-chain) {key-chain}?
  | |     +--rw ldp-ext:key-chain?  key-chain:key-chain-ref
  | +--rw address-families
  | | +--rw ipv4!
  | | | +--rw ldp-ext:label-policy
  | | |   +--rw ldp-ext:advertise
  | | |     | +--rw ldp-ext:prefix-list?  prefix-list-ref
  | | |     +--rw ldp-ext:accept
  | | |       +--rw ldp-ext:prefix-list?  prefix-list-ref
  | | +--rw ldp-ext:ipv6!
  | |   +--rw ldp-ext:label-policy
  | |     +--rw ldp-ext:advertise
  | |       | +--rw ldp-ext:prefix-list?  prefix-list-ref
  | |       +--rw ldp-ext:accept
  | |         +--rw ldp-ext:prefix-list?  prefix-list-ref
  | +--rw ldp-ext:admin-down?          boolean {per-peer-admin-
down}?
  | +--rw ldp-ext:graceful-restart
  | | +--rw ldp-ext:enable?            boolean
  | | +--rw ldp-ext:reconnect-time?    uint16
  | | +--rw ldp-ext:recovery-time?     uint16
  | +--rw ldp-ext:session-ka-holdtime? uint16
  | +--rw ldp-ext:session-ka-interval? uint16
  +--rw ldp-ext:session-downstream-on-demand {session-downstream-on-
demand-config}?

```

```
|  +-rw ldp-ext:enable?      boolean
|  +-rw ldp-ext:peer-list?   peer-list-ref
+-rw ldp-ext:dual-stack-transport-pereference {dual-stack-transport-
pereference}?
    +-rw ldp-ext:max-wait?    uint16
```

```
+--rw ldp-ext:prefer-ipv4!  
  +--rw ldp-ext:peer-list?  peer-list-ref
```

Figure 6

5.2. 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  
  
  o Forwarding parameters
```

Figure 7

Following subsections briefly explain these configuration areas.

5.2.1. 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.

5.2.1.1. 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.

5.2.1.2. 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.

5.2.1.3. 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.

5.2.1.4. 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.

5.2.1.5. 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.

5.2.1.6. 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.

6. 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.

6.1. Operational Tree

6.1.1. Base

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

```
module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
    +--rw global
      | +--rw address-families
      |   +--rw ipv4!
      |     +--ro label-distribution-controlmode? enumeration
      |     +--ro bindings
      |       +--ro address* [address]
      |         | +--ro address inet:ipv4-address
      |         | +--ro advertisement-type? advertised-received
```



```

|         | +--ro peer
|         |   +--ro lsr-id?          leafref
|         |   +--ro label-space-id?  leafref
|         +--ro fec-label* [fec]
|           +--ro fec      inet:ipv4-prefix
|           +--ro peer*
|             [lsr-id label-space-id advertisement-type]
|             +--ro lsr-id          leafref
|             +--ro label-space-id  leafref
|             +--ro advertisement-type  advertised-received
|             +--ro label?          rt-types:mpls-label
|             +--ro used-in-forwarding? boolean
+--rw discovery
| +--rw interfaces
| | +--rw interface* [name]
| |   +--rw name          if:interface-ref
| |   +--ro next-hello?    uint16
| |   +--rw address-families
| |     +--rw ipv4!
| |       +--ro hello-adjacencies
| |         +--ro hello-adjacency* [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
| |             +--ro lsr-id?          leafref
| |             +--ro label-space-id? leafref
+--rw targeted
| +--rw address-families
| | +--rw ipv4!
| |   +--ro hello-adjacencies
| |     +--ro hello-adjacency*
| |       [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
|         | +--ro lsr-id?          leafref
|         | +--ro label-space-id? leafref
+--rw peers
  +--rw peer* [lsr-id label-space-id]
    +--rw lsr-id          rt-types:router-id
    +--rw label-space-id  uint16
    +--rw address-families
      +--rw ipv4!
        +--ro hello-adjacencies
          +--ro hello-adjacency*
            [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?         if: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 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 8

6.1.2. Extended

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

```

module: ietf-mpls-ldp
augment /rt:routing/rt:control-plane-protocols:
  +--rw mpls-ldp!
    +--rw global
      | +--rw address-families
      |   +--rw ldp-ext:ipv6!
      |     +--ro ldp-ext:label-distribution-controlmode?
      |       | enumeration
      |     +--ro ldp-ext:bindings
      |       +--ro ldp-ext:address* [address]
      |         | +--ro ldp-ext:address
      |           | | inet:ipv6-address
      |           | +--ro ldp-ext:advertisement-type?
      |             | | advertised-received
      |             | +--ro ldp-ext:peer
      |               +--ro ldp-ext:lsr-id? leafref
      |               +--ro ldp-ext:label-space-id? leafref
      |             +--ro ldp-ext:fec-label* [fec]
      |               +--ro ldp-ext:fec inet:ipv6-prefix
      |               +--ro ldp-ext:peer* [lsr-id label-space-id advertisement-
type]
      |                 +--ro ldp-ext:lsr-id leafref
      |                 +--ro ldp-ext:label-space-id leafref
      |                 +--ro ldp-ext:advertisement-type
      |                   | advertised-received
      |                 +--ro ldp-ext:label?
      |                   | rt-types:mpls-label
      |                 +--ro ldp-ext:used-in-forwarding? boolean
    +--rw discovery
      | +--rw interfaces
      |   +--rw interface* [name]
      |     +--rw name if:interface-ref
      |     +--rw address-families
      |       +--rw ldp-ext:ipv6!
      |         +--ro ldp-ext:hello-adjacencies
      |           +--ro ldp-ext:hello-adjacency* [adjacent-address]
      |             +--ro ldp-ext:adjacent-address inet:ipv6-address
      |             +--ro ldp-ext:flag* identityref
      |             +--ro ldp-ext:hello-holdtime
      |               | +--ro ldp-ext:adjacent? uint16
      |               | +--ro ldp-ext:negotiated? uint16
      |               | +--ro ldp-ext:remaining? uint16

```

| |

+--ro ldp-ext:next-hello?

uint16

Raza, et al.

Expires September 6, 2018

[Page 29]

```

| |                                     +--ro ldp-ext:statistics
| |                                     | +--ro ldp-ext:discontinuity-time yang:date-and-
time | |                                     | +--ro ldp-ext:hello-received? yang:counter64
| |                                     | +--ro ldp-ext:hello-dropped? yang:counter64
| |                                     +--ro ldp-ext:peer
| |                                     +--ro ldp-ext:lsr-id?          leafref
| |                                     +--ro ldp-ext:label-space-id?  leafref
| +--rw targeted
|   +--rw address-families
|     +--rw ldp-ext:ipv6!
|       +--ro ldp-ext:hello-adjacencies
|         +--ro ldp-ext:hello-adjacency* [local-address adjacent-
address]
|           +--ro ldp-ext:local-address inet:ipv6-address
|           +--ro ldp-ext:adjacent-address inet:ipv6-address
|           +--ro ldp-ext:flag*          identityref
|           +--ro ldp-ext:hello-holdtime
|           | +--ro ldp-ext:adjacent?    uint16
|           | +--ro ldp-ext:negotiated?  uint16
|           | +--ro ldp-ext:remaining?   uint16
|           +--ro ldp-ext:next-hello?    uint16
|           +--ro ldp-ext:statistics
|           | +--ro ldp-ext:discontinuity-time yang:date-and-time
|           | +--ro ldp-ext:hello-received? yang:counter64
|           | +--ro ldp-ext:hello-dropped? yang:counter64
|           +--ro ldp-ext:peer
|           +--ro ldp-ext:lsr-id?          leafref
|           +--ro ldp-ext:label-space-id?  leafref
+--rw peers
  +--rw address-families
    +--rw ldp-ext:ipv6!
      +--ro ldp-ext:hello-adjacencies
        +--ro ldp-ext:hello-adjacency* [local-address adjacent-
address]
          +--ro ldp-ext:local-address inet:ipv6-address
          +--ro ldp-ext:adjacent-address inet:ipv6-address
          +--ro ldp-ext:flag*          identityref
          +--ro ldp-ext:hello-holdtime
          | +--ro ldp-ext:adjacent?    uint16
          | +--ro ldp-ext:negotiated?  uint16
          | +--ro ldp-ext:remaining?   uint16
          +--ro ldp-ext:next-hello?    uint16
          +--ro ldp-ext:statistics
          | +--ro ldp-ext:discontinuity-time yang:date-and-time
          | +--ro ldp-ext:hello-received? yang:counter64
          | +--ro ldp-ext:hello-dropped? yang:counter64
          +--ro ldp-ext:interface? if:interface-ref

```

Figure 9

Raza, et al.

Expires September 6, 2018

[Page 30]

6.2. States

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

Neighbor Adjacencies

Peer

Bindings (FEC-label and address)

Capabilities

6.2.1. 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-families
      |     +--rw ipv4 (or ipv6)
      |       +--ro hello-adjacencies
      |         +--ro hello-adjacencies* [adjacent-address]
      |           +--ro adjacent-address
      |             . . . .
      |             . . . .
    +--rw targeted
      +--rw address-families
        +--rw ipv4 (or ipv6)
          +--ro hello-adjacencies
            +--ro hello-adjacencies* [local-address adjacent-address]
              +--ro local-address
              +--ro adjacent-address
                . . . .
                . . . .

```

Figure 10

6.2.2. Peer state

Peer related 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 label-space-id]
      +--rw lsr-id
      +--rw label-space-id
      +--ro label-advertisement-mode
      +--ro session-state
      +--ro tcp-connection
      +--ro session-holdtime?
      +--ro up-time
      +-- . . . .
      +--ro address-families
      | +--ro ipv4 (or ipv6)
      |   +--ro hello-adjacencies
      |     +--ro hello-adjacencies* [local-address adjacent-
address]
      |       . . . .
      |       . . . .
      +--ro received-peer-state
      | +--ro . . . .
      | +--ro capability
      |   +--ro . . . .
      +--ro statistics
        +-- . . . .
        +-- received
        | +-- ...
        +-- sent
        | +-- ...

```

Figure 11

6.2.3. 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 203.0.113.1/32:
    advertised: local-label 16000
      peer 192.0.2.1:0
      peer 192.0.2.2:0
      peer 192.0.2.3:0
    received:
      peer 192.0.2.1:0, label 16002, used-in-forwarding=Yes
      peer 192.0.2.2:0, label 17002, used-in-forwarding=No
  FEC 203.0.113.2/32:
    . . . .
  FEC 198.51.100.0/24:
    . . . .

Address bindings:
  Addr 192.0.2.10:
    advertised
  Addr 192.0.2.1:
    received, peer 192.0.2.1:0
  Addr 192.0.2.2:
    received, peer 192.0.2.2:0
  Addr 192.0.2.3:
    received, peer 192.0.2.3:0
```

Figure 12

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-families
      +--rw ipv4 (or ipv6)
        +--ro bindings
          +--ro address* [address]
            | +--ro address (ipv4-address or ipv6-address)
            | +--ro advertisement-type? advertised-received
            | +--ro peer? leafref
          +--ro fec-label* [fec]
            +--ro fec (ipv4-prefix or 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

```

Figure 13

6.2.4. 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 peers
    +--rw peer* [lsr-id label-space-id]
      +--rw lsr-id yang:dotted-quad
      +--rw label-space-id
      +--ro received-peer-state
        +--ro capability
          +--ro . . . .
          +--ro . . . .

```

Figure 14

7. 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
    |   +--ro lsr-id?      leafref
    |   +--ro label-space-id? 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?  if: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 15

8. 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?          leafref
    |       +---w label-space-id?  leafref
    +---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?  leafref
    |                           +---w next-hop-address?    inet:ip-address
    +---x mpls-ldp-clear-peer-statistics
    |   +---w input
    |       +---w lsr-id?          leafref
    |       +---w label-space-id?  leafref

```

Figure 16

9. YANG Specification

Following are the actual YANG definition (module) for LDP constructs defined earlier in the document.

9.1. Base

```

<CODE BEGINS> file "ietf-mpls-ldp@2018-02-28.yang"

module ietf-mpls-ldp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp";
  prefix "ldp";

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

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

```



```
import ietf-routing {
  prefix "rt";
}

import ietf-routing-types {
  prefix "rt-types";
}

import ietf-interfaces {
  prefix "if";
}

import ietf-ip {
  prefix "ip";
}

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

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

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

  Editor:  Xufeng Liu
           <mailto:Xufeng_Liu@jabril.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). It is also the base model to
   be augmented for Multipoint LDP (mLDP).

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.
```


Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in [Section 4.c](#) of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

```
revision 2018-02-28 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Data Model for MPLS LDP.";
}

/*
 * 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 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.";
}
```



```
/*
 * 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
    "The operational state attributes of an LDP hello adjacency,
    which can used for basic and extended discoveris, in IPv4 and
    IPv6 address families.";

  leaf-list flag {
    type identityref {
      base adjacency-flag-base;
    }
    description
      "On or more flags to indicate whether the adjacency is
      actively created, passively accepted, or both.";
  }

  container hello-holdtime {
    description
      "Containing hello holdtime state information.";
    leaf adjacent {
      type uint16;
      units seconds;
      description
        "The holdtime value learned from the adjacent LSR.";
    }
    leaf negotiated {
      type uint16;
    }
  }
}
```



```
        units seconds;
        description
            "The holdtime negotiated between this LSR and the adjacent
             LSR.";
    }
    leaf remaining {
        type uint16;
        units seconds;
        description
            "The time remaining until the holdtime timer expires.";
    }
}

leaf next-hello {
    type uint16;
    units seconds;
    description
        "The time when the next Hello message will be sent.";
}

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 dropped.";
    }
} // statistics
} // adjacency-state-attributes
```



```
grouping basic-discovery-timers {
  description
    "The timer attributes for basic discovery, used in the
    per-interface setting and in the all-interface setting.";

  leaf hello-holdtime {
    type uint16 {
      range 15..3600;
    }
    units seconds;
    default 15;
    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;
    default 5;
    description
      "The interval between consecutive LDP link Hello messages
      used in basic LDP discovery";
  }
} // basic-discovery-timers

grouping binding-address-state-attributes {
  description
    "Operational state attributes of an address binding, used in
    IPv4 and IPv6 address families.";

  leaf advertisement-type {
    type advertised-received;
    description
      "Received or advertised.";
  }
  container peer {
    when "../advertisement-type = 'received'" {
      description
        "Applicable for received address.";
    }
    description
      "LDP peer from which this address is received.";
    uses ldp-peer-ref;
  }
} // binding-address-state-attributes
```



```
grouping binding-label-state-attributes {
  description
    "Operational state attributes for a FEC-label binding, used in
    IPv4 and IPv6 address families.";

  list peer {
    key "lsr-id label-space-id advertisement-type";
    description
      "List of advertised and received peers.";
    uses ldp-peer-ref {
      description
        "The LDP peer from which this binding is received, or to
        which this binding is advertised.
        The peer is identified by its LDP ID, which consists of
        the LSR ID and the Label Space ID.";
    }
    leaf advertisement-type {
      type advertised-received;
      description
        "Received or advertised.";
    }
    leaf label {
      type rt-types: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 graceful-restart-attributes-per-peer {
  description
    "Per peer graceful restart attributes.
    On the local side, these attributes are configuration and
    operational state data. One the peer side, these attributes
    are operational state data reveiced from the peer.";

  container graceful-restart {
    description
      "Attributes for graceful restart.";
    leaf enable {
      type boolean;
      default false;
    }
  }
}
```



```
        description
            "Enable or disable graceful restart.";
    }
    leaf reconnect-time {
        type uint16 {
            range 10..1800;
        }
        units seconds;
        default 120;
        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;
        default 120;
        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 ldp-interface-ref {
    description
        "Defining a reference to LDP interface.";

    leaf name {
        type if:interface-ref;
        must "(/if:interfaces/if:interface[if:name=current()]/ip:ipv4)"
            + " or "
            + "(/if:interfaces/if:interface[if:name=current()]/ip:ipv6)"
        {
            description "Interface is IPv4 or IPv6.";
        }
        description
            "The name of an LDP interface.";
    }
}

grouping ldp-peer-ref {
    description
```



```
"An absolute reference to an LDP peer, by the LDP ID, which
consists of the LSR ID and the Label Space ID.";

leaf lsr-id {
  type leafref {
    path "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
      + "ldp:peers/ldp:peer/ldp:lsr-id";
  }
  description
    "The LSR ID of the peer, as a portion of the peer LDP ID.";
}
leaf label-space-id {
  type leafref {
    path "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
      + "ldp:peers/ldp:peer/ldp:label-space-id";
  }
  description
    "The Label Space ID of the peer, as a portion of the peer
    LDP ID.";
}
} // ldp-peer-ref

grouping ldp-peer-ref-container {
  description
    "A container containing an absolute reference to an LDP peer.";

  container peer {
    description
      "Reference to an LDP peer, by the LDP ID, which consists of
      the LSR ID and the Label Space ID.";
    uses ldp-peer-ref;
  } // peer
} // ldp-peer-ref

grouping peer-attributes {
  description
    "Peer configuration attributes, used in the per-peer setting
    can in the all-peer setting.";

  leaf session-ka-holdtime {
    type uint16 {
      range 45..3600;
    }
    units seconds;
    default 180;
    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;
        default 60;
        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 container, used in the per-peer setting
        can in the all-peer setting.";

    container authentication {
        description
            "Containing authentication information.";
        choice auth-type-selection {
            description
                "Options for expressing authentication setting.";
            case auth-key {
                leaf md5-key {
                    type string;
                    description
                        "MD5 Key string.";
                }
            }
        }
    }
} // authentication
} // peer-authentication

grouping peer-state-derived {
    description
        "The peer state information derived from the LDP protocol
        operatoins.";

    container label-advertisement-mode {
        config false;
        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;
  config false;
  description "Time to send the next KeepAlive message.";
}

container received-peer-state {
  config false;
  description
    "Operational state information learned from the peer.";

  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.";
      }
    }
  } // capability
} // received-peer-state

container session-holdtime {
  config false;
  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.";
    }
  }
}
```



```
    }
  }
  config false;
  description
    "Representing the operational status of the LDP session.";
  reference
    "RFC5036, Sec. 2.5.4.";
}

container tcp-connection {
  config false;
  description "TCP connection state.";
  leaf local-address {
    type inet:ip-address;
    description "Local address.";
  }
  leaf local-port {
    type inet:port-number;
    description "Local port number.";
  }
  leaf remote-address {
    type inet:ip-address;
    description "Remote address.";
  }
  leaf remote-port {
    type inet:port-number;
    description "Remote port number.";
  }
} // tcp-connection

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

container statistics {
  config false;
  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 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 and operational state data nodes
 */

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

  container mpls-ldp {
    presence
      "Enables the LDP protocol.";
    description
      "Containing configuration and operational data for the LDP
      protocol.";

    container global {
      description
        "Global attributes for LDP.";

      container capability {
        description
          "Containing the LDP capability data. The container is
          used for augmentations.";
        reference
          "RFC5036: Sec. 1.5.";
      } // capability

      container graceful-restart {
        description
          "Attributes for graceful restart.";
        leaf enable {
          type boolean;
          default false;
          description
            "Enable or disable graceful restart.";
        }
        leaf reconnect-time {
          type uint16 {
            range 10..1800;
          }
          units seconds;
          default 120;
          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;
      default 120;
      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;
      default 180;
      description
        "Specifies the time interval, in seconds, before the
         termination of the recovery phase.";
    }
  } // graceful-restart

  leaf lsr-id {
    type rt-types:router-id;
    description
      "Specify the value to act as the LDP LSR ID.
       If this attribute is not specified, LDP uses the router
       ID as determined by the system.";
  }

  container address-families {
    description
      "Per address family configuration and operational state.
       The address family can be either IPv4 or IPv6.";
    container ipv4 {
      presence
        "Present if IPv4 is enabled, unless the 'enable'
         leaf is set to 'false'";
      description
        "Containing data related to the IPv4 address family.";

      leaf enable {
        type boolean;
        default true;
        description
```



```
        "'true' to enable the address family.";
    }

    leaf label-distribution-controlmode {
        type enumeration {
            enum independent {
                description
                    "Independent label distribution control.";
            }
            enum ordered {
                description
                    "Ordered label distribution control.";
            }
        }
    }
    config false;
    description
        "Label distribution control mode.";
    reference
        "RFC5036: LDP Specification. Sec 2.6.";
}

// ipv4 bindings
container bindings {
    config false;
    description
        "LDP address and label binding information.";
    list address {
        key "address";
        description
            "List of address bindings learned by LDP.";
        leaf address {
            type inet:ipv4-address;
            description
                "The IPv4 address learned from an Address
                 message received from or advertised to a peer.";
        }
        uses binding-address-state-attributes;
    } // binding-address

    list fec-label {
        key "fec";
        description
            "List of FEC-label bindings learned by LDP.";
        leaf fec {
            type inet:ipv4-prefix;
            description
                "The prefix FEC value in the FEC-label binding,
                 learned in a Label Mapping message received from
```



```
        or advertised to a peer.";
    }
    uses binding-label-state-attributes;
  } // fec-label
} // bindings
} // ipv4
} // address-families
} // global

container discovery {
  description
    "Neighbor discovery configuration and operational state.";

  container interfaces {
    description
      "A list of interfaces for LDP Basic Discovery.";
    reference
      "RFC5036: LDP Specification. Sec 2.4.1.";

    uses basic-discovery-timers;

    list interface {
      key "name";
      description
        "List of LDP interfaces used for LDP Basic Discovery.";
      uses ldp-interface-ref;
      leaf next-hello {
        type uint16;
        units seconds;
        config false;
        description "Time to send the next hello message.";
      }
    }

    container address-families {
      description
        "Container for address families.";
      container ipv4 {
        presence
          "Present if IPv4 is enabled, unless the 'enable'
           leaf is set to 'false'";
        description
          "IPv4 address family.";

        leaf enable {
          type boolean;
          default true;
          description
            "Enable the address family on the interface.";
        }
      }
    }
  }
}
```



```
    }

    // ipv4
    container hello-adjacencies {
        config false;
        description
            "Containing a list of hello adjacencies.";

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

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

            uses adjacency-state-attributes;
            uses ldp-peer-ref-container;
        } // hello-adjacency
    } // hello-adjacencies
} // ipv4
} // address-families
} // list interface
} // interfaces

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

    leaf hello-holdtime {
        type uint16 {
            range 15..3600;
        }
        units seconds;
        default 45;
        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;
    default 15;
    description
        "The interval between consecutive LDP targeted Hello
        messages used in extended LDP discovery.";
}

container hello-accept {
    description
        "LDP policy to control the acceptance of extended
        neighbor discovery Hello messages.";

    leaf enable {
        type boolean;
        default false;
        description
            "'true' to accept; 'false' to deny.";
    }
} // hello-accept

container address-families {
    description
        "Container for address families.";
    container ipv4 {
        presence
            "Present if IPv4 is enabled.";
        description
            "IPv4 address family.";

        container hello-adjacencies {
            config false;
            description
                "Containing a list of hello adjacencies.";

            list hello-adjacency {
                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;
        uses ldp-peer-ref-container;
    } // hello-adjacency
} // hello-adjacencies

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.";
    }

    leaf enable {
        type boolean;
        default true;
        description
            "Enable the target.";
    }

    leaf local-address {
        type inet:ipv4-address;
        description
            "The local address used as the source address to
             send targeted Hello messages.
             If the value is not specified, the
             transport-address is used as the source
             address.";
    }
} // target
} // ipv4
} // address-families
} // targeted
} // discovery

container peers {
    description
        "Peers configuration attributes.";

    uses peer-authentication;
    uses peer-attributes;

    list peer {
        key "lsr-id label-space-id";
```



```
description
  "List of peers.";

leaf lsr-id {
  type rt-types:router-id;
  description
    "The LSR ID of the peer, to identify the globally
    unique LSR. This is the first four octets of the LDP
    ID. This leaf is used together with the leaf
    'label-space-id' to form the LDP ID.";
  reference
    "RFC5036. Sec 2.2.2.";
}
leaf label-space-id {
  type uint16;
  description
    "The Label Space ID of the peer, to identify a specific
    label space within the LSR. This is the last two
    octets of the LDP ID. This leaf is used together with
    the leaf 'lsr-id' to form the LDP ID.";
  reference
    "RFC5036. Sec 2.2.2.";
}

uses peer-authentication;
container capability {
  description
    "Per peer capability";
}

container address-families {
  description
    "Per-vrf per-af params.";
  container ipv4 {
    presence
      "Present if IPv4 is enabled.";
    description
      "IPv4 address family.";

    container hello-adjacencies {
      config false;
      description
        "Containing a list of hello adjacencies.";

      list hello-adjacency {
        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 if:interface-ref;
      description "Interface for this adjacency.";
    }
  } // hello-adjacency
} // hello-adjacencies
} // ipv4
} // address-families

  uses peer-state-derived;
} // list peer
} // peers
} // container mpls-ldp
}

/*
 * RPCs
 */
rpc mpls-ldp-clear-peer {
  description
    "Clears the session to the peer.";
  input {
    uses ldp-peer-ref {
      description
        "The LDP peer to be cleared. If this is not provided
        then all peers are cleared.
        The peer is identified by its LDP ID, which consists of
        the LSR ID and the Label Space ID.";
    }
  }
}

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 leafref {
            path "/rt:routing/rt:control-plane-protocols/"
              + "mpls-ldp/discovery/interfaces/interface/name";
          }
          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 {
    uses ldp-peer-ref {
      description
        "The LDP peer whose statistics are to be cleared.
        If this is not provided then all peers' statistics are
        cleared.
        The peer is identified by its LDP ID, which consists of
        the LSR ID and the Label Space ID.";
    }
  }
}

/*
 * 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-container;
}

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.";
  }
  choice hello-adjacency-type {
    description
      "Interface or targeted adjacency.";
    case targeted {
      container targeted {
```



```
        description
            "Targeted adjacency through LDP extended discovery.";
        leaf target-address {
            type inet:ip-address;
            description
                "The target adjacent address learned.";
        }
    } // targeted
}

case link {
    container link {
        description
            "Link adjacency through LDP basic discovery.";
        leaf next-hop-interface {
            type if:interface-ref;
            description
                "The interface connecting to the adjacent next hop.";
        }
        leaf next-hop-address {
            type inet:ip-address;
            must "../next-hop-interface" {
                description
                    "Applicable when interface is specified.";
            }
            description
                "IP address of the next hop. This can be IPv4 or IPv6
                address.";
        }
    } // link
}

}

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.";
    }
    leaf prefix {
        type inet:ip-prefix;
        description
            "The address prefix element of the FEC whose status
            has changed.";
    }
}

}
```



```
}
```

```
<CODE ENDS>
```

Figure 17

9.2. Extended

```
<CODE BEGINS> file "ietf-mpls-ldp-extended@2018-02-28.yang"

module ietf-mpls-ldp-extended {
  namespace "urn:ietf:params:xml:ns:yang:ietf-mpls-ldp-extended";
  prefix "ldp-ext";

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

  import ietf-routing {
    prefix "rt";
  }

  import ietf-key-chain {
    prefix "key-chain";
  }

  import ietf-mpls-ldp {
    prefix "ldp";
  }

  import ietf-interfaces {
    prefix "if";
  }

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

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

    Editor:   Rajiv Asati
```


<mailto:rajiva@cisco.com>

Editor: Xufeng Liu
<mailto:Xufeng_Liu@jabil.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 extended components for the management of Multi-Protocol Label Switching (MPLS) Label Distribution Protocol (LDP). It is also the model to be augmented for extended Multipoint LDP (mLDP).

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in [Section 4.c](#) of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

revision 2018-02-28 {

description

"Initial revision.";

reference

"RFC XXXX: YANG Data Model for MPLS LDP.";

}

/*

* Features

*/

feature dual-stack-transport-pereference {

description

"This feature indicates that the system allows to configure the transport connection pereference in a dual-stack setup.";

}


```
feature capability-end-of-lib {
  description
    "This feature indicates that the system allows to configure
    LDP end-of-lib 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 graceful-restart-helper-mode {
  description
    "This feature indicates that the system supports graceful
    restart helper mode.";
}

feature key-chain {
  description
    "This feature indicates that the system supports keychain for
    authentication.";
}

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-admin-down {
  description
    "This feature indicates that the system allows to
    administratively disable a peer.";
}
```



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

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

/*
 * Typedefs
 */
typedef neighbor-list-ref {
  type string;
  description
    "A type for a reference to a neighbor address list.
    The string value is the name identifier for uniquely
    identifying the referenced address list, which contains a list
    of addresses that a routing policy can applied. The definition
    of such an address list is outside the scope of this document.";
```



```
}

typedef prefix-list-ref {
  type string;
  description
    "A type for a reference to a prefix list.
    The string value is the name identifier for uniquely
    identifying the referenced prefix set, which contains a list
    of prefixes that a routing policy can applied. The definition
    of such a prefix set is outside the scope of this document.";
}

typedef peer-list-ref {
  type string;
  description
    "A type for a reference to a peer address list.
    The string value is the name identifier for uniquely
    identifying the referenced address list, which contains a list
    of addresses that a routing policy can applied. The definition
    of such an address list is outside the scope of this document.";
}

/*
 * Identities
 */

/*
 * Groupings
 */
grouping address-family-ipv4-augment {
  description "Augmentation to address family IPv4.";

  uses policy-container;

  leaf transport-address {
    type inet:ipv4-address;
    description
      "The transport address advertised in LDP Hello messages.
      If this value is not specified, the LDP LSR ID is used as
      the transport address.";
    reference
      "RFC5036. Sec. 3.5.2.";
  }
} // address-family-ipv4-augment

grouping authentication-keychain-augment {
  description "Augmentation to authentication to add keychain.";
```



```
    leaf key-chain {
      type key-chain:key-chain-ref;
      description
        "key-chain name.
        If not specified, no key chain is used.";
    }
  } // authentication-keychain-augment

  grouping capability-augment {
    description "Augmentation to capability.";

    container end-of-lib {
      if-feature capability-end-of-lib;
      description
        "Configure end-of-lib capability.";
      leaf enable {
        type boolean;
        default false;
        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;
      default false;
      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;
      default false;
      description
        "Enable upstream label assignment.";
    }
  }
} // capability-augment

grouping global-augment {
  description "Augmentation to global attributes.";
```



```
leaf igp-synchronization-delay {
  type uint16 {
    range "0 | 3..300";
  }
  units seconds;
  default 0;
  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.
    If the value is not specified, there is no delay.";
}
} // global-augment

grouping global-forwarding-nexthop-augment {
  description
    "Augmentation to global forwarding nexthop interfaces.";

  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 "name";
        description
          "List of LDP interfaces used for LDP Basic Discovery.";
        uses ldp:ldp-interface-ref;
        list address-family {
          key "afi";
          description
            "Per-vrf per-af params.";
          leaf afi {
            type ldp:ldp-address-family;
            description
              "Address family type value.";
          }
          leaf ldp-disable {
            type boolean;
            default false;
            description
              "'true' to disable LDP forwarding on the interface.";
          }
        }
      }
    }
  }
}
```



```
        } // address-family
    } // list interface
} // interfaces
} // forwarding-nexthop
} // global-forwarding-nexthop-augment

grouping graceful-restart-augment {
    description "Augmentation to graceful restart.";

    leaf helper-enable {
        if-feature graceful-restart-helper-mode;
        type boolean;
        default false;
        description
            "Enable or disable graceful restart helper mode.";
    }
} // graceful-restart-augment

grouping interface-address-family-ipv4-augment {
    description "Augmentation to interface address family IPv4.";

    leaf transport-address {
        type union {
            type enumeration {
                enum "use-global-transport-address" {
                    description
                        "Use the transport address set at the global level
                        common for all interfaces for this address family.";
                }
                enum "use-interface-address" {
                    description
                        "Use interface address as the transport address.";
                }
            }
        }
        type inet:ipv4-address;
    }
    default "use-global-transport-address";
    description
        "IP address to be advertised as the LDP transport address.";
}
} // interface-address-family-ipv4-augment

grouping interface-address-family-ipv6-augment {
    description "Augmentation to interface address family IPv6.";

    leaf transport-address {
        type union {
            type enumeration {
```



```
    enum "use-global-transport-address" {
        description
            "Use the transport address set at the global level
            common for all interfaces for this address family.";
    }
    enum "use-interface-address" {
        description
            "Use interface address as the transport address.";
    }
}
type inet:ipv6-address;
}
default "use-global-transport-address";
description
    "IP address to be advertised as the LDP transport address.";
}
} // interface-address-family-ipv6-augment

grouping interface-augment {
    description "Augmentation to interface.";

    uses ldp:basic-discovery-timers {
        if-feature per-interface-timer-config;
    }
    leaf igp-synchronization-delay {
        if-feature per-interface-timer-config;
        type uint16 {
            range "0 | 3..300";
        }
        units seconds;
        default 0;
        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.
            If the value is not specified, there is no delay.";
    }
} // interface-augment

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 filter outgoing label
       advertisements.
       If the value is not specified, no prefix filter
       is applied.";
  }
}
container accept {
  description
    "Label advertisement acceptance policies.";
  leaf prefix-list {
    type prefix-list-ref;
    description
      "Applies the prefix list to filter incoming label
       advertisements.
       If the value is not specified, no prefix filter
       is applied.";
  }
} // accept
} // label-policy
} // peer-af-policy-container

grouping peer-augment {
  description "Augmentation to each peer list entry.";

  leaf admin-down {
    if-feature per-peer-admin-down;
    type boolean;
    default false;
    description
      "'true' to disable the peer.";
  }

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

  uses ldp:peer-attributes {
    if-feature per-peer-session-attributes-config;
  }
} // peer-augment

grouping peers-augment {
  description "Augmentation to peers container.";
```



```
container session-downstream-on-demand {
  if-feature session-downstream-on-demand-config;
  description
    "Session downstream-on-demand attributes.";
  leaf enable {
    type boolean;
    default false;
    description
      "'true' if session downstream-on-demand is enabled.";
  }
  leaf peer-list {
    type peer-list-ref;
    description
      "The name of a peer ACL, to be applied to the
       downstream-on-demand sessions.
       If this value is not specified, no filter is applied to
       any downstream-on-demand sessions.";
  }
}

container dual-stack-transport-pereference {
  if-feature dual-stack-transport-pereference;
  description
    "The settings of peers to establish TCP connection in a
     dual-stack setup.";
  leaf max-wait {
    type uint16 {
      range "0..60";
    }
    default 30;
    description
      "The maximum wait time in seconds for preferred transport
       connection establishment. 0 indicates no preference.";
  }
}

container prefer-ipv4 {
  presence
    "Present if IPv4 is prefered for transport connection
     establishment, subject to the 'peer-list' in this
     container.";
  description
    "Uses IPv4 as the prefered address family for transport
     connection establishment, subject to the 'peer-list' in
     this container.
     If this container is not present, as a default, IPv6 is
     the prefered address family for transport connection
     establishment.";
  leaf peer-list {
    type peer-list-ref;
    description
```



```
        "The name of a peer ACL, to be applied to the IPv4
        transport connections.
        If this value is not specified, no filter is applied,
        and the IPv4 is preferred for all peers.";
    }
}
} // peers-augment

grouping policy-container {
    description
        "LDP policy attributes.";
    container label-policy {
        description
            "Label policy attributes.";
        container advertise {
            description
                "Label advertising policies.";
            container egress-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;
                    default false;
                    description
                        "'true' to enable explicit null.";
                }
            }
        }
        leaf prefix-list {
            type prefix-list-ref;
            description
                "Applies the prefix list to filter outgoing label
                advertisements.
                If the value is not specified, no prefix filter
                is applied.";
        }
    } // advertise
    container accept {
        description
            "Label advertisement acceptance policies.";
        leaf prefix-list {
            type prefix-list-ref;
            description
                "Applies the prefix list to filter incoming label
                advertisements.
```



```
        If the value is not specified, no prefix filter
        is applied.";
    }
} // accept
container assign {
    if-feature policy-label-assignment-config;
    description
        "Label assignment policies";
    container independent-mode {
        description
            "Independent label policy attributes.";
        leaf prefix-list {
            type prefix-list-ref;
            description
                "Assign labels according to certain prefixes.
                If the value is not specified, no prefix filter
                is applied (labels are assigned to all learned
                routes).";
        }
    } // independent-mode
    container ordered-mode {
        if-feature policy-ordered-label-config;
        description
            "Ordered label policy attributes.";
        leaf egress-prefix-list {
            type prefix-list-ref;
            description
                "Assign labels according to certain prefixes for
                egress LSR.";
        }
    } // ordered-mode
} // assign
} // label-policy
} // policy-container

/*
 * Configuration and state data nodes
 */
// Forwarding nexthop augmentation to the global tree
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:global" {
    description "Graceful forwarding nexthop augmentation.";
    uses global-forwarding-nexthop-augment;
}

// global/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
```



```
+ "ldp:global/ldp:address-families" {
  description "Global IPv6 augmentation.";

  container ipv6 {
    presence
      "Present if IPv6 is enabled, unless the 'enable'
       leaf is set to 'false'";
    description
      "Containing data related to the IPv6 address family.";

    leaf enable {
      type boolean;
      default true;
      description
        "'true' to enable the address family.";
    }

    uses policy-container;

    leaf transport-address {
      type inet:ipv6-address;
      mandatory true;
      description
        "The transport address advertised in LDP Hello messages.";
    }

    leaf label-distribution-controlmode {
      type enumeration {
        enum independent {
          description
            "Independent label distribution control.";
        }
        enum ordered {
          description
            "Ordered label distribution control.";
        }
      }
      config false;
      description
        "Label distribution control mode.";
      reference
        "RFC5036: LDP Specification. Sec 2.6.";
    }
  }

  // ipv6 bindings
  container bindings {
    config false;
    description
```



```
    "LDP address and label binding information.";
  list address {
    key "address";
    description
      "List of address bindings learned by LDP.";
    leaf address {
      type inet:ipv6-address;
      description
        "The IPv6 address learned from an Address
        message received from or advertised to a peer.";
    }
    uses ldp:binding-address-state-attributes;
  } // binding-address

  list fec-label {
    key "fec";
    description
      "List of FEC-label bindings learned by LDP.";
    leaf fec {
      type inet:ipv6-prefix;
      description
        "The prefix FEC value in the FEC-label binding,
        learned in a Label Mapping message received from
        or advertised to a peer.";
    }
    uses ldp:binding-label-state-attributes;
  } // fec-label
} // bindings
} // ipv6
}

// discovery/interfaces/interface/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:discovery/ldp:interfaces/ldp:interface/"
+ "ldp:address-families" {
  description "Interface IPv6 augmentation.";

  container ipv6 {
    presence
      "Present if IPv6 is enabled, unless the 'enable'
      leaf is set to 'false'";
    description
      "IPv6 address family.";

    leaf enable {
      type boolean;
      default true;
      description

```



```
        "Enable the address family on the interface.";
    }

    // ipv6
    container hello-adjacencies {
        config false;
        description
            "Containing a list of hello adjacencies.";

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

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

            uses ldp:adjacency-state-attributes;
            uses ldp:ldp-peer-ref-container;
        } // hello-adjacency
    } // hello-adjacencies
} // ipv6
}

// discovery/targeted/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:discovery/ldp:targeted/ldp:address-families" {
    description "Targeted discovery IPv6 augmentation.";

    container ipv6 {
        presence
            "Present if IPv6 is enabled.";
        description
            "IPv6 address family.";

        container hello-adjacencies {
            config false;
            description
                "Containing a list of hello adjacencies.";

            list hello-adjacency {
                key "local-address adjacent-address";
                config false;
                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 ldp:adjacency-state-attributes;
    uses ldp:ldp-peer-ref-container;
  } // hello-adjacency
} // hello-adjacencies

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.";
  }
  leaf enable {
    type boolean;
    default true;
    description
      "Enable the target.";
  }
  leaf local-address {
    type inet:ipv6-address;
    description
      "The local address used as the source address to send
       targeted Hello messages.
       If the value is not specified, the transport-address
       is used as the source address.";
  }
} // target
} // ipv6
}

// /peers/peer/state/address-families/ipv6
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
```



```
+ "ldp:peers/ldp:peer/ldp:address-families" {
  description "Peer state IPv6 augmentation.";

  container ipv6 {
    presence
      "Present if IPv6 is enabled.";
    description
      "IPv6 address family.";

    container hello-adjacencies {
      config false;
      description
        "Containing a list of hello adjacencies.";

      list hello-adjacency {
        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 ldp:adjacency-state-attributes;

      leaf interface {
        type if:interface-ref;
        description "Interface for this adjacency.";
      }
    } // hello-adjacency
  } // hello-adjacencies
} // ipv6
}

/*
 * Configuration data and operational state data nodes
 */
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:global" {
  description "Graceful restart augmentation.";
  uses global-augment;
}
```



```
augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:global/ldp:capability" {
  description "Capability augmentation.";
  uses capability-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:global/ldp:graceful-restart" {
  description "Graceful restart augmentation.";
  uses graceful-restart-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:global/ldp:address-families/ldp:ipv4" {
  description "Address family IPv4 augmentation.";
  uses address-family-ipv4-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:discovery/ldp:interfaces/ldp:interface" {
  description "Interface augmentation.";
  uses interface-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:discovery/ldp:interfaces/ldp:interface/"
+ "ldp:address-families/ldp:ipv4" {
  description "Interface address family IPv4 augmentation.";
  uses interface-address-family-ipv4-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:discovery/ldp:interfaces/ldp:interface/"
+ "ldp:address-families/ldp-ext:ipv6" {
  description "Interface address family IPv6 augmentation.";
  uses interface-address-family-ipv6-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:discovery/ldp:targeted/ldp:hello-accept" {
  description "Targeted discovery augmentation.";
  leaf neighbor-list {
    if-feature policy-targeted-discovery-config;
    type neighbor-list-ref;
    description
      "The name of a neighbor ACL, to accept Hello messages from
       LDP peers as permitted by the neighbor-list policy.
       If this value is not specified, targeted Hello messages from
```



```
        any source are accepted.";
    }
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:peers" {
    description "Peers augmentation.";
    uses peers-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:peers/ldp:authentication/ldp:auth-type-selection" {
    if-feature key-chain;
    description "Peers authentication augmentation.";
    case auth-key-chain {
        uses authentication-keychain-augment;
    }
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:peers/ldp:peer" {
    description "Peer list entry augmentation.";
    uses peer-augment;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:peers/ldp:peer/ldp:authentication/"
+ "ldp:auth-type-selection" {
    if-feature key-chain;
    description "Peer list entry authentication augmentation.";
    case auth-key-chain {
        uses authentication-keychain-augment;
    }
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:peers/ldp:peer/ldp:address-families/ldp:ipv4" {
    description
        "Peer list entry IPv4 augmentation.";
    uses peer-af-policy-container;
}

augment "/rt:routing/rt:control-plane-protocols/ldp:mpls-ldp/"
+ "ldp:peers/ldp:peer/ldp:address-families/ldp-ext:ipv6" {
    description
        "Peer list entry IPv6 augmentation.";
    uses peer-af-policy-container;
}
```



```
}  
  
<CODE ENDS>
```

Figure 18

10. 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 \[RFC6536\]](#).

11. IANA Considerations

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

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

12. 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.

13. References

13.1. Normative References

[I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K.,
and R. Wilton, "Network Management Datastore
Architecture", [draft-ietf-netmod-revised-datastores-10](#)
(work in progress), January 2018.

[I-D.ietf-netmod-rfc6087bis]

Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", [draft-ietf-netmod-rfc6087bis-18](#) (work in progress), February 2018.

[I-D.ietf-rtgwg-ni-model]

Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Network Instances", [draft-ietf-rtgwg-ni-model-11](#) (work in progress), March 2018.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://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, <<https://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, <<https://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, <<https://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, <<https://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, <<https://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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6389] Aggarwal, R. and JL. Le Roux, "MPLS Upstream Label Assignment for LDP", [RFC 6389](#), DOI 10.17487/RFC6389, November 2011, <<https://www.rfc-editor.org/info/rfc6389>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [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, <<https://www.rfc-editor.org/info/rfc7552>>.
- [RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", [RFC 8022](#), DOI 10.17487/RFC8022, November 2016, <<https://www.rfc-editor.org/info/rfc8022>>.

13.2. Informative References

- [I-D.ietf-mpls-mldp-yang]
Raza, K., Krishnaswamy, S., Liu, X., Esale, S., Andersson, L., and J. Tantsura, "YANG Data Model for MPLS mLDP", [draft-ietf-mpls-mldp-yang-03](#) (work in progress), November 2017.
- [I-D.ietf-rtgwg-policy-model]
Qu, Y., Tantsura, J., Lindem, A., Liu, X., and A. Shaikh, "A YANG Data Model for Routing Policy Management", [draft-ietf-rtgwg-policy-model-02](#) (work in progress), March 2018.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", [RFC 4364](#), DOI 10.17487/RFC4364, February 2006, <<https://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, <<https://www.rfc-editor.org/info/rfc7307>>.

Appendix A. Additional Contributors

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

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

Authors' Addresses

Kamran Raza
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, ON K2K-3E8
CA
Email: skraza@cisco.com

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

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

Xufeng Liu
Jabil
USA
Email: xufeng_liu@jabil.com

Jeff Tantsura
Email: jefftant.ietf@gmail.com

Santosh Esale
Juniper Networks
USA
Email: sesale@juniper.net

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

Loa Andersson
Huawei Technologies
Email: loa@pi.nu

Himanshu Shah
Ciena Corporation
USA
Email: hshah@ciena.com

Matthew Bocci
Nokia
Email: matthew.bocci@nokia.com